Network Working Group
Internet-Draft
Intended status: Informational
Expires: 9 October 2025

F. G端nther
IBM Research Europe - Zurich
M. Thomson
Mozilla
C. A. Wood
Cloudflare

7 April 2025

Usage Limits on AEAD Algorithms
draft-irtf-cfrg-aead-limits-10

Abstract

An Authenticated Encryption with Associated Data (AEAD) algorithm provides confidentiality and integrity. Excessive use of the same key can give an attacker advantages in breaking these properties. This document provides simple guidance for users of common AEAD functions about how to limit the use of keys in order to bound the advantage given to an attacker. It considers limits in both single- and multi-key settings.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Crypto Forum Research Group mailing list (cfrg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=cfrg.

Source for this draft and an issue tracker can be found at https://github.com/cfrg/draft-irtf-cfrg-aead-limits.

Status of This Memo

This Internet-Draft will expire on 9 October 2025.

Table of Contents

1. Introduction

An Authenticated Encryption with Associated Data (AEAD) algorithm provides confidentiality and integrity. [RFC5116] specifies an AEAD as a function with four inputs -- secret key, nonce, plaintext, associated data (of which nonce, plaintext, and associated data can optionally be zero-length) -- that produces ciphertext output and an error code indicating success or failure. The ciphertext is typically composed of the encrypted plaintext bytes and an authentication tag.

The generic AEAD interface does not describe usage limits. Each AEAD algorithm does describe limits on its inputs, but these are formulated as strict functional limits, such as the maximum length of inputs, which are determined by the properties of the underlying AEAD composition. Degradation of the security of the AEAD as a single key is used multiple times is not given the same thorough treatment.

Effective limits can be influenced by the number of "users" of a given key. In the traditional setting, there is one key shared between two parties. Any limits on the maximum length of inputs or encryption operations apply to that single key. The attacker's goal is to break security (confidentiality or integrity) of that specific key. However, in practice, there are often many parties with independent keys, multiple sessions between two parties, and even many keys used within a single session due to rekeying. This multi-key security setting, often referred to as the multi-user setting in the academic literature, considers an attacker's advantage in breaking security of any of these many keys, further assuming the attacker may have done some offline work (measuring time, but not memory) to help break any key. As a result, AEAD algorithm limits may depend on offline work and the number of keys. However, given that a multi-key attacker does not target any specific key, acceptable advantages may differ from that of the single-key setting.

The number of times a single pair of key and nonce can be used might also be relevant to security. For some algorithms, such as AEAD_AES_128_GCM or AEAD_AES_256_GCM, this limit is 1 and using the same pair of key and nonce has serious consequences for both confidentiality and integrity; see [NonceDisrespecting]. Nonce-reuse resistant algorithms like AEAD_AES_128_GCM_SIV can tolerate a limited amount of nonce reuse. This document focuses on AEAD schemes requiring non-repeating nonces.

It is good practice to have limits on how many times the same key (or pair of key and nonce) are used. Setting a limit based on some measurable property of the usage, such as number of protected messages, amount of data transferred, or time passed ensures that it is easy to apply limits. This might require the application of simplifying assumptions. For example, TLS 1.3 and QUIC both specify limits on the number of records that can be protected, using the simplifying assumption that records are the same size; see Section 5.5 of [TLS] and Section 6.6 of [RFC9001].

Exceeding the determined usage limit for a single key can be avoided using rekeying. Rekeying can also provide a measure of forward and backward (post-compromise) security. [RFC8645] contains a thorough survey of rekeying and the consequences of different design choices. When considering rekeying, the multi-user limits SHOULD be applied.

Currently, AEAD limits and usage requirements are scattered among peer-reviewed papers, standards documents, and other RFCs. Determining the correct limits for a given setting is challenging as papers do not use consistent labels or conventions, and rarely apply any simplifications that might aid in reaching a simple limit.

