# Authentication Key Recovery on Galois/Counter Mode (GCM)

John Mattsson and Magnus Westerlund

Ericsson Research, Stockholm, Sweden
{firstname.lastname}@ericsson.com

**Abstract.** GCM is used in a vast amount of security protocols and is quickly becoming the de facto mode of operation for block ciphers due to its exceptional performance. In this paper we analyze the NIST standardized version (SP 800-38D) of GCM, and in particular the use of short tag lengths. We show that feedback of successful or unsuccessful forgery attempts is almost always possible, contradicting the NIST assumptions for short tags. We also provide a complexity estimation of Ferguson's authentication key recovery method on short tags, and suggest several novel improvements to Fergusons's attacks that significantly reduce the security level for short tags. We show that for many truncated tag sizes; the security levels are far below, not only the current NIST requirement of 112-bit security, but also the old NIST requirement of 80-bit security. We therefore strongly recommend NIST to revise SP 800-38D.

**Keywords.** Secret-key Cryptography, Message Authentication Codes, Block Ciphers, Cryptanalysis, Galois/Counter Mode, GCM, Authentication Key Recovery, AES-GCM, Suite B, IPsec, ESP, SRTP, Re-forgery.

## 1 Introduction

Galois/Counter Mode (GCM) [1] is quickly becoming the de facto mode of operation for block ciphers. GCM is included in the NSA Suite B set of cryptographic algorithms [2], and AES-GCM is the benchmark algorithm for the AEAD competition CAESAR [3]. Together with Galois Message Authentication Code (GMAC), GCM is used in a vast amount of security protocols:

- Many protocols such as IPsec [4], TLS [5], SSH [6], JOSE [7], 802.1AE (MACsec) [8], 802.11ad (WiGig) [9], 802.11ac (Wi-Fi) [10], P1619.1 (data storage) [11], Fibre Channel [12], and SRTP [13, 14][1] only allow 128-bit tags.
- The exceptions are IPsec [15] that allows 64, 96, and 128 bit tags, CMS [16] that allows 96, 104, 112, 120, and 128 bit tags, NFC-SEC [17, 18] that only allows 96 bit tags, and QUIC [19] that only allows 96 bit tags.

[1] The Internet Drafts specifying the use of GCM in SRTP did originally allow also 64-bit and 96-bit tags, but this was removed after the publication of this paper on the Cryptology ePrint Archive and the discussion of this paper on the IETF AVTCORE mailing list.

GCM is also used in several cryptography APIs:

- W3C Web Cryptography API [20] and Oracle Java SE [21] allow 32, 64, 96, 104, 112, 120, and 128 bit tags. PKCS #11 [22] allows tags of any length between 0 and 128 bits, and for Microsoft Cryptography API [23] we could not find any information on allowed tag lengths.

The popularity is very well deserved, GCM has exceptional performance and proven security, it is online and fully parallelizable, and it is efficient in both hardware and software, especially on new processors with dedicated AES-GCM instructions. Weaknesses of the GCM decryption function were described by Ferguson [24], which showed that the forgery probability is not $2^{-t}$, and that feedback on successful forgeries allows an attacker to recover the authentication key $H$. As a note, the fact that the substitution probability decreases as message length increases was already known [25]. The results in this paper rely heavily on Ferguson's attack [24] and do not violate the provable security given in the original version of GCM [24]. The version standardized by NIST [1] makes normative changes to short tag lengths (32 and 64 bits) aimed to improve security, but NIST does not provide any estimated security levels given by these changes. The complexity of Ferguson's authentication key recovery method for the NIST approved short tags has not previously been analyzed.

Our results:

- In Sect. 3.1 we describe how to extend Fergusons's method for message forgery and authentication key recovery method [24] to use associated data, which is needed to apply the attack to IPv6 Jumbograms. We then describe an improvement that reduces the effective tag lengths for re-forgeries, derive a formula for the effective tag lengths, and use this improved method to calculate the probabilities for multiple message forgeries.
- In Sect. 3.2 we use these probabilities to calculate the complexity for authentication key recovery using Ferguson's method for the NIST approved short tag lengths (32 and 64 bits) showing that NIST seems to have chosen the parameters for 64 bit tags to get 80-bit security against Ferguson's attack.
- In Sect. 3.3 we suggest several novel improvements to Fergusons's method that significantly reduces the security levels for short tags, in one case the already low complexity is reduced from $2^{81.0}$ to $2^{70.0}$. We show that independently of the encryption key size, the security levels (i.e. the effective key lengths) are only 62–67 bits for 32-bit tags, and 70–75 bits for 64-bit tags. For these tag sizes, the security levels are far below, not only the current NIST requirement of 112-bit security, but also the old NIST requirement of 80-bit security. The results are applicable to both GCM and GMAC.
- In Sect 3.4 we show that feedback of successful or unsuccessful forgery attempts is almost always possible, contradicting the NIST assumptions for short tags. This illustrates that the key recovery attacks are practical and that the NIST assumption of no feedback is not valid for reasonable protocols and deployments. This is true especially for SRTP, which NIST claims meet the guidelines for use of short tags.

We strongly recommend NIST to revise SP 800-38D [1] so that the security levels of all allowed options are clearly stated, that short tags are removed, and that it is explained why any options offering less than 112-bit security against online attacks are acceptable.

We do however fully recommend GCM for usage with 128-bit tags, especially with AES-128. In fact we believe that with its excellent performance and proven security, it should be the first choice for everybody wanting an AEAD algorithm.

