Limits on Authenticated Encryption Use in TLS

Atul Luykx and Kenneth G. Paterson

March 8, 2016

Abstract

This technical note presents limits on the security (as a function of the number of plaintext bytes encrypted and the number of forgery attempts made by an adversary) for the main Authenticated Encryption schemes available in TLS 1.2 and the draft of TLS 1.3. These limits are derived from security proofs for the considered schemes available in the literature. Our intention is to provide considered technical input to on-going discussions in the TLS Working Group of the IETF concerning, amongst other things, the necessity of adding a key update feature to the TLS 1.3 specification.

1 Summary

1.1 `ChaCha20+Poly1305`

The AE security of `ChaCha20+Poly1305` degrades proportionally with the number of forgery attempts, assuming nonces are used properly and the block function underlying ChaCha20 is secure as a pseudorandom function. If only a single forgery attempt is allowed, as is the case for TLS, then the 64-bit TLS sequence number will wrap before any significant security loss is incurred. If multiple forgery attempts are allowed, as is the case for DTLS, then adversaries will have AE success probability bounded by $v \cdot 2^{-93}$, with v being the number of forgery attempts. For example, adversaries making $v = 2^{60}$ forgery attempts will succeed with probability at most 2^{-33}.

1.2 `AES-GCM`

Limits on the amount of data that `AES-GCM` can process without needing a key change can be found in Table 1. Here we assume that AES is secure as a pseudorandom permutation and that nonces are used properly; in particular, we assume that the 8 octet `nonce_explicit` part of the `AES-GCM` nonce is unique for each encryption (for example, it is set to be the TLS sequence number).

Table 1: Lower bounds on the maximum amount of data that can be processed under one key for AES-GCM, given a certain success probability for a confidentiality attack. Integrity will not be breached with probability greater than $1/2^{57}$ assuming the data limits are respected, and at most 2^{60} verification attempts are made. Equation (7) was used to calculate the maximum number of records that can be processed, and the maximum amount of data was computed under the assumption that all records are of length 2^{14} Bytes.

Attack Success Probability	Max Records	Max Data (terabytes)
2^{-60}	$2^{24.5}$	0.3887
2^{-50}	$2^{29.5}$	12.44
2^{-40}	$2^{34.5}$	398.1
2^{-30}	$2^{39.5}$	12,738
2^{-20}	$2^{44.5}$	407,619
2^{-10}	$2^{49.5}$	1.304×10^{7}

In this table, the attack considered is one in which adversaries attempt to determine which one of two equal length messages was encrypted by `AES-GCM` (an IND-CPA attack); for the quoted figures, ciphertext integrity will not be breached with probability greater than $1/2^{57}$. Bounds on AE security can then be derived by adding $1/2^{57}$ to the bounds from Table 1. Section 2 provides further rationale for considering these types of attack, and also explains what the bounds mean, why there is a difference between `ChaCha20+Poly1305` and `AES-GCM`, and gives an overview of known vulnerabilities when the limits and assumptions are not respected.

1.3 Security Degradation for Multiple Keys

The computations in this text focus on security degradation under a *single* key. However, security also degrades proportional to the number of keys used, as described by Bernstein [5] and Chatterjee, Menezes, and Sarkar [8, 9]. Concretely, if k keys are used, then attacker success probability increases by a factor k. Therefore, all success probabilities in this text should be multiplied by k, where k is the number of keys used in all TLS connections that may be of interest to the adversary.

2 Explanation

2.1 Primitives and Modes

Efficient symmetric-key cryptographic algorithms can roughly be divided into two groups: primitives and modes of operation. Primitives, such as block ciphers or pseudorandom functions, are relatively simple (though not simple to design!) and can be regarded as attempting to approximate ideal mathematical objects, such as random permutations or functions. Examples of primitives used in the TLS record layer are AES and the block function underlying ChaCha20.

Primitives on their own do not provide sufficient security for TLS's needs. They must be used in modes of operation, which in turn enable secure communication. For example, `AES-GCM` is a mode of operation for AES; `ChaCha20+Poly1305` can be viewed as a mode of operation for a certain pseudorandom function (PRF). Both modes enable authenticated encryption, which the underlying primitives on their own do not.

The quality of the primitives is determined through extensive cryptanalysis, and confidence in how well they approximate the ideal mathematical object only increases as a function of how much the primitives have been studied. In contrast, modes of operation do not rely so much on maturity to create confidence in the security they provide; rather, one can formally *reduce* the security of modes of operation to that of the underlying primitive. This means that any attack against the mode with a given complexity can be converted into an attack against the primitive with a related complexity, and so any confidence in the primitive can be translated to confidence in the mode.