The intent of this document is to collate all relevant information about the proper usage and limits of AEAD algorithms in one place. This may serve as a standard reference when considering which AEAD algorithm to use, and how to use it.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Notation

This document defines limitations in part using the quantities in Table 1 below.

Symbol	Description
n	AEAD block length (in bits), of the underlying block cipher
k	AEAD key length (in bits)
r	AEAD nonce length (in bits)
t	Size of the authentication tag (in bits)
L	Maximum length of each message, including both plaintext and AAD (in blocks)
s	Total plaintext length in all messages (in blocks)
q	Number of protected messages (AEAD encryption invocations)
v	Number of attacker forgery attempts (failed AEAD decryption invocations + 1)
p	Upper bound on adversary attack probability
o	Offline adversary work (measured in number of encryption and decryption queries)
u	Number of keys (multi-key setting only)
B	Maximum number of blocks encrypted by any key (multi-key setting only)

Table 1: Notation

For each AEAD algorithm, we define the chosen-plaintext confidentiality (IND-CPA) and ciphertext integrity (INT-CTXT) advantage roughly as the advantage an attacker has in breaking the corresponding classical security property for the algorithm. An IND-CPA attacker can query ciphertexts for arbitrary plaintexts. An INT-CTXT attacker can additionally query plaintexts for arbitrary ciphertexts. Moreover, we define the combined authenticated encryption advantage guaranteeing both confidentiality and integrity against an active attacker. Specifically:

* Confidentiality advantage (CA): The probability of an attacker succeeding in breaking the IND-CPA (confidentiality) properties of the AEAD scheme. In this document, the definition of confidentiality advantage roughly is the probability that an attacker successfully distinguishes the ciphertext outputs of the AEAD scheme from the outputs of a random function.

* Integrity advantage (IA): The probability of an attacker succeeding in breaking the INT-CTXT (integrity) properties of the AEAD scheme. In this document, the definition of integrity advantage roughly is the probability that an attacker is able to forge a ciphertext that will be accepted as valid.

* Authenticated Encryption advantage (AEA): The probability of an active attacker succeeding in breaking the authenticated-encryption properties of the AEAD scheme. In this document, the definition of authenticated encryption advantage roughly is the probability that an attacker successfully distinguishes the ciphertext outputs of the AEAD scheme from the outputs of a random function or is able to forge a ciphertext that will be accepted as valid.

Here, we consider advantages beyond distinguishing underyling primitives from their ideal instances, for example, a blockcipher from a random permutation (PRP advantage) or a pseudorandom function from a truly random function (PRF advantage).

See [AEComposition], [AEAD] for the formal definitions of and relations between IND-CPA (confidentiality), INT-CTXT (integrity), and authenticated encryption security (AE). The authenticated encryption advantage subsumes, and can be derived as the combination of, both CA and IA:

```
CA <= AEA
IA <= AEA
AEA <= CA + IA
```

Each application requires an individual determination of limits in order to keep CA and IA sufficiently small. For instance, TLS aims to keep CA below 2^-60 and IA below 2^-57 in the single-key setting; see Section 5.5 of [TLS].

4. Calculating Limits

Once upper bounds on CA, IA, or AEA are determined, this document defines a process for determining three overall operational limits:

* Confidentiality limit (CL): The number of messages an application can encrypt before giving the adversary a confidentiality advantage higher than CA.

* Integrity limit (IL): The number of ciphertexts an application can decrypt unsuccessfully before giving the adversary an integrity advantage higher than IA.

* Authenticated encryption limit (AEL): The combined number of messages and number of ciphertexts an application can encrypt or decrypt before giving the adversary an authenticated encryption advantage higher than AEA.

When limits are expressed as a number of messages an application can encrypt or decrypt, this requires assumptions about the size of messages and any authenticated additional data (AAD). Limits can instead be expressed in terms of the number of bytes, or blocks, of plaintext and maybe AAD in total.

To aid in translating between message-based and byte/block-based limits, a formulation of limits that includes a maximum message size (L) and the AEAD schemes' block length in bits (n) is provided.