## 2 Preliminaries

### 2.1 Galois/Counter Mode (GCM)

Galois/Counter Mode (GCM) is an Authenticated Encryption with Associated Data (AEAD) mode of operation for block ciphers with a block size of 128 bits. It was designed by McGrew and Viega [26, 27] and is standardized in NIST SP 800-38D [1] and ISO/IEC 19772:2009 [28]. The analysis in this paper is based on [1]. GCM combines the well-known counter mode of encryption with the Galois mode of authentication, which is based on universal hashing. The Galois mode of authentication makes use of the function $\mathrm{GHASH}_H(A, C)$, which uses multiplications in $\mathrm{GF}(2^{128})$ that can easily be parallelized. The 128-bit authentication tag is defined as

$$\mathrm{Tag} = E_K(N) \oplus \mathrm{GHASH}_H(A, C) \ , \tag{1}$$

where $K$ is the encryption key, $N$ is the nonce, $H = E_K(0^{128})$ is the authentication key (the encryption of 128 zero bits), $A$ is the associated data (to be authenticated but not encrypted), and $C$ is the ciphertext. The output of the authenticated decryption function is either the plaintext $P$ or the special error code *FAIL*. Explicit weaknesses of the GCM functions have been discussed by Ferguson [24], Joux [29], Handschuh and Preneel [30], Saarinen [31], Procter and Cid [32], and Abdelraheem et al. [33]. An extensive evaluation of GCM was done by Rogaway [34].

Galois Message Authentication Code (GMAC) is an authentication-only variant of GCM. It can be seen as a special case of GCM where the ciphertext $C$ is the empty string. We refer to [1] for the full specification of GCM and GMAC.

### 2.2 Authentication Weaknesses in GCM

During the NIST standardization of AES-GCM, Fergusson [24] demonstrated through a concrete attack that due to the linear behavior of the GCM authentication function, the forgery probability is not $2^{-t}$, and that feedback on successful forgeries allows an attacker to recover the authentication key $H$.

Fergusson considers the case when there is no associated data and the attacker tries to change the ciphertext $C$ without changing the tag. Let $C_i$ be

block $i$ of $C$, where the blocks are numbered so that $C_1$ encodes the length of the ciphertext. The tag can now be written as

$$\text{Tag} = E_K(N) \oplus \sum_{i \geq 1} C_i \cdot H^i \ . \tag{2}$$

The attacker does not change the number of blocks in $C$ and only changes blocks in $C$ where $i$ is a power two. Let $C'$ be the modified ciphertext and define the error polynomial $E$ as

$$E = \sum_{i \geq 0} D_i \cdot H^{2^i} = \sum_{i \geq 0} (C_{2^i} - C'_{2^i}) \cdot H^{2^i} \ , \tag{3}$$

where $D_i = (C_{2^i} - C'_{2^i})$. Fergusson shows that the error polynomial $E$ is a linear function of $H$ and that the attacker can force a number of bits in $E$ to zero. If the length of $C$ is at least $2^l - 1$ blocks and not a multiple of 16, the attacker has $128l$ free variables and can in the first forgery force $e_0 = l$ bits of the error polynomial $E$ to zero. The effective tag length for the first forgery is therefore $t_0 = t - l$.

Fergusson then shows that feedback of successful forgery of a message with effective tag length $t_n$ allows recovery of $t_n$ additional bits of the authentication key $H$. The effective tag length for each succeeding forgery is therefore decreasing until the attacker has full knowledge of $H$ and can forge all subsequent tags with probability 1. As the attack is dominated by the complexity of finding the first forgery, full authentication key recovery requires approximately $2^{t_0} = 2^{-l} \cdot 2^t$ forgery attempts. As pointed out by McGrew and Viega in [35], Fergusson's attack does not break the security guarantees of GCM; it proves that the bounds in [27] are tight.

### 2.3 NIST Standardized Version of GCM

The NIST standard SP 800-38D [1] specifies that the 128-bit authentication tag may be truncated to 96, 104, 112, or 120 bits. For tag lengths of at least 96 bits, the maximum combined length of $A$ and $C$ is $L = 2^{57}$ blocks and the maximum number of invocations $q$ of the authenticated decryption function is unlimited. For certain applications the tag may be truncated to 32 or 64 bits, and for these tag lengths, $L$ and $q$ are more restricted. In Appendix B of SP 800-38D [1], NIST summarizes some particulars of the GCM authentication function that were pointed out by Ferguson [24]:

- For $t$-bit tags, the forgery probability is not the ideal $2^{-t}$ but instead $2^l \cdot 2^{-t}$ where $2^l$ is the length in blocks of the largest message ($A$ and $C$) processed by the authenticated encryption function.
- Each successful forgery enables the adversary to recover parts of the authentication key $H$ and increases the probability of subsequent forgeries.

NIST then draws the conclusion that the following additional requirement shall apply to short tags:

1. There should not be feedback of whether a forgery attempt is successful or unsuccessful.
2. The maximum combined length $L$ of $A$ and $C$ shall be heavily restricted.
3. The maximum number of invocations $q$ of the authenticated decryption function shall be restricted.

The details of requirement 2 and 3 are listed in Table 1. Unfortunately, NIST does not give any motivations for the specific choice of parameters, or for that matter the security levels they were assumed to give. In Sect. 3.4 we show that requirement 1 on feedback is not realistic and that feedback is almost always possible when security protocols like IPsec or SRTP are used. In Sect. 3.3 we show that with our improvements to Fergusson's attack, requirement 2 and 3 has smaller effect than expected.