For well-designed primitives, the best attacks do not improve significantly when adversaries have access to more plaintext-ciphertext pairs: it matters little if you have twenty or 2^{20} plaintext-ciphertext pairs, you will not improve your attack against AES. However, the security of modes of operation *could* start to degrade with extended use, particularly when the security reduction from modes to primitives is not "tight".

If we look at `ChaCha20+Poly1305` for example, then its security proof in [18] establishes a fairly tight reduction from the mode to the underlying primitive, a PRF. This means there is no essential loss of security when going from mode to primitive. However, the same is not true for `AES-GCM` because of particular characteristics of this mode's construction.

2.2 Analytic Approach

In our analysis, to simplify matters, we will assume that AES with a random key is a uniform random permutation and that the PRF underlying `ChaCha20+Poly1305` with a random key is a uniform random function. This means that neither primitive can be distinguished from its "ideal" version, when keyed with a uniformly random value.

Making this assumption enables us to focus on the quality of the reductions from the modes to the underlying primitives without becoming entangled in too many details. In particular, the assumptions rule out attacks based on key recovery for the underlying primitives, and remove the dependence of success probabilities on running times of adversaries. This allows us to examine the relationship between attack success probability and the amount of encrypted records available to the adversary.

The assumptions can be relaxed at the cost of a more involved analysis. This analysis would introduce additional parameters relating to the distinguishing advantage of adversaries against the underlying primitives and their running times.

2.3 Security Notions

Here we are concerned with the security of `ChaCha20+Poly1305` and `AES-GCM` in the Authenticated Encryption (AE) sense. This is a strong and very conservative notion of security that covers a broad range of attacks that we wish to protect against in TLS. The AE security notion can be subdivided into two other notions: IND-CPA security and INT-CTXT security; their combination is equivalent to AE security.

The first, IND-CPA security, is a confidentiality notion, which measures how well an adversary can distinguish encryptions of different messages of the same lengths. In particular, IND-CPA captures the intuition that whatever properties adversaries can determine about the messages given the ciphertext, they could have determined without the ciphertext.

The second, INT-CTXT, is an integrity notion which measures the success of an adversary in creating fresh ciphertexts which are accepted as genuine upon decryption. In this notion, we measure the adversary's success in terms of the number of verification queries (trial decryptions) it is permitted to make, denoted v.

Note that in TLS, we can in fact set $v = 1$, since a failed verification query leads to the termination of the TLS connection and the disposal of the connection keys. However the same is not true in DTLS, and many verification attempts would be tolerated in a typical attack scenario.

Bounds on success probability for IND-CPA security and for INT-CTXT security (for a given v) can be *added* to produce bounds on AE security (for a given v).

2.4 Attacks

If the limits and assumptions in this text are not respected, then attacks are possible. Repeating a nonce with either `GCM` or `ChaCha20+Poly1305` means attackers can determine the XOR of two messages using only the ciphertext, and, in the case of `GCM`, partial key recovery is possible [13]. Successful forgeries also allow for partial key recovery attacks on `GCM` [10]; see Abdelraheem et al. [1, 2] for an overview of forgery attacks on `GCM`. Outputting multiple decryption errors [6, 7] or decrypting the ciphertext before the integrity check is complete [3, 4] could also result in vulnerabilities.

3 Computing the Bounds

The analysis by Procter [18] gives a security reduction for `ChaCha20+Poly1305` which tightly relates confidentiality to security of the underlying PRF, and shows an integrity degradation of $v \cdot 2^{-93}$ where v is the number of verification queries permitted, assuming all messages have length 2^{14} Bytes.

Our focus henceforth is on `AES-GCM`. In the analysis that follows, we set $v = 2^{60}$ to cater for DTLS as well as TLS, though the results would be materially the same with $v = 1$.

RFC 5288 specifies that `AES-GCM` use a 12-octet nonce, with 4 octets being a salt that is set from either the `client_write_IV` (when the client is sending) or the `server_write_IV` (when the server is sending), and the remaining 8 octets, called `nonce_explicit`, being required to be distinct for each distinct invocation of the GCM encrypt function for any fixed key. The `nonce_explicit` field may be set to the 64-bit TLS sequence number, but this is not required by RFC 5288.

In our further analysis, we assume that all the nonces are unique. Note that this is unlikely to be the case when the number of encryptions exceeds 2^{32} if `nonce_explicit` is selected at random for each encryption.