All limits are based on the total number of messages, either the number of protected messages (q) or the number of forgery attempts (v); which correspond to CL and IL respectively.

Limits are then derived from those bounds using a target attacker probability. For example, given an integrity advantage of $IA = v * (8L / 2^{106})$ and a targeted maximum attacker success probability of $IA = p$, the algorithm remains secure, i.e., the adversary's advantage does not exceed the targeted probability of success, provided that $v <= (p * 2^{106}) / 8L$. In turn, this implies that $v <= (p * 2^{103}) / L$ is the corresponding limit.

To apply these limits, implementations can count the number of messages that are protected or rejected against the determined limits (q and v respectively). This requires that messages cannot exceed the maximum message size (L) that is chosen.

4.1. Approximations

This analysis assumes a message-based approach to setting limits. Implementations that use byte counting rather than message counting could use a maximum message size (L) of one to determine a limit for the number of protected messages (q) that can be applied with byte counting. This results in attributing per-message overheads to every byte, so the resulting limit could be significantly lower than necessary. Actions, like rekeying, that are taken to avoid the limit might occur more often as a result.

To simplify formulae, estimates in this document elide terms that contribute negligible advantage to an attacker relative to other terms.

In other respects, this document seeks to make conservative choices that err on the side of overestimating attacker advantage. Some of these assumptions are present in the papers that this work is based on. For instance, analyses are simplified by using a single message size that covers both AAD and plaintext. AAD can contribute less toward attacker advantage for confidentiality limits, so applications where AAD comprises a significant proportion of messages might find the estimates provided to be slightly more conservative than necessary to meet a given goal.

This document assumes the use of non-repeating nonces (in particular, non-zero-length nonces). The modes covered here are not robust if the same nonce and key are used to protect different messages, so deterministic generation of nonces from a counter or similar techniques is strongly encouraged. If an application cannot guarantee that nonces will not repeat, a nonce-misuse resistant AEAD like AES-GCM-SIV [SIV] is likely to be a better choice.

5. Single-Key AEAD Limits

This section summarizes the confidentiality and integrity bounds and limits for modern AEAD algorithms used in IETF protocols, including: AEAD_AES_128_GCM [RFC5116], AEAD_AES_256_GCM [RFC5116], AEAD_AES_128_CCM [RFC5116], AEAD_CHACHA20_POLY1305 [RFC8439], AEAD_AES_128_CCM_8 [RFC6655]. The limits in this section apply to using these schemes with a single key; for settings where multiple keys are deployed (for example, when rekeying within a connection), the limits in Section 6 SHOULD be used.

These algorithms, as cited, all define a nonce length (r) of 96 bits. Some definitions of these AEAD algorithms allow for other nonce lengths, but the analyses in this document all fix the nonce length to r = 96. Using other nonce lengths might result in different bounds; for example, [GCMProofs] shows that using a variable-length nonce for AES-GCM results in worse security bounds.

The CL and IL values bound the total number of encryption and forgery queries (q and v). Alongside each advantage value, we also specify these bounds.

5.1. Offline Work

Single-key analyses of different cipher modes typically concentrate on the advantage that an attacker might gain through the mode itself. These analyses assume that the underlying cipher is an ideal PRP or PRF, and we make the same assumptions here. But even an ideal PRP or PRF can be attacked through exhaustive key search (in the key length, k) given sufficient resources.

An attacker that is able to deploy sufficient offline resources (o) can increase their success probability independent of any usage. In even the best case, single key bounds are always limited to the maximum of the stated bound and

$$AEA \le o / 2^k$$

This constrains the security that can be achieved for modes that use smaller key sizes, depending on what assumptions can be made about attacker resources.

For example, if an attacker could be assumed to have the resources to perform in the order of 2^{80} AES operations, an attacker gains an attack probability of 2^{-48}. That might seem like it requires a lot of compute resources, but amount of compute could cost less than 1 million USD in 2025. That cost can only reduce over time, suggesting that a much greater advantage is likely achievable for a sufficiently motivated attacker.