**Table 1.** NIST requirements on the usage of GCM with short tags.

| $t$ | 32 | | | | | | 64 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $L$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^{11}$ | $2^{13}$ | $2^{15}$ | $2^{17}$ | $2^{19}$ | $2^{21}$ |
| $q$ | $2^{22}$ | $2^{20}$ | $2^{18}$ | $2^{15}$ | $2^{13}$ | $2^{11}$ | $2^{32}$ | $2^{29}$ | $2^{26}$ | $2^{23}$ | $2^{20}$ | $2^{17}$ |

## 3 Our Results

### 3.1 Use of Associated Data and Lowered Effective Tag Length

As mentioned above, Fergusson [24] demonstrated through a concrete attack that due to the linear behavior of the GCM authentication function, the forgery probability for $t$-bit tags is not $2^{-t}$. The tag and message lengths must therefore be chosen so that the forgery probability $L \cdot 2^{-t}$ is acceptable. We do not find this overly problematic and our view is that complexity is a better and more natural measure of forgery resistance. For an ideal MAC, the data complexity to perform a single forgery is $2^0 \cdot 2^t = 2^t$. For GCM, the data complexity to perform a single forgery is $2^l \cdot 2^{t-l} = 2^t$. The fact that each successful forgery enables the adversary to recover parts of the authentication key $H$ and increases the probability of subsequent forgeries is more problematic.

Reading [24], it is not trivial to understand or calculate the effective tag lengths for re-forgeries. In this section we extend Fergusson's method to use associated data in addition to ciphertext. This extension is needed to apply Ferguson's attack to IPv6 Jumbograms. We then suggest an improvement to Ferguson's method, derive a formula for the effective tag lengths, and apply this formula to the NIST approved tag and message lengths. These effective tag lengths are then used in Sect. 3.2 to evaluate the data complexity of Ferguson's method for authentication key recovery.

**Extension to use associated data.** The attacker tries to change the associated data $A$ and the ciphertext $C$ without changing the tag. The attacker does not change the number of blocks in $A$ and $C$. Let $A'$ be the modified associated data, let $C'$ be the modified ciphertext, and define $B$ and $B'$ as

$$\begin{aligned} B &= A \,||\, 0^{128-v} \,||\, C \,||\, 0^{128-u} \,||\, \mathrm{len}(A) \,||\, \mathrm{len}(C) \ , \\ B' &= A' \,||\, 0^{128-v} \,||\, C' \,||\, 0^{128-u} \,||\, \mathrm{len}(A) \,||\, \mathrm{len}(C) \ , \end{aligned} \tag{4}$$

where $v$ is the bit length of the final block of $A$ and $u$ is the bit length of the final block of $C$. Let $B_i$ be block $i$ of $B$, where we number the blocks in the same order as Ferguson, i.e. $B_1 = \mathrm{len}(A) \,||\, \mathrm{len}(C)$. We can now define the error polynomial $E$ as

$$E = \sum_{i \geq 0} D_i \cdot H^{2^i} = \sum_{i \geq 0} (B_{2^i} - B'_{2^i}) \cdot H^{2^i} \ , \tag{5}$$

where $D_i = (B_{2^i} - B'_{2^i})$.

**Effective Tag Length.** Let $t_n$ be the effective tag length after $n$ successful forgeries (with feedback). Following the procedures in [24] and assuming that:

- The byte length of $A$ or $C$ is not a multiple of 16. This implies that the attacker can modify the length encoding in $D_0$.
- The combined length of $A$ and $C$ is at least $2^l - 1$ blocks.

With these assumptions, the attacker has $128l$ free variables and can in the first forgery force $e_0 = l$ bits of the error polynomial $E$ to zero. The effective tag length is therefore $t_0 = t - l$. In subsequent forgeries the attacker knows more bits of the authentication key $H$ and can force even more bits of the error polynomial $E$ to zero. Feedback of successful forgery of a message with effective tag length $t_n$ allows recovery of $t_n$ additional bits of the authentication key $H$. After $n$ successful forgeries, the attacker knows $h_n$ bits of $H$ and can force $e_n$ bits of the error polynomial $E$ to zero where

$$h_n = \sum_{j=0}^{n-1} t_j \quad \text{and} \quad e_n = \left\lfloor \frac{128l}{128 - h_n} \right\rfloor \ . \tag{6}$$

Following [24], the effective tag length is $t_n = t - e_n$ until the attacker knows all 128 bits of $H$ ($h_n \geq 128$) or when the attacker can force more then $t$ bits of the error polynomial to zero ($e_n \geq t$), in which case the effective tag size is zero.

**Exhaustive search improvement.** We notice that when $128 - h_n < t - e_n$, the effective tag length can be reduced by doing exhaustive search on the $128 - h_n$ unknown bits of $H$ instead of doing exhaustive search on the $t - e_n$ bits of the tag that could not be forced to zero. With this improvement, the effective tag size is

$$t_n = \max\left(0,\, \min(t - e_n,\, 128 - h_n)\right) \ . \tag{7}$$

This improvement significantly reduces some of the effective tag lengths, but has negligible effect on the authentication key recovery complexities in the coming sections. The result of applying the improved formula (7) to the NIST approved tag and maximum message lengths, as well as the maximum message lengths of $2^{12}$ and $2^{28}$ blocks imposed by IPv4 and IPv6 is shown in Table 2.

While the values $t_0$ might look short, the complexity of performing a single forgery is still the expected $2^t$. If a tag length of $t = 128$ is used with an encryption key of length 128 bits, performing a single forgery is as hard as recovering the encryption key, hardly a weakness.

The effective tag lengths in Table 2 are calculated with the greedy algorithm used by Ferguson. Using the suggestions we propose in Sect. 3.3, it is possible to decrease the effective tag length of later forgeries by increasing the effective tag length of earlier forgeries.