For `AES-GCM` we use the currently best known bounds provided by Iwata et al. [11, 12]. These bounds correct those in the original proof of security for `AES-GCM` by McGrew and Viega [14, 15]. Note that the security bound improvement proposed by Niwa et al. [16, 17] does not apply to the way `AES-GCM` is used in TLS, since it only holds when the construction is used with nonce-length not equal to 96 bits, which is not an option in TLS.

Table 2: Notation

Parameter	Description
n	Block size, 128 bits
τ	Tag size, 128 bits
ℓ	input length in blocks, 2^{10} blocks = 2^{14} Bytes
σ	total plaintext length in blocks
q	number of encryption queries
v	number of verification attempts

Starting with INT-CTXT (integrity), the best bound for `AES-GCM` can be found in equation (22) from Iwata et al.'s extended paper [12]:

$$\frac{v(\ell+1)}{2^\tau} \cdot \delta_n(\sigma + q + v + 1)\,, \tag{1}$$

with the notation explained in Table 2, and

$$\delta_n(x) := \frac{1}{\left(1 - \frac{x-1}{2^n}\right)^{x/2}}\,. \tag{2}$$

Assuming that $\sigma + q + v + 1 \leq 2^{64}$, then as pointed out by Iwata et al., we have that $\delta_n(\sigma + q + v + 1) \leq 2$, and we get an upper bound of

$$2\frac{v(\ell+1)}{2^\tau}\,. \tag{3}$$

So if σ, q, and v are not greater than 2^{60}, we have that the success probability of any attacker in breaking the integrity of `AES-GCM` is at most

$$2\frac{2^{60}(2^{10}+1)}{2^{128}} = \frac{1}{2^{57}} + \frac{1}{2^{67}}\,. \tag{4}$$

Corollary 3 from Iwata et al.'s papers establishes the following IND-CPA (confidentiality) bound for `AES-GCM`:

$$\frac{(\sigma+q+1)^2}{2^{n+1}}\,. \tag{5}$$

Since $\sigma \leq q\ell$, we get

$$\frac{(\sigma+q+1)^2}{2^{n+1}} \leq \frac{(q(\ell+1)+1)^2}{2^{n+1}}\,, \tag{6}$$

hence if we want to bound attack success probability by ϵ, we get

$$\frac{(q(\ell+1)+1)^2}{2^{n+1}} \leq \epsilon \quad \text{or} \quad q \leq \frac{\sqrt{2^{n+1}\epsilon}-1}{\ell+1}\,. \tag{7}$$

Plugging in the numbers, we get the bounds shown in Table 1 for `AES-GCM`.

References

[1] Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted Polynomials and Forgery Attacks on GCM. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 762–786. Springer, 2015.

[2] Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted Polynomials and Forgery Attacks on GCM. *IACR Cryptology ePrint Archive*, 2015:1224, 2015.

[3] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[4] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. *IACR Cryptology ePrint Archive*, 2014:144, 2014.

[5] Daniel J. Bernstein. 2015.11.20: Break a dozen secret keys, get a million more for free. The cr.yp.to blog, 2015. `https://blog.cr.yp.to/20151120-batchattacks.html`.

[6] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On Symmetric Encryption with Distinguishable Decryption Failures. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 367–390. Springer, 2013.

[7] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On Symmetric Encryption with Distinguishable Decryption Failures. *IACR Cryptology ePrint Archive*, 2013:433, 2013.

[8] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another Look at Tightness. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 293–319. Springer, 2011.

[9] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another Look at Tightness. *IACR Cryptology ePrint Archive*, 2011:442, 2011.

[10] Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008.

[11] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer Berlin Heidelberg, 2012.

[12] Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and Repairing GCM Security Proofs. *IACR Cryptology ePrint Archive*, 2012:438, 2012.

[13] Antoine Joux. Authentication Failures in NIST Version of GCM. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/Joux_comments.pdf`, 2006. Date accessed 2016.02.20.

[14] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

[15] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode of Operation (Full Version). *IACR Cryptology ePrint Archive*, 2004:193, 2004.

[16] Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. GCM Security Bounds Reconsidered. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 385–407. Springer, 2015.

[17] Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. GCM Security Bounds Reconsidered. *IACR Cryptology ePrint Archive*, 2015:214, 2015.

[18] Gordon Procter. A Security Analysis of the Composition of ChaCha20 and Poly1305. *IACR Cryptology ePrint Archive*, 2014:613, 2014.