5.2. AEAD_AES_128_GCM and AEAD_AES_256_GCM

The CL and IL values for AES-GCM are derived in [AEBounds], following [GCMProofs], and summarized below. For this AEAD, n = 128 (the AES block length) and t = 128 [GCM], [RFC5116]. In this example, the length s is the sum of AAD and plaintext (in blocks of 128 bits), as described in [GCMProofs].

5.2.1. Confidentiality Limit

Applying Corollary 3 from [GCMProofs], assuming AES behaves like a random permutation, the following bound applies:

$$CA \le ((s + q + 1)^2) / 2^{129}$$

This implies the following usage limit:

$$q + s \le p^{(1/2)} * 2^{(129/2)} - 1$$

Which, for a message-based protocol with s <= q * L, if we assume that every packet is size L (in blocks of 128 bits), produces the limit:

q <= (p^(1/2) * 2^(129/2) - 1) / (L + 1)

5.2.2. Integrity Limit

Applying Equation (22) from [GCMProofs], in which the assumption of s + q + v < 2^64 ensures that the delta function cannot produce a value greater than 2, the following bound applies:

IA <= 2 * (v * (L + 1)) / 2^128

For the assumption of s + q + v < 2^64, observe that this applies when p > L / 2^63. s + q <= q * (L + 1) is always small relative to 2^64 if the same advantage is applied to the confidentiality limit on q.

This produces the following limit:

v <= min(2^64, (p * 2^127) / (L + 1))

5.3. AEAD_CHACHA20_POLY1305

The known single-user result for AEAD_CHACHA20_POLY1305 [ChaCha20Poly1305-MU] (correcting the prior one from [ChaCha20Poly1305-SU]) gives a combined AE limit, which we separate into confidentiality and integrity limits below. For this AEAD, n = 512 (the ChaCha20 block length), k = 256, and t = 128; the length L’ is the sum of AAD and plaintext (in Poly1305 blocks of 128 bits), see [ChaCha20Poly1305-MU].

5.3.1. Confidentiality Limit

CA <= 0

This implies there is no limit beyond the PRF security of the underlying ChaCha20 block function.

5.3.2. Integrity Limit

IA <= (v * (L’ + 1)) / 2^103

This implies the following limit:

v <= (p * 2^103) / (L’ + 1)

5.4. AEAD_AES_128_CCM

The CL and IL values for AEAD_AES_128_CCM are derived from [CCM-ANALYSIS] and specified in the QUIC-TLS mapping specification [RFC9001]. This analysis uses the total number of underlying block cipher operations to derive its bound. For CCM, this number is the sum of: the length of the associated data in blocks, the length of the ciphertext in blocks, the length of the plaintext in blocks, plus 1.

In the following limits, this is simplified to a value of twice the length of the packet in blocks, i.e., 2L represents the effective length, in number of block cipher operations, of a message with L blocks. This simplification is based on the observation that common applications of this AEAD carry only a small amount of associated data compared to ciphertext. For example, QUIC has 1 to 3 blocks of AAD.

For this AEAD, n = 128 (the AES block length) and t = 128.

5.4.1. Confidentiality Limit

```
CA <= (2L * q)^2 / 2^n
   <= (2L * q)^2 / 2^128
```

This implies the following limit:

```
q <= sqrt(p) * 2^63 / L
```

5.4.2. Integrity Limit

```
IA <= v / 2^t + (2L * (v + q))^2 / 2^n
   <= v / 2^128 + (2L * (v + q))^2 / 2^128
```

This implies the following limit:

```
v + (2L * (v + q))^2 <= p * 2^128
```

In a setting where v or q is sufficiently large, v is negligible compared to (2L * (v + q))^2, so this this can be simplified to:

```
v + q <= sqrt(p) * 2^63 / L
```

5.5. AEAD_AES_128_CCM_8

The analysis in [CCM-ANALYSIS] also applies to this AEAD, but the reduced tag length of 64 bits changes the integrity limit calculation considerably.