**Table 2.** Effective tag lengths for the NIST approved tag and message lengths.

| $t$ | 32 | | | | | | 64 | | | | | | 96 | | | 104 | | | 112 | | | 120 | | | 128 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\log_2 L$ | 1 | 2 | 3 | 4 | 5 | 6 | 11 | 13 | 15 | 17 | 19 | 21 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 |
| $t_0$ | 31 | 30 | 29 | 28 | 27 | 26 | 53 | 51 | 49 | 47 | 45 | 43 | 84 | 68 | 39 | 92 | 76 | 47 | 100 | 84 | 55 | 108 | 92 | 63 | 116 | 100 | 71 |
| $t_1$ | 31 | 30 | 29 | 27 | 26 | 25 | 46 | 43 | 40 | 38 | 35 | 33 | 44 | 37 | 15 | 36 | 36 | 14 | 28 | 31 | 13 | 20 | 21 | 8 | 0 | 0 | 0 |
| $t_2$ | 31 | 29 | 27 | 25 | 24 | 23 | 16 | 16 | 15 | 14 | 14 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | 29 | 26 | 24 | 22 | 20 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_4$ | 6 | 13 | 12 | 13 | 12 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_5$ | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 3.2 Complexity of Ferguson's Authentication Key Recovery Method

The discussions [35, 36] after Ferguson's paper [24] focused mostly on multiple forgeries and authentication key recovery after nonce collisions in the encryption function, i.e. the forbidden attack later discussed by Joux [29]. We think the most important aspect of Ferguson's paper is the full recovery of the authentication key $H$ after successful forgeries to the decryption function. While we agree with McGrew and Viega that the expected complexity to perform multiple forgeries is unclear, the expected complexity against key recovery is very clear. The complexity of performing full key recovery is expected to be $2^k$ where $k$ is the stated security level. Unless stated otherwise, $k$ is expected to be equal to the key length. In e.g. HMAC-SHA-256 the complexity for key recovery is believed to be $2^{256}$, unless the authentication key is derived from a smaller key. In GCM, the authentication key is always 128 bits, which means that the security level against authentication key recovery is never more than 128 bits, even if block ciphers with larger key sizes like AES-192 or AES-256 are used. Other AEAD

schemes like CCM and OCB give a security level equal to the encryption key size. This shortcoming is not mentioned in [1, 27, 34].

An important detail mentioned in [29] but not in [1, 24] is that as the authentication tag depends on $E_K(N)$, authentication key recovery in GCM does not mean that the attacker can independently create new messages. If the length of $N$ is fixed, knowledge of the authentication key $H$ enables an attacker to modify a valid message by freely choosing $A$ and $C$, but not $N$. Assuming known-plaintext, an attacker can freely chose $A$ and $P$, where $P$ is the plaintext. Still, we would expect a security level of no less than the encryption key length against authentication key recovery attacks. In [33] Abdelraheem et al. show that if a GCM implementation supports variable nonce lengths and the attacker has knowledge of $H$, slide universal forgeries using twisted polynomials enable an attacker to choose $N$ as well.

**Complexity without query restrictions.** Assuming a maximum combined length of $L = 2^l$ blocks, the effective tag length is $t_0 = t - l$, and the data complexity (measured in blocks) of performing the first forgery is $2^l \cdot 2^{t-l} = 2^t$. As the complexity of Ferguson's authentication key recovery method is dominated by the complexity of the first forgery, this is also the data complexity $c$ for full authentication key recovery

$$c \approx 2^t \quad . \tag{8}$$

Hence, without restrictions on $q$ and irrespective of encryption key length, the security level of GCM against full authentication key recovery is only equal to the tag length $t$. This shortcoming is not mentioned in [1, 34].

**Complexity with query restrictions.** The complexity of Fergusson's key recovery method with restrictions $q$ and $L$ has not previously been analyzed. In this section we derive the complexities for the NIST approved tag and maximum message lengths. Let $p_n$ be the probability that an attacker succeeds with $n$ forgeries in $q$ attempts and let $l = \log_2 L$. We can now calculate the complexity $c$ of authentication key recovery with Ferguson's method as

$$c \approx q \cdot 2^l / p_n \quad , \tag{9}$$

where $n$ is the number of forgeries needed to recover the full authentication key. Limiting the maximum number of invocations $q$ of the decryption function so that $2^{t_0} \gg q \gg 2^{t_1}$ does not increase the complexity of authentication key recovery. The data complexity is $q \cdot 2^l$ and the probability that the attacker succeeds with one forgery in $q$ attempts is $p_1 \approx q \cdot 2^{-t_0}$, resulting in the same total complexity of $q \cdot 2^l / p_1 = 2^l / 2^{l-t} = 2^t$.

Restricting $q$ so that $2^{t_1} \gg q$ does however increase the complexity of Ferguson's method. Let $\phi_i = 2^{-t_i}$. The probability that the first successful forgery will occur on forgery attempt $f$ is approximately $\phi_0(1 - \phi_0)^{f-1}$ and the probability of a second forgery is approximately $\phi_1(q - f)$. The probability $p_2$ that an

attacker succeeds with two forgeries in $q$ attempts is therefore:

$$p_2 \approx \sum_{f=1}^{q} \phi_0(1-\phi_0)^{f-1} \cdot \phi_1(q-f) = \frac{\phi_0\phi_1}{2}q^2 + \mathcal{O}\left(\frac{\phi_0^2\phi_1}{6}q^3\right) \quad . \tag{10}$$

We used SageMath to calculate the Taylor series and then collected the leading terms for the domain $\phi_0, \phi_1 \ll q^{-1}$. McGrew and Viega prove a formula similar to (10) in [36], but do not calculate further values. With the above approximation for $p_2$ we can approximate $p_3$ using that the probability of a second and third forgery is approximately $\phi_1\phi_2(q-f)^2/2$, and with $p_n$ we can approximate $p_{n+1}$, etc.[2]

$$\begin{aligned}
p_3 &\approx \sum_{f=1}^{q} \phi_0(1-\phi_0)^{f-1} \cdot \frac{\phi_1\phi_2}{2}(q-f)^2 = \frac{\phi_0\phi_1\phi_2}{6}q^3 + \mathcal{O}\left(\frac{\phi_0^2\phi_1\phi_2}{24}q^4\right) \quad , \\
p_4 &\approx \sum_{f=1}^{q} \phi_0(1-\phi_0)^{f-1} \cdot \frac{\phi_1\phi_2\phi_3}{6}(q-f)^3 = \frac{q^4}{4!}\prod_{j=0}^{3}\phi_j + \mathcal{O}\left(\frac{\phi_0 q^5}{5!}\prod_{j=0}^{3}\phi_j\right) \quad , \\
p_5 &\approx \sum_{f=1}^{q} \phi_0(1-\phi_0)^{f-1} \cdot \frac{\phi_1\phi_2\phi_3\phi_4}{24}(q-f)^4 = \frac{q^5}{5!}\prod_{j=0}^{4}\phi_j + \mathcal{O}\left(\frac{\phi_0 q^6}{6!}\prod_{j=0}^{4}\phi_j\right) \quad .
\end{aligned} \tag{11}$$

**Complexity for the NIST tag and message lengths.** With the approximations for $p_1, p_2, p_3, p_4, p_5$ we can calculate the complexity of authentication key recovery with Ferguson's method. Table 3 shows the complexities achieved by applying (9), (10), and (11) to the NIST approved tag and maximum message lengths. The grey coloring shows the $t_n$ that was used in the calculation In a few cases the domain assumption does not hold as $2^{t_n} \approx q$. In these cases we have chosen $n$ to overestimate rather than underestimate the complexity. Note that the complexities for authentication key recovery are independent of the encryption key length.

Our analysis show that with Ferguson's method the security levels for 32-bit tags are below the old NIST requirement of 80-bit security (that was in place in 2007 when [1] was published), while 64-bit tags are just on the border. In fact, NIST seems to have chosen the parameters for 64 bit tags to get 80-bit security against Ferguson's attack.

Only 112, 120, and 128 bit tags fulfill the current NIST requirement of 112-bit security. Unfortunately, NIST does not give any motivations for the exact restrictions they put on 32 and 64 bit tags, or for that matter the security levels they were assumed to give.

[2] The calculations below lead us to the hypothesis that $p_n \approx \frac{q^n}{n!}\prod_{j=0}^{n-1}\phi_j + \mathcal{O}\left(\frac{\phi_0 q^{n+1}}{(n+1)!}\prod_{j=0}^{n-1}\phi_j\right)$. This is however something that we do not use and that we do not prove, but by dividing $q$ into $n$ intervals, it is easy to prove that $p_n \geq \frac{q^n}{n!}\prod_{j=0}^{n-1}\phi_j$.

**Table 3.** Data complexity with Ferguson's method for full authentication key recovery.

| $t$ | 32 | | | | | | 64 | | | | | | 96 | | | 104 | | | 112 | | | 120 | | | 128 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\log_2 L$ | 1 | 2 | 3 | 4 | 5 | 6 | 11 | 13 | 15 | 17 | 19 | 21 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 |
| $\log_2 q$ | 22 | 20 | 18 | 15 | 13 | 11 | 32 | 29 | 26 | 23 | 20 | 17 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| $t_0$ | 31 | 30 | 29 | 28 | 27 | 26 | 53 | 51 | 49 | 47 | 45 | 43 | 84 | 68 | 39 | 92 | 76 | 47 | 100 | 84 | 55 | 108 | 92 | 63 | 116 | 100 | 71 |
| $t_1$ | 31 | 30 | 29 | 27 | 26 | 25 | 46 | 43 | 40 | 38 | 35 | 33 | 44 | 37 | 15 | 36 | 36 | 14 | 28 | 31 | 13 | 20 | 21 | 8 | 0 | 0 | 0 |
| $t_2$ | 31 | 29 | 27 | 25 | 24 | 23 | 16 | 16 | 15 | 14 | 14 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | 29 | 26 | 24 | 22 | 20 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_4$ | 6 | 13 | 12 | 13 | 12 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\log_2 c$ | 62 | 62 | 63 | 66 | 69 | 72 | 79 | 79 | 79 | 80 | 80 | 81 | 96 | 96 | 96 | 104 | 104 | 104 | 112 | 112 | 112 | 120 | 120 | 120 | 128 | 128 | 128 |

### 3.3 Our Improved Method for Authentication Key Recovery

In this section we propose several novel improvements to Ferguson's method for authentication key recovery. These improvements significantly reduce the security levels for short tags.

- The attacker may choose to modify a message with a message length that is smaller than the maximum message length $L$.
- After each successful forgery, the attacker may choose to modify a different message.
- The attacker may choose to modify messages with different lengths $2^{l_0}$, $2^{l_1}$, $2^{l_2}$, ...

Let the length of the first message be $2^{l_0}$ and let $l = \max(l_1, l_2, \dots)$. The probability that the attacker does not achieve a single successful forgery in $q$ attempts is $(1-\phi_0)^q$ in which case the attacker sends $q2^{l_0}$ blocks of data. The probability that the first successful forgery will occur on forgery attempt $f$ is approximately $\phi_0(1-\phi_0)^{f-1}$ in which case the attacker sends at most $f2^{l_0} + (q-f)2^l$ blocks of data. The average number of blocks $w$ sent by the attacker is therefore bounded by:

$$\begin{aligned} w &\leq (1-\phi_0)^q \cdot q2^{l_0} + \sum_{f=1}^{q} \phi_0(1-\phi_0)^{f-1} \cdot \left(q2^l - f(2^l - 2^{l_0})\right) \\ &= q2^{l_0} + \frac{1}{2}\phi_0 q^2(2^l - 2^{l_0}) + \mathcal{O}(\phi_0^2 q^3 2^{l_0}) \ . \end{aligned} \tag{12}$$