$$
\begin{aligned}
IA &<= v / 2^t + (2L * (v + q))^2 / 2^n \\
&<= v / 2^{64} + (2L * (v + q))^2 / 2^{128}
\end{aligned}
$$

This results in reducing the limit on v by a factor of 2^64.

$$v * 2^{64} + (2L * (v + q))^2 <= p * 2^{128}$$

Note that, to apply this result, two inequalities can be produced, with the first applied to determine v, then applying the second to find q:

$$
\begin{aligned}
&v * 2^{64} <= p * 2^{127} \\
&(2L * (v + q))^2 <= p * 2^{127}
\end{aligned}
$$

This approach produces much smaller values for v than for q. Alternative allocations tend to greatly reduce q without significantly increasing v.

5.6. Single-Key Examples

An example protocol might choose to aim for a single-key CA and IA that is at most 2^-50. (This in particular limits offline work to o <= 2^(k-50), see Section 5.1.) If the messages exchanged in the protocol are at most a common Internet MTU of around 1500 bytes, then a value for L might be set to 2^7. Table 2 shows limits for q and v that might be chosen under these conditions.

AEAD	Maximum q	Maximum v
AEAD_AES_128_GCM	2^32.5	2^64
AEAD_AES_256_GCM	2^32.5	2^64
AEAD_CHACHA20_POLY1305	n/a	2^46
AEAD_AES_128_CCM	2^30	2^30
AEAD_AES_128_CCM_8	2^30.4	2^13

Table 2: Example single-key limits; see text for parameter details

AEAD_CHACHA20_POLY1305 provides no limit to q based on the provided single-user analyses.

The limit for q on AEAD_AES_128_CCM and AEAD_AES_128_CCM_8 is reduced due to a need to reduce the value of q to ensure that IA does not exceed the target. This assumes equal proportions for q and v for AEAD_AES_128_CCM. AEAD_AES_128_CCM_8 permits a much smaller value of v due to the shorter tag, which permits a higher limit for q.

Some protocols naturally limit v to 1, such as TCP-based variants of TLS, which terminate sessions on decryption failure. If v is limited to 1, q can be increased to 2^31 for both CCM AEADs.

6. Multi-Key AEAD Limits

In the multi-key setting, each user is assumed to have an independent and uniformly distributed key, though nonces may be re-used across users with some very small probability. The success probability in attacking one of these many independent keys can be generically bounded by the success probability of attacking a single key multiplied by the number of keys present [MUSecurity], [GCM-MU]. Absent concrete multi-key bounds, this means the attacker advantage in the multi-key setting is the product of the single-key advantage and the number of keys.

This section summarizes the confidentiality and integrity bounds and limits for the same algorithms as in Section 5 for the multi-key setting. The CL and IL values bound the total number of encryption and forgery queries (q and v). Alongside each value, we also specify these bounds.

6.1. AEAD_AES_128_GCM and AEAD_AES_256_GCM

Concrete multi-key bounds for AEAD_AES_128_GCM and AEAD_AES_256_GCM exist due to Theorem 4.3 in [GCM-MU2], which covers protocols with nonce randomization, like TLS 1.3 [TLS] and QUIC [RFC9001]. Here, the full nonce is XORed with a secret, random offset. The bound for nonce randomization was further improved in [ChaCha20Poly1305-MU].

Results for AES-GCM with random, partially implicit nonces [RFC5288] are captured by Theorem 5.3 in [GCM-MU2], which apply to protocols such as TLS 1.2 [RFC5246]. Here, the implicit part of the nonce is a random value, of length at least 32 bits and fixed per key, while we assume that the explicit part of the nonce is chosen using a non-repeating process. The full nonce is the concatenation of the two parts. This produces similar limits under most conditions. Note that implementations that choose the explicit part at random have a higher chance of nonce collisions and are not considered for the limits in this section.

For this AEAD, n = 128 (the AES block length), t = 128, and r = 96; the key length is k = 128 or k = 256 for AEAD_AES_128_GCM and AEAD_AES_256_GCM respectively.

6.1.1. Authenticated Encryption Security Limit

Protocols with nonce randomization have a limit of:

AEA <= (q+v)*L*B / 2^127

This implies the following limit:

q + v <= p * 2^127 / (L * B)

This assumes that B is much larger than 100; that is, each user enciphers significantly more than 1600 bytes of data. Otherwise, B should be increased by 161 for AEAD_AES_128_GCM and by 97 for AEAD_AES_256_GCM. For AEAD_AES_128_GCM, it further assumes o <= 2^70, otherwise a term in the order of o / 2^120 starts dominating.

Protocols with random, partially implicit nonces have the following limit, which is similar to that for nonce randomization:

AEA <= (((q+v)*o + (q+v)^2) / 2^(k+26)) + ((q+v)*L*B / 2^127)

The first term is negligible if k = 256; this implies the following simplified limits:

AEA <= (q+v)*L*B / 2^127
q + v <= p * 2^127 / (L * B)

For k = 128, assuming o <= q + v (i.e., that the attacker does not spend more work than all legitimate protocol users together), the limits are:

AEA <= (((q+v)*o + (q+v)^2) / 2^154) + ((q+v)*L*B / 2^127)
q + v <= min(sqrt(p) * 2^76, p * 2^126 / (L * B))

6.1.2. Confidentiality Limit

The confidentiality advantage is essentially dominated by the same term as the AE advantage for protocols with nonce randomization:

CA <= q*L*B / 2^127

This implies the following limit:

q <= p * 2^127 / (L * B)

Similarly, the limits for protocols with random, partially implicit nonces are:

```
CA <= ((q*o + q^2) / 2^(k+26)) + (q*L*B / 2^127)
q <= min( sqrt(p) * 2^76,  p * 2^126 / (L * B) )
```

6.1.3. Integrity Limit

There is currently no dedicated integrity multi-key bound available for AEAD_AES_128_GCM and AEAD_AES_256_GCM. The AE limit can be used to derive an integrity limit as:

```
IA <= AEA
```

Section 6.1.1 therefore contains the integrity limits.

6.2. AEAD_CHACHA20_POLY1305

Concrete multi-key bounds for AEAD_CHACHA20_POLY1305 are given in Theorem 7.2 in [ChaCha20Poly1305-MU], covering protocols with nonce randomization like TLS 1.3 [TLS] and QUIC [RFC9001].

For this AEAD, n = 512 (the ChaCha20 block length), k = 256, t = 128, and r = 96; the length (L’) is the sum of AAD and plaintext (in Poly1305 blocks of 128 bits).

6.2.1. Authenticated Encryption Security Limit

Protocols with nonce randomization have a limit of:

```
AEA <= (v * (L’ + 1)) / 2^103
```

It implies the following limit:

```
v <= (p * 2^103) / (L’ + 1)
```

Note that this is the same limit as in the single-user case except that the total number of forgery attempts (v) and maximum message length in Poly1305 blocks (L’) is calculated across all used keys.

6.2.2. Confidentiality Limit

While the AE advantage is dominated by the number of forgery attempts v, those are irrelevant for the confidentiality advantage. The relevant limit for protocols with nonce randomization becomes dominated, at a very low level, by the adversary’s offline work o and the number of protected messages q across all used keys:

CA <= (o + q) / 2^247

This implies the following simplified limit, which for most reasonable values of p is dominated by a technical limitation of approximately q = 2^100:

q <= min(p * 2^247 - o, 2^100)

6.2.3. Integrity Limit

The AE limit for AEAD_CHACHA20_POLY1305 essentially is the integrity (multi-key) bound. The former hence also applies to the latter:

IA <= AEA

Section 6.2.1 therefore contains the integrity limits.