We used SageMath to calculate the Taylor series and then collected the leading terms for the domain $\phi_0 \ll q^{-1}$. Using this improved method, the data complexity $c$ of authentication key recovery is

$$c \approx q \cdot 2^{l_0}/p_n \ . \tag{13}$$

**64-bit tags.** Let $l_0 = 0$ and $l = \log_2 L$. For 64-bit tags, the effective tag lengths are $t_0 = 64$, $t_1 = 64 - 2l$, and the complexity is

$$c_{64} \approx q \cdot 2^{l_0}/p_2 = 2^{t_0+t_1+1}/q = 2^{129}/qL^2 \ , \qquad qL^2 \ll 2^{64} \ . \tag{14}$$

By applying (14) to the column ($L = 2^{21}$, $q = 2^{17}$), the already low complexity is reduced from $2^{81.0}$ to $2^{70.0}$. It seems infeasible to increase the security level to 112 bits, as this would either restrict the message length too much or make deployments vulnerable to denial-of-service attacks.

Table 4 shows the complexities achieved by applying our improved method (13) with $l_0 = 0$ and $l = \log_2 L$ to the NIST approved tag and maximum message lengths. This significantly reduces the data complexities of authentication key recovery for short tags. With our improved method, the security levels are 62–67 bits for 32-bit tags and 70–75 bits for 64-bit tags; this is below the old NIST requirement of 80-bit security and far below the current NIST requirement of 112-bit security.

**Table 4.** Data complexity with our improved method for authentication key recovery.

| $t$ | 32 | | | | | | 64 | | | | | | 96 | | | 104 | | | 112 | | | 120 | | | 128 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\log_2 L$ | 1 | 2 | 3 | 4 | 5 | 6 | 11 | 13 | 15 | 17 | 19 | 21 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 | 12 | 28 | 57 |
| $\log_2 q$ | 22 | 20 | 18 | 15 | 13 | 11 | 32 | 29 | 26 | 23 | 20 | 17 | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| $t_0$ | 32 | 32 | 32 | 32 | 32 | 32 | 64 | 64 | 64 | 64 | 64 | 64 | 96 | 96 | 96 | 104 | 104 | 104 | 112 | 112 | 112 | 120 | 120 | 120 | 128 | 128 | 128 |
| $t_1$ | 31 | 30 | 28 | 27 | 26 | 24 | 42 | 38 | 34 | 30 | 26 | 22 | 32 | 0 | 0 | 24 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_2$ | 31 | 29 | 27 | 25 | 23 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_3$ | 29 | 26 | 23 | 21 | 19 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $t_4$ | 5 | 9 | 11 | 10 | 10 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\log_2 c$ | 62 | 62 | 61 | 65 | 66 | 67 | 75 | 74 | 73 | 72 | 71 | 70 | 96 | 96 | 96 | 104 | 104 | 104 | 112 | 112 | 112 | 120 | 120 | 120 | 128 | 128 | 128 |
| $\Delta \log_2 c$ | 0 | 0 | -2 | -1 | -3 | -5 | -4 | -5 | -6 | -8 | -9 | -11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 3.4 Analysis of the use of GCM in Security Protocols

We show that neither IPsec nor SRTP fulfills the NIST requirements for short tags. The specification of the use of GCM with 64 bit tags in IPsec [15] was published shortly after Fergusson's paper [24] and does not refer to the NIST specification [1]. The RFC [13] and Internet Draft [14] specifying the use of GCM in SRTP do no longer allow the use of truncated tags, but the NIST specification mentions SRTP as an example of a protocol fulfilling the guidelines for short tags. Two of these guidelines are:

- There should not be feedback of whether a forgery attempt is successful or unsuccessful.
- The AAD within packets should be limited to the necessary header information.

**Analysis of GCM usage in IPsec ESP.** RFC 4106 [15] specifies the use of GCM with 64, 96, and 128 bit tags. The specification does not discuss the problems with short tags and does not require implementations to restrict the

maximum message length $L$ or the maximum number of invocations $q$ of the authenticated decryption function. While ESP limits the AAD to necessary header information and silently discards datagrams that fail the integrity check, ESP does not silently discard datagrams that passes the integrity check and information leakage regarding the integrity of individual packets is therefore possible in many deployments.

- If any request-response protocol is sent over an IPsec protected path, an attacker can attempt forgery by modifying a datagram containing a request (e.g. HTTP GET). If integrity fails, the IPsec implementation will silently discard the datagram. If the datagram passes the integrity check, a response (e.g. 200 OK) will be sent. The datagram containing the response will also be encrypted, but assuming small amounts of other traffic (the attacker may e.g. block certain traffic) the attacker can see that a response was sent and conclude that the forgery was successful.
- If tunnel mode is used, the attacker may modify the inner destination IP address so that the packet in case of a successful forgery is routed to the adversary himself/herself.

If multicast is used [37], the attacker may attempt forgery towards several instances of the GCM decryption function in parallel, and the maximum number of invocations $q$ of the decryption function would need to be calculated over all instances of the decryption function. Theoretically this could be done with synchronization, but in practice the only solution would be to restrict the number of invocations of each instance to $q/r$ where $r$ is the total number of receivers. This makes $q/r$ impractically small and makes the system vulnerable to denial-of-service attacks.

IPsec ESP with GCM and 64-bit tags offers 64 bits of security against online authentication key recovery and IPsec ESP with GCM and 96-bit tags offers 96 bits of security. A probable attack could be detected by an intrusion detection system by identifying a large number of messages only differencing in blocks $B_i$ where $i$ is a power two according to (4).

**Analysis of GCM usage in SRTP.** The Real-time Transport Protocol (RTP) [38] is a network protocol for transmitting real-time data, such as audio, video, and text. RTP is used in conjunction with the RTP Control Protocol (RTCP) to specify quality of service feedback and synchronization between media streams. The Secure Real-time Transport Protocol (SRTP) [39] provides encryption, message authenticity, and replay protection to RTP and RTCP. While RTP and SRTP are standardized in RFC 3550 [38] and RFC 3711 [39], there are numerous extensions to both protocols. In Appendix C of [1], NIST makes the statement:

> *"An example of a protocol that meets these guidelines is Secure Real-time Transport Protocol carrying Voice over Internet Protocol, running over User Datagram Protocol".*

This is not a correct statement and SRTP does in fact violate both of the guidelines mentioned before.

- The AAD is not at all limited. In RTP, the associated data consists of the RTP header, which is not limited as e.g. the header in the TLS record layer. The RTP header is extensible with proprietary header extensions carrying any type of information. In RTCP, the scope of the AAD depends on the encryption flag E. If the encryption flag is '1', the AAD data is limited to necessary header information, but if the encryption flag is '0', the AAD consists of the *entire* RTCP packet.
- RTCP receiver reports (RR) provide a wealth of information that can be used to determine the integrity of individual forged RTP packages, e.g. SSRC of the source, cumulative number of packets lost, extended highest sequence number received, last sender report (SR) timestamp, and delay since last SR. The RTCP extension for port mapping [40] is even worse as it echoes back the 64-bit nonce received in the request.
- RTP Rapid Synchronisation [41] is used; a forged Rapid Resynchronisation Request results in a RTP header extension with sync information sent from the sender.
- If the RTP header extension Client-to-Mixer Audio Level Indication [42] is used, a forged RTP packet with a high audio level will result in the Multipoint Control Unit (MCU) forwarding the SSRC. As the SSRC is not encrypted, this is easily detected by the attacker.

Even if encryption of RTCP is mandated and specific RTP header extensions and RTCP packets types are forbidden, an attacker may still in many cases determine whether a forgery was successful by looking at the length of packets. Either by looking at the length of RTCP packets from the sender or by looking at the length of RTP packets forwarded by an MCU.

A further problem with SRTP and GCM is that SRTP is very often used in one-to-many scenarios. The maximum number of invocations of each instance of the authenticated decryption function would have to be restricted to $q/r$, where $q$ is the maximum total number of invocations of the authenticated decryption function, and $r$ is the total number of receivers, including any late joiners.

All in all, SRTP does absolutely not meet the NIST guidelines for usage of GCM with short tags.

**Summary.** While many protocols silently discard packets with failed integrity check, very few are totally silent when the integrity check is valid. Even if the security protocol itself does not provide feedback, the higher level messages protected by the security protocol likely do. We believe that feedback of successful or unsuccessful forgery attempt is almost always possible. The NIST guideline is therefore unrealistic, and the authentication key recovery attacks practically possible. Analyzing the possibility of information feedback from successful forgeries is not trivial and the NIST statement regarding SRTP is obviously incorrect. We strongly recommend NIST to remove short tags from SP 800-38D [1].

## 4 Conclusions

The security levels of GCM and GMAC against authentication key recovery are for many tag sizes far below, not only the current NIST requirement of 112-bit security, but also the old NIST requirement of 80-bit security. With our improved authentication key recovery method, the security levels are 62–67 bits for the NIST approved usage of 32-bit tags and 70–75 bits for the NIST approved usage of 64-bit tags. For larger tags the security levels are as previously known $t$ bits for $t$-bit tags where $t = 96$, 104, 112, 120, or 128. It seems infeasible to increase the low security levels to 112 bits, as this would either restrict the message length too much or make deployments vulnerable to denial-of-service attacks.

We note that as the authentication key is always 128 bits, the security level against authentication key recovery is never more than 128 bits, even if block ciphers with larger key sizes like AES-192 or AES-256 are used. Other AEAD schemes like CCM and OCB give a security level equal to the encryption key size.

One might argue that it is acceptable to allow a lower security level against authentication key recovery than encryption key recovery, especially if authentication key recovery requires online access to the hopefully short-lived GCM instances. With this arguing, 96-bit tags could be acceptable, even if they only offer 96 bits of security against online authentication key recovery. We do not take a stance on this, but note that the current NIST requirements in NIST SP 800-57 Part 3 [43] states that the authentication key strength shall be equal or greater than 112 bits and that less than 112 bits of security shall not be used.

NIST states that implementations should not provide feedback on the integrity of individual packets and then nevertheless heavily restricts the number of invocations of the decryption function. We have illustrated that feedback on the integrity of individual packets is almost always possible. The NIST guideline is therefore unrealistic, and the authentication key recovery attacks practically possible. Analyzing the possibility of information feedback of successful forgeries is not trivial and the NIST statement regarding SRTP is obviously incorrect. We therefore strongly recommend NIST to remove short tags from SP 800-38D [1].

Furthermore, we recommend that such analysis is never left to the user, and we strongly recommend against standardizing any cryptographic algorithms that relies on the assumption of no information feedback from successful forgeries.

We strongly recommend NIST to make a revise SP 800-38D [1] so that the security levels of all allowed options are clearly stated, that short tags are removed, and that it is explained why any options offering less than 112-bit security against online attacks are acceptable. We do however fully recommend GCM for usage with 128-bit tags, especially with AES-128. In fact we believe that with its excellent performance and proven security, it should be the first choice for everybody wanting an AEAD algorithm. We note that the design choices causing the security problems with truncated tags are also responsible for the excellent performance of GCM.

## Acknowledgements

The authors would like to thank Ben Smeets and Mats Näslund for their advise and helpful comments.