6.3. AEAD_AES_128_CCM and AEAD_AES_128_CCM_8

There are currently no concrete multi-key bounds for AEAD_AES_128_CCM or AEAD_AES_128_CCM_8. Thus, to account for the additional factor u, i.e., the number of keys, each p term in the confidentiality and integrity limits is replaced with p / u.

Note that this factor also applies to the generic exhaustive key search attack discussed for single-key analyses in Section 5.1.

The multi-key integrity limit for AEAD_AES_128_CCM is as follows.

v + q <= sqrt(p / u) * 2^63 / L

Likewise, the multi-key integrity limit for AEAD_AES_128_CCM_8 is as follows.

v * 2^64 + (2L * (v + q))^2 <= (p / u) * 2^128

6.4. Multi-Key Examples

An example protocol might choose to aim for a multi-key AEA, CA, and IA that is at most 2^-50. If the messages exchanged in the protocol are at most a common Internet MTU of around 1500 bytes, then a value for L might be set to 2^7. Table 3 shows limits for q and v across all keys that might be chosen under these conditions.

AEAD	Maximum q	Maximum v
AEAD_AES_128_GCM	2^69/B	2^69/B
AEAD_AES_256_GCM	2^69/B	2^69/B
AEAD_CHACHA20_POLY1305	2^100	2^46
AEAD_AES_128_CCM	2^30/sqrt(u)	2^30/sqrt(u)
AEAD_AES_128_CCM_8	2^30.9/sqrt(u)	2^13/u

Table 3: Example multi-key limits; see text for parameter details

The limits for AEAD_AES_128_GCM, AEAD_AES_256_GCM, AEAD_AES_128_CCM, and AEAD_AES_128_CCM_8 assume equal proportions for q and v. The limits for AEAD_AES_128_GCM, AEAD_AES_256_GCM and AEAD_CHACHA20_POLY1305 assume the use of nonce randomization, like in TLS 1.3 [TLS] and QUIC [RFC9001], and offline work limited to o <= 2^70.

The limits for AEAD_AES_128_GCM and AEAD_AES_256_GCM further depend on the maximum number (B) of 128-bit blocks encrypted by any single key. For example, limiting the number of messages (of size <= 2^7 blocks) to at most 2^20 (about a million) per key results in B of 2^27, which limits both q and v to 2^42 messages.

Only the limits for AEAD_AES_128_CCM and AEAD_AES_128_CCM_8 depend on the number of used keys (u), which further reduces them considerably. If v is limited to 1, q can be increased to 2^31/sqrt(u) for both CCM AEADs. (In particular offline work is assumed limited to o <= 2^(k-50) / u = 2^78 / u, see Section 5.1.)

7. Security Considerations

The different analyses of AEAD functions that this work is based upon generally assume that the underlying primitives are ideal. For example, that a pseudorandom function (PRF) used by the AEAD is indistinguishable from a truly random function or that a pseudorandom permutation (PRP) is indistinguishable from a truly random permutation. Thus, the advantage estimates assume that the attacker is not able to exploit a weakness in an underlying primitive.

Many of the formulae in this document depend on simplifying assumptions, from differing models, which means that results are not universally applicable. When using this document to set limits, it is necessary to validate all these assumptions for the setting in which the limits might apply. In most cases, the goal is to use assumptions that result in setting a more conservative limit, but this is not always the case. As an example of one such simplification, this document defines v as the total number of decryption queries leading to a successful forgery (that is, the number of failed forgery attempts plus one), whereas models usually include all forgery attempts when determining v.

The CA, IA, and AEA values defined in this document are upper bounds based on existing cryptographic research. Future analysis may introduce tighter bounds. Applications SHOULD NOT assume these bounds are rigid, and SHOULD accommodate changes.

Note that the limits in this document apply to the adversary’s ability to conduct a single successful forgery. For some algorithms and in some cases, an adversary’s success probability in repeating forgeries may be noticeably larger than that of the first forgery. As an example, [MF05] describes such multiple forgery attacks in the context of AES-GCM in more detail.

8. IANA Considerations

This document does not make any request of IANA.

9. References

9.1. Normative References

[AEAD] Rogaway, P., "Authenticated-Encryption with Associated-Data", September 2002, <https://web.cs.ucdavis.edu/~rogaway/papers/ad.pdf>.

[AEBounds] Luykx, A. and K. Paterson, "Limits on Authenticated Encryption Use in TLS", 8 March 2016, <https://eprint.iacr.org/2024/051.pdf>.

[AEComposition] Bellare, M. and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", July 2007, <https://eprint.iacr.org/2000/025.pdf>.

[CCM-ANALYSIS]
Jonsson, J., "On the Security of CTR + CBC-MAC", Springer Berlin Heidelberg, Lecture Notes in Computer Science pp. 76-93, DOI 10.1007/3-540-36492-7_7, ISBN ["9783540006220", "9783540364924"], 2003, <https://doi.org/10.1007/3-540-36492-7_7>.

[ChaCha20Poly1305-MU]
Degabriele, J. P., Govinden, J., G端nther, F., and K. G. Paterson, "The Security of ChaCha20-Poly1305 in the Multi-user Setting", 24 January 2023, <https://eprint.iacr.org/2023/085.pdf>.

[ChaCha20Poly1305-SU]
Procter, G., "A Security Analysis of the Composition of ChaCha20 and Poly1305", 11 August 2014, <https://eprint.iacr.org/2014/613.pdf>.

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, November 2007.

[GCM-MU] Bellare, M. and B. Tackmann, "The Multi-User Security of Authenticated Encryption: AES-GCM in TLS 1.3", 27 November 2017, <https://eprint.iacr.org/2016/564.pdf>.

[GCM-MU2] Hoang, V. T., Tessaro, S., and A. Thiruvengadam, "The Multi-user Security of GCM, Revisited: Tight Bounds for Nonce Randomization", 15 October 2018, <https://eprint.iacr.org/2018/993.pdf>.

[GCMProofs]
Iwata, T., Ohashi, K., and K. Minematsu, "Breaking and Repairing GCM Security Proofs", 1 August 2012, <https://eprint.iacr.org/2012/438.pdf>.

[MUSecurity]
Bellare, M., Boldyreva, A., and S. Micali, "Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements", May 2000, <https://cseweb.ucsd.edu/~mihir/papers/musu.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/rfc/rfc5116>.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/rfc/rfc6655>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/rfc/rfc8439>.

9.2. Informative References

[MF05] McGrew, D. A. and S. R. Fluhrer, "Multiple forgery attacks against Message Authentication Codes", 31 May 2005, <https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/CWC-GCM/multi-forge-01.pdf>.

[NonceDisrespecting] Bock, H., Zauner, A., Devlin, S., Somorovsky, J., and P. Jovanovic, "Nonce-Disrespecting Adversaries -- Practical Forgery Attacks on GCM in TLS", 17 May 2016, <https://eprint.iacr.org/2016/475.pdf>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/rfc/rfc5288>.

[RFC8645] Smyshlyaev, S., Ed., "Re-keying Mechanisms for Symmetric Keys", RFC 8645, DOI 10.17487/RFC8645, August 2019, <https://www.rfc-editor.org/rfc/rfc8645>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[SIV] Gueron, S., Langley, A., and Y. Lindell, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption", RFC 8452, DOI 10.17487/RFC8452, April 2019, <https://www.rfc-editor.org/rfc/rfc8452>.

[TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

Acknowledgments

In addition to the authors of papers performing analysis of ciphers, thanks are owed to Mihir Bellare, Daniel J. Bernstein, Scott Fluhrer, Thomas Fossati, John Mattsson, David McGrew, Yoav Nir, Thomas Pornin, and Alexander Tereschenko for helping making this document better.

Authors’ Addresses

Felix G端nther
IBM Research Europe - Zurich
Email: mail@felixguenther.info

Martin Thomson
Mozilla
Email: mt@lowentropy.net

Christopher A. Wood
Cloudflare
Email: caw@heapingbits.net