## References

1. NIST SP 800-38D.: Recommendations for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. November 2007. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
2. NSA.: Suite B Cryptography. https://www.nsa.gov/ia/programs/suiteb_cryptography/
3. CAESAR.: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html
4. IETF RFC 4543.: The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH'. May 2006. https://tools.ietf.org/html/rfc4543
5. IETF RFC 5288.: AES Galois Counter Mode (GCM) Cipher Suites for TLS. August 2008. https://tools.ietf.org/html/rfc5288
6. IETF RFC 5647.: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. August 2009. https://tools.ietf.org/html/rfc5647
7. IETF RFC 7518.: JSON Web Algorithms (JWA). May 2015. https://tools.ietf.org/html/rfc7518
8. IEEE 802.1AE-2006.: Media Access Control (MAC) Security. August 2006. http://standards.ieee.org/getieee802/download/802.1AE-2006.pdf
9. IEEE 802.11ad-2012.: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band. October 2012. http://standards.ieee.org/getieee802/download/802.11ad-2012.pdf
10. IEEE 802.11ac-2013.: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. December 2013. http://standards.ieee.org/getieee802/download/802.11ac-2013.pdf
11. IEEE 1619.1-2007.: IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. May 2008.
12. ANSI INCITS 496-2012.: Information technology - Fibre Channel Security Protocol 2 (FC-SP-2).
13. IETF RFC 7714.: AES-GCM Authenticated Encryption in Secure RTP (SRTP). December 2015. https://tools.ietf.org/html/rfc7714
14. Kim, Lee, Kim, Park, Kwon.: The ARIA Algorithm and Its Use with the Secure Real-time Transport Protocol (SRTP). (IETF work in progress). November 2015. https://tools.ietf.org/html/draft-ietf-avtcore-aria-srtp-09
15. IETF RFC 4106.: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). June 2005. https://tools.ietf.org/html/rfc4106
16. IETF RFC 5084.: Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS). November 2007. https://tools.ietf.org/html/rfc5084
17. ECMA-409.: NFC-SEC-02: NFC-SEC Cryptography Standard using ECDH-256 and AES-GCM. December 2014. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-409.pdf

18. ECMA-411.: NFC-SEC-04: NFC-SEC Entity Authentication and Key Agreement using Symmetric Cryptography. December 2014. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-411.pdf
19. Langley, Chang.: QUIC Crypto. July 2015. https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/edit
20. W3C.: Web Cryptography API. December 2014. http://www.w3.org/TR/WebCryptoAPI/
21. Oracle.: Java Platform, Standard Edition 8 API Specification. https://docs.oracle.com/javase/8/docs/api/index.html
22. OASIS.: PKCS #11 Cryptographic Token Interface Current Mechanisms Specification Version 2.40. September 2014. http://docs.oasis-open.org/pkcs11/pkcs11-curr/v2.40/cs01/pkcs11-curr-v2.40-cs01.pdf
23. Microsoft.: Cryptography API: Next Generation. https://msdn.microsoft.com/en-us/library/windows/desktop/aa376210
24. Ferguson.: Authentication weaknesses in GCM. May 2005. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf
25. Kabatianskii, Smeets, Johansson.: On the Cardinality of Systematic Authentication Codes Via Error-Correcting Codes. IEEE Transactions on Information Theory, Vol. 42, No 2. March 1996
26. McGrew, Viega.: The Galois/Counter Mode of Operation (GCM). May 2005. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf
27. McGrew, Viega.: The Security and Performance of the Galois/Counter Mode of Operation. October 2004. http://eprint.iacr.org/2004/193.pdf
28. ISO/IEC 9772:2009.: Information technology – Security techniques – Authenticated encryption. July 2008. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=46345
29. Joux.: Authentication Failures in NIST version of GCM. 2006. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf
30. Handschuh, Preneel.: Key-recovery attacks on universal hash function based MAC algorithms. CRYPTO 2008. http://www.cosic.esat.kuleuven.be/publications/article-1150.pdf
31. Saarinen.: GCM, GHASH and Weak Keys. 2011. http://www.iacr.org/archive/fse2012/75490220/75490220.pdf
32. Procter, Cid.: On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes. FSE 2013. https://eprint.iacr.org/2013/144.pdf
33. Abdelraheem, Beelen, Bogdanov, Tischhauser.: Twisted Polynomials and Forgery Attacks on GCM. EUROCRYPT 2015. https://eprint.iacr.org/2015/1224.pdf
34. CRYPTREC TR No. 2012.: Evaluation of Some Blockcipher Modes of Operation. February 2011. http://www.cryptrec.go.jp/estimation/techrep_id2012_2.pdf
35. McGrew, Viega.: GCM Update. May 2005. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/gcm-update.pdf
36. McGrew, Fluhrer.: Multiple forgery attacks against Message Authentication Codes. May 2005. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/multi-forge-01.pdf
37. IETF RFC 5374.: Multicast Extensions to the Security Architecture for the Internet Protocol. November 2008. https://tools.ietf.org/html/rfc5374

38. IETF RFC 3550.: RTP: A Transport Protocol for Real-Time Applications. July 2003. https://tools.ietf.org/html/rfc3550
39. IETF RFC 3711.: The Secure Real-time Transport Protocol (SRTP). March 2004 https://tools.ietf.org/html/rfc3711
40. IETF RFC 6284.: Port Mapping between Unicast and Multicast RTP Sessions. June 2011. https://tools.ietf.org/html/rfc6284
41. IETF RFC 6051.: Rapid Synchronisation of RTP Flows. November 2010. https://tools.ietf.org/html/rfc6051
42. IETF RFC 6464.: A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication. December 2011. https://tools.ietf.org/html/rfc6464
43. NIST SP 800-57 Part 3-Rev.1.: Recommendation for Key Management: Part 3 - Application-Specific Key Management Guidance. January 2015. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf