Generic Security of GCM-SST

Akiko Inoue[1], Ashwin Jha[2], Bart Mennink[3], and Kazuhiko Minematsu[1,4]

[1] NEC Corporation, Kawasaki, Japan, `a_inoue@nec.com`, `k-minematsu@nec.com`
[2] Ruhr-Universität Bochum, Bochum, Germany, `letterstoashwin@gmail.com`
[3] Radboud University, Nijmegen, The Netherlands, `b.mennink@cs.ru.nl`
[4] Yokohama National University, Yokohama, Japan

Abstract. Authenticated encryption schemes guarantee that parties who share a secret key can communicate confidentially and authentically. One of the most popular and widely used authenticated encryption schemes is GCM by McGrew and Viega (INDOCRYPT 2004). However, despite its simplicity and efficiency, GCM also comes with its deficiencies, most notably devastating insecurity against nonce-misuse and imperfect security for short tags. Very recently, Campagna, Maximov, and Mattsson presented GCM-SST (IETF Internet draft 2024), a variant of GCM that uses a slightly more involved universal hash function composition, and claimed that this construction achieves stronger security in case of tag truncation. GCM-SST already received various interest from industries (e.g., Amazon and Ericsson) and international organizations (e.g., IETF and 3GPP) but it has not received any generic security analysis to date. In this work, we fill this gap and perform a detailed security analysis of GCM-SST. In particular, we prove that GCM-SST achieves security in the nonce-misuse resilience model of Ashur et al. (CRYPTO 2017), roughly guaranteeing that even if nonces are reused, evaluations of GCM-SST for new nonces are secure. Our security bound also verified the designers' (informal) claim on tag truncation. Additionally, we investigate and describe possibilities to optimize the hashing in GCM-SST further, and we describe a universal forgery attack in a complexity of around $2^{33.6}$, improving over an earlier attack of 2^{40} complexity of Lindell, when the tag is 32 bits.

Keywords: Authenticated encryption, GCM, GCM-SST, universal forgery, nonce misuse.

1 Introduction

Authenticated encryption (AE) is a vital concept for the security of today's digital environment. It allows for two parties sharing a symmetric cryptographic key to achieve confidentiality as well as authenticity for the data they communicate. In a bit more detail, an AE scheme gets as input a key, optionally associated data, and plaintext, and it outputs a ciphertext and a tag. The corresponding decryption scheme gets as input a key, optionally associated data, ciphertext, and tag, and it outputs the message corresponding to the ciphertext if (and only if) the tag verifies correctly. Whereas initial designs (such as IAPM in 2000 [37] and OCB1 in 2001 [51]) and the well-known formalism of Bellare and Namprempre [7,44] mostly saw AE as a version of encryption that *additionally* obtained some level of authenticity, the current view is more holistic in the sense that AE is a well-matured and maybe the most important building block of symmetric cryptography.

Many AE schemes have been introduced since. Undoubtedly the most notable one of them is the Galois/Counter Mode (GCM) that McGrew and Viega introduced in 2004 [41]. In a nutshell, GCM is a block cipher based AE mode that operates by encrypting the plaintext using nonce-based counter mode, and sequentially authenticating associated data and ciphertext using a Wegman-Carter-Shoup [59,15,55] authenticator. Refer to Figure 1 for a visual representation. McGrew and Viega [41] performed an initial generic security analysis, that got later fixed by Iwata et al. [35] and improved by Niwa et al. [46]. In detail, GCM achieves security up to $2^{n/2}$ data blocks, under the assumption that the underlying n-bit block cipher is PRP-secure. Presumably because of this security result, as well as the simplicity and efficiency of the construction, GCM has become one of the most widely used AE schemes, being standardized in NIST SP 800 38D [23] and ISO/IEC 19772 [33] as well as being used in TLS [53], IPSec [58], SSH [31], Wifi (WPA3) [60], NSA Suite B [47], and IEEE 802.1 [30].

However, over the years, the range of applications of AE has significantly broadened, and so have the requirements imposed on these designs. For example, use cases may range from lightweight applications to regular platforms, whereas threat models may range from regular nonce-based security to security against adversaries that may freely manipulate the nonce. These evolutions have lead to two notable competitions, the CAESAR competition [16] running from 2014 to 2019 and the NIST Lightweight Cryptography competition [45] running from 2019 to 2023, that have both boosted the detailed research on security models of AE [44,3,5] and have lead to a large amount of novel AE schemes [51,50,38,4,43,12,2,34,6,29,20].

Despite these efforts, GCM is still omnipresent in our digital environment, even though it does not achieve security in a wide arrange of models. To give a few examples:

- GCM does not achieve security under nonce reuse as observed by Joux [36]. To the contrary, an adversary misusing the nonce can obtain the subkey used for authentication. This is a practical problem, as guaranteeing a unique nonce is hard in some settings [14];
- The algorithm of GCM allows arbitrary nonce length [41,46], however 96-bit nonce is often recommended for its efficiency and simplicity. The use of 96-bit nonce could be problematic in multi-user attacks if an adversary has access to many endpoints. This has lead to an ad-hoc solution for AES-GCM in TLS 1.3 [49,53] to introduce extra key material and use it to blind the nonce (an approach that was later proven to be sound by Bellare and Tackmann [9] and that was generalized in the context of duplex-based [12,42] AE by Dobraunig and Mennink [21]);

– GCM does not achieve security under release of unverified plaintext [3], where an adversary may obtain plaintext coming from the decryption function before the tag is verified. To salvage this, Ashur et al. [5] introduced a variant GCM-RUP that achieves RUP-security with minor alteration;
– Finally, there is the issue with short tags, as already brought up by Ferguson in 2005 [27]. In a nutshell, shorter tags are easier to forge, and successful forgeries make it easier for the adversary to recover the subkey used for authentication. Nyberg et al. [48] highlighted the issue that forgeries are possible in a complexity of around $2^t/\ell$, where t is the tag size and ℓ the message block length (see also Theorem 2). This problem is particularly problematic when short tags become relevant, e.g., when bandwidth is critical, such as in low-power wireless communication and memory encryption.

In a very recent attempt, Campagna, Mattsson, and Maximov [17] tried to address the issue of short tags in GCM. In detail, they presented GCM-SST, that differs from GCM in the sense that in authentication, the universal hash function is replaced by a cascade of two hash functions under different subkeys, following the idea originally suggested by Nyberg et al. [48]. Overall, this is a minor penalty in efficiency for a significant gain, and because of this, the scheme has quickly gained attention. Currently, it is strongly supported by certain industries (e.g., Amazon and Ericsson [18]), it is in the process of being standardized by IETF [17], and ETSI SAGE has finalized the specifications of GCM-SST for 5G communication [1,19,24,26,40]. Google also showed interest in GCM-SST for use in their video conference applications.[5]

However, at the same time, there is no clarity about the actual security of the proposal of Campagna et al.: to the contrary, GCM-SST was presented without security proofs, and Lindell [39] even described a universal forgery attack with complexity around 2^{40} decryption queries when the scheme uses 32-bit tags (which is a typical "short tag" that the designers suggest), that casts a shadow over the scheme's security [25].

1.1 Our Contributions

In this work, we perform an in-depth generic security analysis of GCM-SST.

First, in Section 4, we prove that GCM-SST is a provably secure AE mode. Clearly, it will not achieve full nonce-misuse resistance, mostly as it is a one-pass AE scheme, but instead we demonstrate that it is secure in the nonce-misuse resilience model of Ashur et al. [5]: in a nutshell, this model guarantees security for evaluations under *fresh* nonces even if earlier nonces may have been reused in the past. This model is particular relevant, e.g., in use cases where the nonce contains receiver addresses or time stamps, and various modes [5,11,10,32] have been presented and analyzed in this model. In addition, by definition, security in this model immediately implies security in the nonce-respecting setting up

[5] https://datatracker.ietf.org/meeting/119/materials/minutes-119-avtcore-202403182330-00

to at least the same bound. Thus, we adopt this nonce-misuse resilience model, and in this model we prove that GCM-SST achieves security up to a complexity of around $\min\{2^{n/2}, 2^t\}$ data blocks, where n is the block size and t the tag size (see also Theorems 3 and 4).

Second, in Section 5, we put the results in perspective, from three angles. We start in Section 5.1 with drawing a comparison between the bounds of GCM and GCM-SST, with a focus on authenticity and in particular the security guaranteed under shorter tag sizes, and demonstrate that GCM-SST has significantly reduced the security degradation for short tags as the designers claimed without a formal proof. Note that this observation is already suggested by the security bounds, where the dominating authentication portion improves from $2^t/\ell$ to 2^t. Then, we discuss in Section 5.2 the possibility to minimize the masking in GCM-SST, and observe that one masking block could be avoided if one would opt for a slightly more expensive padding function. This variant generally improves efficiency. Finally, in Section 5.3 we describe a universal forgery attack with $3 \cdot 2^t + 4(n-t) + 1$ decryption queries for any t, that is based on the idea that for a smart choice of forgery attempts, two successful attempts simply give t bits of the multiplication of the two hashing subkeys, and the remaining $(n-t)$ bits can be obtained bit-by-bit by sliding through the multiplication in the universal hash function. When $t = 32$-bit tag, our attack complexity is about $2^{33.6}$, improving the aforementioned Lindell's attack with 2^{40} queries.

1.2 Outline

We present preliminary material in Section 2. The GCM and GCM-SST modes are specified in our terminology in Section 3. In Section 4 we present our security analysis of GCM-SST, with nonce-misuse resilience privacy in Section 4.2 and authenticity in Section 4.4. We conclude the work in Section 5, where we discuss how GCM compares with GCM-SST (Section 5.1), where we investigate the possibilities to minimize the amount of masking material in GCM-SST (Section 5.2), and where we describe a universal forgery attack against GCM-SST improving the attack by Lindell [39] (Section 5.3).

2 Preliminaries

Let $\mathbb{N}$ denote the set of all non-negative integers, and $\mathbb{N}^+ \coloneqq \mathbb{N} \setminus \{0\}$. Let $\{0,1\}^*$ denote the set of all binary strings of arbitrary length, including the empty string ε. For any $n \in \mathbb{N}$, $\{0,1\}^n$ denotes the set of all n-bit strings, and $[n]$ denotes the set $\{1,\ldots,n\}$. For any $x, y \in \{0,1\}^*$, $|x|$ denotes the bit-length[6] (or simply length) of x, and $x \,\|\, y$ denotes the concatenation of x and y. Let $|x|_n \coloneqq \max\{1, \lceil |x|/n \rceil\}$. For any $x \in \{0,1\}^*$ and any $n \in \mathbb{N}^+$, $\lceil\!\lceil x \rceil\!\rceil_n$ (resp., $\lfloor\!\lfloor x \rfloor\!\rfloor_n$) denotes the leftmost (resp., rightmost) $\min\{n, |x|\}$ bits of x. For $n \in \mathbb{N}^+$ and $x \in \{0,1\}^n$, let $x \ll 1$ be a one-bit left-shifted value of x; thus, $x \ll 1 =$

[6] The number of bits in the given string. For an empty string, it is zero.

$\llbracket x \rrbracket_{n-1} \| 0$. For any $n, x \in \mathbb{N}^+$, $\langle x \rangle_n$ denotes the canonical unsigned n-bit binary representation of x. For any $x \in \{0,1\}^*$,

$$\mathsf{pad}_n(x) := x \,\|\, 0^{(n - |x| \bmod n)}.$$

Note that $\mathsf{pad}_n(\cdot)$ is non-injective. By extending the notation, $\mathsf{pad}_n(x, y) := \mathsf{pad}_n(x) \,\|\, \mathsf{pad}_n(y)$ for any arbitrary tuple (x, y). For any $x \in \{0,1\}^*$, we write $(x_1, \ldots, x_m) \xleftarrow{}_n x$ to denote the n-bit parsing of x after the padding, where $x_1 \,\|\, x_2 \,\|\, \ldots \,\|\, x_m = \mathsf{pad}_n(x)$, and $|x_i| = n$ for all $i \in [m]$. Here, $m = |x|_n$, which is called the *block length* of x (for the given n). If $x = \varepsilon$, we use the convention that $x_1 \leftarrow_n x$ and $x_1 = \mathsf{pad}_n(x) = 0^n$.

For any $m, n \in \mathbb{N}^+$, $\mathsf{F}_{m,n}$ denotes the set of all functions $f : \{0,1\}^m \to \{0,1\}^n$, and P_n the set of all permutations of $\{0,1\}^n$. For any finite set $\mathcal{X}$, $X \twoheadleftarrow \mathcal{X}$ denotes the uniform sampling of X from $\mathcal{X}$.

We identify the Galois field of order 2^n, denoted $\mathbb{F}_{2^n}$, by $\{0,1\}^n$. In this context, for any $x, y \in \{0,1\}^n$, the bitwise XOR of x and y, denoted $x \oplus y$, corresponds to the field addition operation. Additionally, we write $x \cdot y$ to denote the multiplication of field elements x and y modulo some (implicitly) fixed primitive polynomial.

Let $\mathcal{A}$ be an adversary that queries an oracle $\mathcal{O}$ in a game. We say that $\mathcal{A}$ is a distinguisher if it outputs $x \in \{0,1\}$ as an outcome. If the outcome is 1, we write $\mathcal{A}^{\mathcal{O}} = 1$ to denote this event. It is a probabilistic event whose randomness comes from that of $\mathcal{A}$ and $\mathcal{O}$. The adversary $\mathcal{A}$ may choose queries in an adaptive manner. If there are multiple oracles, $\mathcal{O}_1, \mathcal{O}_2, \ldots$ then $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \ldots}$ means the environment where $\mathcal{A}$ can query any accessible oracle in an arbitrary order. We may impose additional conditions specified by the game environment.

2.1 Block Ciphers

Let $\mathcal{K}$ be a finite set and $n \in \mathbb{N}$. A block cipher with key space $\mathcal{K}$ and message space $\{0,1\}^n$ is a keyed function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for any $K \in \mathcal{K}$, $E(K, \cdot) = E_K(\cdot)$ is a permutation over $\{0,1\}^n$. The computational security of block ciphers is defined as follows.

Definition 1 (Pseudorandom Permutation (PRP) Security). *Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathcal{A}$ be a distinguisher who queries to the encryption routine of E_K, i.e., performs a chosen-plaintext attack under uniformly sampled key $K \twoheadleftarrow \mathcal{K}$. The pseudorandom permutation (PRP) advantage of E against $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{A}) := |\Pr[\mathcal{A}^{E_K} = 1] - \Pr[\mathcal{A}^{P} = 1]|, \quad (1)$$

where $P \twoheadleftarrow \mathsf{P}_n$.

2.2 Universal Hashing

Let $\mathcal{K}_H$ be a finite set, $\mathcal{X}$ an arbitrary set, and $n \in \mathbb{N}$. We will consider a hash function $H : \mathcal{K}_H \times \mathcal{X} \to \{0,1\}^n$, and in particular how it behaves when the first input is randomly sampled.

Definition 2 (Almost Uniformity). *A hash function $H : \mathcal{K}_H \times \mathcal{X} \to \{0,1\}^n$ is said to be ϵ-almost uniform, if for any $X \in \mathcal{X}$ and any $Y \in \{0,1\}^n$*

$$\Pr_{K_H \twoheadleftarrow \mathcal{K}_H}[H_{K_H}(X) = Y] \leq \epsilon \tag{2}$$

holds for some $\epsilon \in [0,1]$.

Definition 3 (Almost (XOR) Universality). *A hash function $H : \mathcal{K}_H \times \mathcal{X} \to \{0,1\}^n$ is said to be ϵ-almost XOR universal (AXU), if for any $X \neq X' \in \mathcal{X}$ and any $Y \in \{0,1\}^n$,*

$$\Pr_{K_H \twoheadleftarrow \mathcal{K}_H}[H_{K_H}(X) \oplus H_{K_H}(X') = Y] \leq \epsilon \tag{3}$$

holds for some $\epsilon \in [0,1]$.

H is said to be ϵ-almost universal (AU) if (3) holds for $Y = 0^n$.

The following proposition is a standard result in the theory of XOR universal hash functions. See, e.g., [13].

Proposition 1. *Let $H : \mathcal{K}_H \times \mathcal{X} \to \{0,1\}^n$ be an ϵ-AXU hash function. For any $t \leq n$, $H' : \mathcal{K}_H \times \mathcal{X} \to \{0,1\}^t$ defined as $H'(K_H, X) = \lceil\!\lceil H(K_H, X) \rceil\!\rceil_t$ is $(2^{n-t} \cdot \epsilon)$-AXU.*

Polynomial Hashing. Let $p(x)$ be an irreducible polynomial of $\mathbb{F}_{2^n}$. For some fixed $\ell \in \mathbb{N}^+$, let $\mathcal{X} \subseteq \mathbb{F}_{2^n}^{\leq \ell}$ and $\mathcal{X}^+ \subset \mathcal{X}$ be the set of all non-zero vectors in $\mathcal{X}$. A polynomial hash function with respect to $p(x)$ is a function $\mathsf{POLY}^p : \mathbb{F}_{2^n} \times \mathcal{X} \to \mathbb{F}_{2^n}$, is defined for each $L \in \mathbb{F}_{2^n}$ and $(X_1, \ldots, X_k) \in \mathcal{X}$ as follows:

$$\mathsf{POLY}^p_L(X_1, \ldots, X_k) = X_1 \cdot L^k \oplus \ldots \oplus X_k \cdot L, \tag{4}$$

where the filed multiplications are done modulo $p(x)$. A closely related function $\mathsf{UPOLY}^p : \mathbb{F}_{2^n} \times \mathcal{X} \to \mathbb{F}_{2^n}$ is defined as:

$$\mathsf{UPOLY}^p_L(X_1, \ldots, X_k) = \mathsf{POLY}^p_L(X_1, \ldots, X_{k-1}) \oplus X_k. \tag{5}$$

Whenever convenient, we drop p from the notation.

Both GCM and $\mathsf{GCM\text{-}SST}$ utilize variants of POLY and UPOLY with particular instances of $p(x)$ and input encodings. GCM uses GHASH [41], while $\mathsf{GCM\text{-}SST}$ employs $\mathsf{POLYVAL}$, which was first introduced by Gueron and Lindell in [28] as part of $\mathsf{GCM\text{-}SIV}$. (See also Section 3.1.)

The set $\mathcal{X}$ is said to be *suffix-free* if for any $X \neq X' \in \mathcal{X}$, X is not a suffix of X' and vice-versa. The following propositions on the uniformity and XOR universality of POLY (and universality of UPOLY) are particularly useful in our analysis. The proofs are well-known [57,41] and therefore omitted.

Proposition 2. *The restriction of POLY on $\mathcal{X}^+$ is $2^{\log_2(\ell)-n}$-almost uniform.*

Proposition 3. *For any suffix-free $\mathcal{X}$, POLY is $2^{\log_2(\ell)-n}$-AXU and UPOLY is $2^{\log_2(\ell)-n}$-AU.*

We note that the suffix-freeness of $\mathcal{X}$ is a necessary condition, as evident from the attacks [54] on a plethora of hash-based constructions.

2.3 Authenticated Encryption

A nonce-based AE (NAE) is defined as a tuple of algorithms, $\Pi = (\Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$. The (deterministic) encryption algorithm $\Pi.\mathsf{Enc}$ takes a key $K \in \mathcal{K}$ and a tuple (N, A, M) consisting of a nonce $N \in \mathcal{N}$, an associated data (AD) $A \in \mathcal{H}$, and a plaintext $M \in \mathcal{D}$ as input, and returns a ciphertext $C \in \mathcal{D}$ and a tag $T \in \mathcal{T} = \{0,1\}^t$ for a fixed $t \in \mathbb{N}$. The (deterministic) decryption algorithm $\Pi.\mathsf{Dec}$ takes $K \in \mathcal{K}$ and the tuple $(N, A, C, T) \in \mathcal{N} \times \mathcal{H} \times \mathcal{D} \times \{0,1\}^t$ and returns $M \in \mathcal{D}$ or the reject symbol $\perp$. We require any NAE scheme to be sound: for any $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{D}$, if $(C, T) = \Pi.\mathsf{Enc}(K, N, A, M)$, $\Pi.\mathsf{Dec}(K, N, A, C, T) = M$ must hold.

Security Notions. The traditional security notions for AE schemes are nonce-respecting privacy (PRIV) and authenticity (AUTH), where the nonce can be arbitrarily chosen by the adversary as long as all nonce values in encryption queries are distinct (see e.g. [51]). Such an adversary is called nonce-respecting.

Rogaway and Shrimpton [52] proposed the concept of nonce-misuse resistance (NMR) to capture the use cases where nonces may repeat, which can happen due to, e.g., misconfiguration or poor randomness for nonce generation. Their security notions give best-possible security guarantees, however, schemes fulfilling this notion require somewhat heavier computation than nonce-respecting secure AE schemes. Especially, the privacy notion under NMR must require the encryption to be offline i.e., the first ciphertext bit can be output only after the algorithm has processed the whole input (N, A, M).

To overcome the efficiency limitation for achieving NMR security, a relaxation of the concept, called Nonce-Misuse resiLience (NML) was introduced by Ashur et al. at CRYPTO 2017 [5]. Specifically, the privacy and authenticity notions under NML divide encryption queries into *challenge* and *non-challenge* ones: in a *challenge query*, the nonce is different from those used by earlier encryption queries, whereas in a *non-challenge* query the nonce may be a repeated one. The nonce-misuse in non-challenge queries should not break the challenge ciphertexts (for privacy) or enable a forgery with the challenge nonce (for authenticity). We take the privacy and authenticity notions under NML setting, NML-PRIV and NML-AUTH, from [5,32]. A remarkable characteristic of NML notions is that it could be fulfilled by schemes whose encryption and decryption are on-line, unlike the case of NMR.

Definition 4 (Nonce-Misuse Resilience Privacy (NML-PRIV)). *Let* $\Pi = (\Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ *be an NAE scheme and let* $\mathcal{A}$ *be a distinguisher against* Π *under uniformly sampled key* $K \twoheadleftarrow \mathcal{K}$*. The nonce-misuse resilience privacy (NML-PRIV) advantage of* Π *against* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}_{\Pi}^{\text{nml-priv}}(\mathcal{A}) := \left|\Pr\left[\mathcal{A}^{\Pi.\mathsf{Enc}_K, \Pi.\mathsf{Enc}_K} = 1\right] - \Pr\left[\mathcal{A}^{\$, \Pi.\mathsf{Enc}_K} = 1\right]\right|,$$

where $\$$ *is an oracle that returns a random string of* $|M| + t$ *bits for any query* (N, A, M)*.* $\mathcal{A}$ *may re-use nonces with its right oracle* $\mathcal{O}_2$*, but it may not re-use* *nonces with its left oracle $\mathcal{O}_1$, nor may it use a nonce already queried to $\mathcal{O}_2$ for an $\mathcal{O}_1$-query and vice versa.*

We may call $\mathcal{O}_1$ the challenge oracle and $\mathcal{O}_2$ the non-challenge oracle.

Definition 5 (Nonce-Misuse Resilience Authenticity (NML-AUTH)). *Let $\Pi = (\Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ be an NAE scheme and let $\mathcal{A}$ be an adversary against Π under uniformly sampled key $K \twoheadleftarrow \mathcal{K}$. The nonce-misuse resilience authenticity (NML-AUTH) advantage of Π against $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\Pi}^{\text{nml-auth}}(\mathcal{A}) \coloneqq \Pr\left[\mathcal{A}^{\Pi.\mathsf{Enc}_K, \Pi.\mathsf{Dec}_K} \textit{ forges}\right].$$

Here, (i) nonces in encryption queries may repeat, but (ii) any nonce value in a decryption query must appear at most once in encryption queries. A trivial forgery is not allowed, namely if $\mathcal{A}$ receives (C, T) from an encryption query (N, A, M), $\mathcal{A}$ should not make a decryption query (N, A, C, T) after that query. We say $\mathcal{A}$ forges if it receives $M \neq \perp$ from the decryption oracle under these conditions.

The aforementioned (nonce-respecting) PRIV notion ($\mathbf{Adv}_{\Pi}^{\text{priv}}(\mathcal{A})$) is obtained by removing the second oracle from the NML-PRIV game from Definition 4. Similarly, the (nonce-respecting) AUTH notion ($\mathbf{Adv}_{\Pi}^{\text{auth}}(\mathcal{A})$) is obtained from Definition 5 by imposing that nonce values in the encryption queries must be distinct. As a consequence, NML-PRIV implies PRIV and NML-AUTH implies AUTH.

Remark 1. In the context of the aforementioned games, for each encryption (resp., decryption) input of the form (N, A, M) (resp., (N, A, C, T)), we will refer to (A, M) (resp., (A, C)) as the *input-data* of the query.

3 Specifications of GCM and GCM-SST

Throughout this section we fix $\kappa \geq 128$ as the key size, $n = 128$ as the block size, $s = 96$ as the nonce size, and $t \leq 128$ as the tag size. Let $r = n - s = 32$ denote the counter size, and $\mathcal{R} = \{0,1\}^r$ denote the counter space. Fix $\mathcal{K} = \{0,1\}^\kappa$, $\mathcal{N} = \{0,1\}^s$, $\mathcal{H} \subseteq \{0,1\}^*$, $\mathcal{D} \subseteq \{0,1\}^*$, and $\mathcal{T} = \{0,1\}^t$.

3.1 Internal Components of GCM and GCM-SST

We describe the well-known cryptographic components used in GCM and GCM-SST, tailored specifically to aid our description of these modes.

The CTR *Mode.* The counter or CTR mode [22] is a block cipher mode of operation, often employed as a nonce-based keystream generator.

Formally, the CTR mode can be viewed as a family of nonce-based keystream generators, indexed by the counter size, r. For any key $K \in \mathcal{K}$, any nonce $N \in \mathcal{N}$, any initial counter value $i \in \{0, \dots, 2^n - 1\}$, and any output length $m \in [2^r]$, we have:

$$\mathsf{CTR}^r(K, N, i, m) \coloneqq E_K(N \| \langle i \rangle_r) \parallel \dots \parallel E_K(N \| \langle i + m - 1 \rangle_r), \tag{6}$$

where the addition is done modulo 2^r.

The GHASH *and* POLYVAL *Functions.* As mentioned in Section 2.2, both GHASH and POLYVAL can be seen as specific instances of the larger class of polynomial hashing.

It is well-known that $g(x) = x^{128} + x^7 + x^2 + x + 1$ and $p(x) = x^{128} + x^{127} + x^{126} + x^{121} + 1$ are two irreducible polynomials of $\mathbb{F}_{2^n}$. The bit representation of p is simply the bit-reversed representation of g.

For some $\ell \in \mathbb{N}^+$, let $\mathcal{X} = \mathbb{F}_{2^n}^{\leq \ell}$. Using [23] and [28, Appendix A], one can define $\mathsf{GHASH}, \mathsf{POLYVAL} : \mathbb{F}_{2^n} \times \mathcal{X} \to \mathbb{F}_{2^n}$ for any $L \in \mathbb{F}_{2^n}$ and any $(X_1, \dots, X_k) \in \mathcal{X}$ in terms of POLY as follows:

$$\mathsf{GHASH}_L(X_1, \dots, X_k) \coloneqq \mathsf{POLY}^g_L(X_1, \dots, X_k), \tag{7}$$

$$\mathsf{POLYVAL}_L(X_1, \dots, X_k) \coloneqq \overline{\mathsf{POLY}^g_{2 \cdot \overline{L}}(\overline{X}_1, \dots, \overline{X}_k)}, \tag{8}$$

where $\overline{Y}$ denotes the byte-reversed copy of Y, i.e., $\overline{Y} \coloneqq Y_{\frac{n}{8}} \| \dots \| Y_1$ for all $Y = Y_1 \| \dots \| Y_{\frac{n}{8}}$. Thus, GHASH and POLYVAL are equivalent up to a byte-reversal of the key, encoded input, and output. More importantly, they are equivalent to POLY. Hence, Proposition 3 gives that both GHASH and POLYVAL are $2^{\log_2(\ell) - n}$-AXU as long as the input space $\mathcal{X}$ is suffix-free.

Additionally, using (5) and (8), we define:

$$\mathsf{UPOLYVAL}_L(X_1, \dots, X_k) \coloneqq \overline{\mathsf{POLY}^g_{2 \cdot \overline{L}}(\overline{X}_1, \dots, \overline{X}_{k-1})} \oplus X_k. \tag{9}$$

In this paper, GHASH and POLYVAL (or POLY, in general) are used to hash inputs from $\mathcal{H} \times \mathcal{D}$. Thus, we need to preprocess the inputs to encode them into strings from $(\{0, 1\}^n)^*$. A frequently used input encoding is the following:

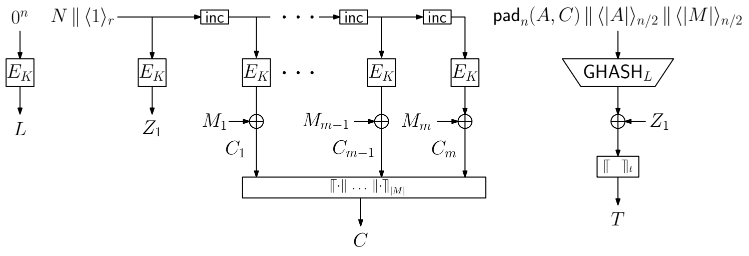
$$\boldsymbol{\rho}(A, C) = \mathsf{pad}_n(A, C) \parallel \langle |A| \rangle_{\frac{n}{2}} \parallel \langle |C| \rangle_{\frac{n}{2}} \tag{10}$$

for $(A, C) \in \mathcal{H} \times \mathcal{D}$. For $\mathcal{H} \times \mathcal{D} \subseteq \left(\{0, 1\}^{2^{\frac{n}{2}}} \right)^2$, the set $\{\boldsymbol{\rho}(A, C) : (A, C) \in \mathcal{H} \times \mathcal{D}\}$ is suffix-free.

3.2 Galois Counter Mode (GCM)

We briefly describe GCM [41,23]. In [23], GCM is defined for arbitrary nonce lengths; however, we focus only on the 96-bit case to ensure a fair comparison with GCM-SST, which, as we will see, is defined for a fixed nonce size. A simplified academic[7] version of this mode is described in Algorithms 1 and 2.

[7] This specification of GCM is inspired by the one given in [46] which in turn follows [35,41].



0^n
$N \| \langle 1 \rangle_r$
inc
inc
inc
$\mathsf{pad}_n(A,C) \| \langle |A| \rangle_{n/2} \| \langle |M| \rangle_{n/2}$
E_K
GHASH_L
L
Z_1
M_1
M_{m-1}
M_m
Z_1
C_1
C_{m-1}
C_m
$[\![\cdot\|\ldots\|\cdot]\!]_{|M|}$
$[\![\ ]\!]_t$
C
T

Fig. 1. GCM encryption process for 96-bit nonce. The inc box denotes increment in the last r bits.

Algorithm 1 The encryption algorithm of Galois Counter Mode (GCM).

```
1:  function GCM.Enc($K$, $N$, $A$, $M$)
2:      $L \leftarrow E_K(0^n)$
3:      $(M_1, \ldots, M_m) \leftarrow_n M$
4:      $(Z_1, \ldots, Z_{m+1}) \leftarrow_n \mathsf{CTR}^r(K, N, 1, m+1)$
5:      for $i = 1, \ldots, m$ do
6:          $C_i \leftarrow M_i \oplus Z_{i+1}$
7:      $C \leftarrow [\![C_1\| \ldots \|C_m]\!]_{|M|}$
8:      $D \leftarrow \mathsf{GHASH}_L\,(\boldsymbol{\rho}(A, C))$
9:      $T \leftarrow [\![Z_1 \oplus D]\!]_t$
10:     return $(C, T)$
```

Algorithm 2 The decryption algorithm of Galois Counter Mode (GCM).

```
1:  function GCM.Dec($K$, $N$, $A$, $C$, $T$)
2:      $L \leftarrow E_K(0^n)$
3:      $(C_1, \ldots, C_m) \leftarrow_n C$
4:      $(Z_1, \ldots, Z_{m+1}) \leftarrow_n \mathsf{CTR}^r(K, N, 1, m+1)$
5:      $D \leftarrow \mathsf{GHASH}_L\,(\boldsymbol{\rho}(A, C))$
6:      if $[\![Z_1 \oplus D]\!]_t = T$ then
7:          for $i = 1, \ldots, m$ do
8:              $M_i \leftarrow C_i \oplus Z_{i+1}$
9:          return $M \leftarrow [\![M_1\| \ldots \|M_m]\!]_{|C|}$
10:     else return $M \leftarrow \perp$
```

Security of GCM*.* The following security bounds on GCM are adapted from the more general bounds proven by Niwa et al. [46], tailored to the case of a 96-bit nonce.

Theorem 1 (GCM **Privacy** [46])**.** *For any PRIV adversary $\mathcal{A}$ that runs in time θ makes q encryption queries, each having input-data of length at most ℓ blocks and a total of at most σ blocks of input-data across all queries, there exists a PRP adversary $\mathcal{B}$ that runs in time $O(\sigma\theta)$ and makes $(\sigma + q + 1)$ encryption queries such that*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{GCM}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{B}) + \frac{0.5(\sigma + q + 1)^2}{2^n} + \frac{2(\sigma + q)}{2^n}.$$

Theorem 2 (GCM **Authenticity** [46])**.** *For any AUTH adversary $\mathcal{A}$ that runs in time θ and makes q encryption and v decryption queries, each having input-data of length at most ℓ blocks and a total of at most σ blocks of input-data across all queries, there exists a PRP adversary $\mathcal{B}$ that runs in time $O(\sigma\theta)$ and makes $(\sigma + q + v + 1)$ encryption queries such that*

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathsf{GCM}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(\mathcal{B}) + \frac{0.5(\sigma + q + v + 1)^2}{2^n} + \frac{2(\sigma + q + v)}{2^n} + \frac{v(\ell + 1)}{2^t}.$$

Note that the authentication bound degrades *linearly in* ℓ, which could be problematic when t is small. NIST [23] specifies $t \in \{96, 104, 112, 120, 128\}$ for general use and $t = 64$ or 32 with specific requirements and warnings.

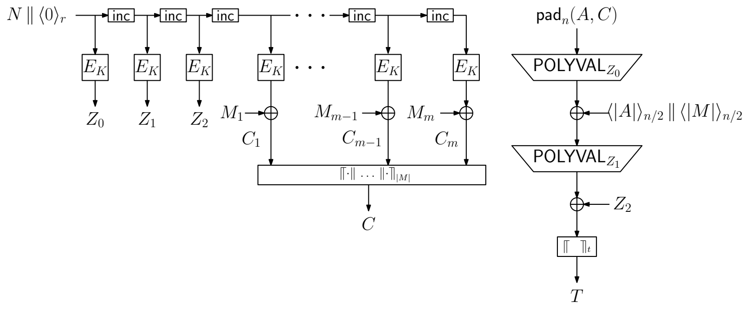


Fig. 2. GCM-SST encryption process. The inc box denotes increment in the last r bits.

3.3 Galois Counter Mode With Secure Short Tag (GCM-SST)

GCM-SST is an NAE mode that modifies GCM to generate shorter tags with comparably better security. Based on the latest available Internet draft [17], the GCM-SST algorithm differs from the GCM algorithm in the following points:

– The nonce is fixed to 96 bits;
– POLYVAL function from GCM-SIV [28] replaces the GHASH function;
– The CTR mode uses 0 as the initial counter value, unlike 1 in GCM;
– The associated data and ciphertext are hashed in two steps, unlike the single-step hashing in GCM;
– The hash keys are nonce-dependent. Specifically, the first two blocks of the keystream generated via CTR mode are used as the hash keys.

Algorithms 3 and 4 give a description of GCM-SST, using largely the same notations as used in case of GCM. See Figure 2 for an illustration of the encryption process. In [17], the tag length is specified as $t \in \{32, 64, 80\}$. Maximum lengths of associated data and message are slightly different from GCM.

Algorithm 3 The encryption algorithm of Galois Counter Mode with Secure Short Tag (GCM-SST).

```
1: function GCM-SST.Enc($K$, $N$, $A$, $M$)
2:     $(M_1, \dots, M_m) \leftarrow_n M$
3:     $(Z_0, \dots, Z_{m+2}) \leftarrow_n \mathsf{CTR}^r(K, N, 0, m+3)$
4:     for $i = 1, \dots, m$ do
5:         $C_i \leftarrow M_i \oplus Z_{i+2}$
6:     $C \leftarrow [\![C_1\| \dots \|C_m]\!]_{|M|}$
7:     $D' \leftarrow \mathsf{UPOLYVAL}_{Z_0}(\boldsymbol{\rho}(A, C))$
8:     $D \leftarrow \mathsf{POLYVAL}_{Z_1}(D')$
9:     $T \leftarrow [\![Z_2 \oplus D]\!]_t$
10:    return $(C, T)$
```

Algorithm 4 The decryption algorithm of Galois Counter Mode with Secure Short Tag (GCM-SST).

```
1: function GCM-SST.Dec($K$, $N$, $A$, $C$, $T$)
2:     $(C_1, \dots, C_m) \leftarrow_n C$
3:     $(Z_0, \dots, Z_{m+2}) \leftarrow_n \mathsf{CTR}^r(K, N, 0, m+3)$
4:     $D' \leftarrow \mathsf{UPOLYVAL}_{Z_0}(\boldsymbol{\rho}(A, C))$
5:     $D \leftarrow \mathsf{POLYVAL}_{Z_1}(D')$
6:     if $[\![Z_2 \oplus D]\!]_t = T$ then
7:         for $i = 1, \dots, m$ do
8:             $M_i \leftarrow C_i \oplus Z_{i+2}$
9:         return $M \leftarrow [\![M_1\| \dots \|M_m]\!]_{|C|}$
10:    else return $M \leftarrow \perp$
```

4 Security of GCM-SST

We prove the security of GCM-SST in the stronger security model of nonce-misuse resilience. We first set notations and conventions in Section 4.1. Then, nonce-misuse resilience privacy is derived in Section 4.2. Nonce-misuse resilience authenticity is derive in Section 4.4, and this result internally uses a core result on the XOR universality of the hashing in GCM-SST, which we derive in Section 4.3.

4.1 Notations and Conventions

Recall from Section 2.3 that, in our setting, an NML-PRIV adversary has access to two encryption oracles, namely the challenge oracle $\mathcal{O}_1$ and the non-challenge oracle $\mathcal{O}_2$. It can make queries with repeated nonce values to $\mathcal{O}_2$, but the queries to $\mathcal{O}_1$ must have distinct nonce values. In addition, the two sets of nonce values (challenge and non-challenge) must be distinct at all times. In NML-AUTH game, all nonce values used in decryption queries must appear at most once in encryption queries.

Throughout this section we use the following notations in the context of the ith adversarial query:

- the ith challenge (or nonce-respecting) query-response tuple is of the form $(N^i, A^i, M^i, C^i, T^i)$, where N^i, A^i, M^i, C^i, and T^i denote the nonce, associated data, plaintext, ciphertext and tag, respectively. Let $a_i = |A^i|_n$, $m_i = |M^i|_n = |C^i|_n$;
- the ith non-challenge (or nonce-misusing) query-response tuple is of the form $(\overline{N}^i, \overline{A}^i, \overline{M}^i, \overline{C}^i, \overline{T}^i)$, where $\overline{N}^i$, $\overline{A}^i$, $\overline{M}^i$, $\overline{C}^i$, and $\overline{T}^i$ denote the nonce, associated data, plaintext, ciphertext and tag, respectively. Let $\overline{a}_i = |\overline{A}^i|_n$ and $\overline{m}_i = |\overline{M}^i|_n = |\overline{C}^i|_n$;
- the ith decryption query-response tuple is of the form $(\widetilde{N}^i, \widetilde{A}^i, \widetilde{C}^i, \widetilde{T}^i, \widetilde{M}^i)$, where $\widetilde{N}^i$, $\widetilde{A}^i$, $\widetilde{C}^i$, $\widetilde{T}^i$ and $\widetilde{M}^i$ denote the nonce, associated data, ciphertext, tag and plaintext, respectively. Let $\widetilde{a}_i = |\widetilde{A}^i|_n$ and $\widetilde{m}_i = |\widetilde{C}^i|_n$.

Let $\ell = \max\{a_i + m_i, \widetilde{a}_i + \widetilde{m}_i, \overline{a}_i + \overline{m}_i\}$. Additionally, we write Z^i (resp., $\overline{Z}^i$ or $\widetilde{Z}^i$) to denote the keystream generated in the real world for the ith challenge (resp., non-challenge or decryption) query.

Remark 2. The CTR mode computes the keystream by encrypting the counter-encoded nonce, where the counter is incremented by one for each new block. Thus, the keystream starts repeating almost certainly once the counter value becomes 2^r. In addition, GCM-SST uses the first three blocks of the keystream in authentication phase. So, in all the results that follow, we assume the plaintext/ciphertext length to be at most $2^r - 4$ blocks.

4.2 NML Privacy of GCM-SST

Theorem 3. *For any NML-PRIV adversary $\mathcal{A}$ that runs in time θ and makes q encryption queries consisting of q_1 challenge and $q_2 = q - q_1$ non-challenge encryption queries, each having input-data of length at most ℓ blocks and a total* *of at most σ blocks of input-data across all queries, there exists a PRP adversary $\mathcal{B}$ that runs in time $O(\sigma\theta)$ and makes $(\sigma + 3q)$ encryption queries such that*

$$\mathbf{Adv}^{\text{nml-priv}}_{\mathsf{GCM\text{-}SST}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prp}}_{E}(\mathcal{B}) + \frac{0.5(\sigma + 3q)^2}{2^n}.$$

Proof. Using the PRP-PRF switch [8], we have

$$\mathbf{Adv}^{\text{nml-priv}}_{\mathsf{GCM\text{-}SST}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prp}}_{E}(\mathcal{B}) + \frac{(\sigma + 3q)^2}{2^{n+1}} + \mathbf{Adv}^{\text{nml-priv}}_{\mathsf{GCM\text{-}SST}[F]}(\mathcal{C}), \qquad (11)$$

where F denotes a uniform random function from $\mathsf{F}_{n,n}$, and $\mathcal{C}$ is a computationally unbounded and deterministic adversary making at most q_1 challenge and q_2 non-challenge encryption queries, each of length at most ℓ blocks and a total of at most σ blocks across queries.

Next, observe that $F(N \parallel \cdot)$ is independent of $F(N' \parallel \cdot)$ for distinct $N, N' \in \{0,1\}^s$. Similarly, $F(\cdot \parallel i)$ is independent of $F(\cdot \parallel i')$ for distinct $i, i' \in \{0,1\}^r$. Thus, the combined sequence

$$(Z^i_j \;:\; i \in [q_1],\, j \in \{0, m_i + 2\}) \cup (\overline{Z}^{i'}_{j'} \;:\; i' \in [q_2],\, j' \in \{0, \ldots, \overline{m}_{i'} + 2\})$$

is mutually independent, and each Z^i_j is uniformly distributed. As a consequence, the output distributions in the two worlds are identically distributed. Thus, $\mathbf{Adv}^{\text{nml-priv}}_{\mathsf{GCM\text{-}SST}[F]}(\mathcal{C}) = 0$ in (11), which completes the proof. □

4.3 XOR Universality of the Two-Stage Hashing in GCM-SST

Specification of SST *Hash.* For any positive integer ℓ and $a < \ell$, let $m = \ell - a$. For any fixed $k \leq n$, define $\mathsf{SST}^k : \mathbb{F}^2_{2^n} \times \{0,1\}^{\leq an} \times \{0,1\}^{\leq mn} \to \mathbb{F}_{2^k}$, for all $(L_0, L_1) \in \mathbb{F}^2_{2^n}$ and $(A, C) \in \{0,1\}^{\leq an} \times \{0,1\}^{\leq mn}$, by the mapping

$$\mathsf{SST}^k_{L_0,L_1}(A, C) := [\![\mathsf{POLY}_{L_1} \circ \mathsf{UPOLY}_{L_0} \circ \boldsymbol{\rho}(A, C)]\!]_k, \qquad (12)$$

where recall that $\boldsymbol{\rho}(A, C) = \mathsf{pad}_n(A, C) \parallel \langle |A| \rangle_{\frac{n}{2}} \parallel \langle |C| \rangle_{\frac{n}{2}}$. The two-stage hashing in GCM-SST is obtained by setting $k = t$ and choosing POLYVAL as the underlying polynomial hash function in the resulting SST^t hash function.

Core Result. The core result necessary for proving nonce-misuse resilience authenticity of GCM-SST, in Section 4.4, is abstracted out in the following proposition, first appearing in Stinson [56] and Bierbrauer et al. [13].

Proposition 4. *For some $\epsilon_1, \epsilon_2 \in [0,1]$, let $H : \mathcal{L} \times \mathcal{D} \to \mathcal{D}'$ and $H' : \mathcal{L}' \times \mathcal{D}' \to \mathcal{R}$ be ϵ_1-AU and ϵ_2-AXU hash functions, respectively. Then, the composition $H'' := H' \circ H : \mathcal{L} \times \mathcal{L}' \times \mathcal{D} \to \mathcal{R}$ defined as $H''_{L,L'}(X) = H'_{L'}(H_L(X))$ is an $(\epsilon_1 + \epsilon_2)$-almost XOR universal hash function.*

Proof. Let $(L, L') \twoheadleftarrow \mathcal{L} \times \mathcal{L}'$. Fix any arbitrary pair of distinct inputs $X, X' \in \mathcal{D}$ and an arbitrary difference $Y \in \mathcal{R}$. Consider the following three events from the event space of (L, L'):

$$\begin{aligned}
\mathsf{E}_1 &: H_L(X) = H_L(X'), \\
\mathsf{E}_2^L &: H'_{L'}(H_L(X)) \oplus H'_{L'}(H_L(X')) = Y, \\
\mathsf{E}_3 &: H''_{L,L'}(X) \oplus H''_{L,L'}(X') = Y.
\end{aligned}$$

Then, we have

$$\begin{aligned}
\Pr(\mathsf{E}_3) &\leq \Pr(\mathsf{E}_1) + \Pr(\mathsf{E}_3 \wedge \neg \mathsf{E}_1) \\
&\leq \epsilon_1 + \sum_{\substack{L \in \mathcal{L} \\ H_L(X) \neq H_L(X')}} \Pr\left(\mathsf{E}_2^L\right) \times \frac{1}{|\mathcal{L}|} \\
&\leq \epsilon_1 + \epsilon_2 \sum_{\substack{L \in \mathcal{L} \\ H_L(X) \neq H_L(X')}} \frac{1}{|\mathcal{L}|} \\
&\leq \epsilon_1 + \epsilon_2,
\end{aligned}$$

where the second and third inequality follows from the ϵ_1-AU and ϵ_2-AXU properties of H and H', respectively. The result now follows from the arbitrariness of X, X', and Y. □

The following corollary is a direct implication of Propositions 1, 3, and 4.

Corollary 1. *For all $\ell \geq 0$, SST^k is $(2^{\log_2(\ell)-n} + 2^{-k})$-AXU.*

4.4 NML Authenticity of GCM-SST

Theorem 4. *For any NML-AUTH adversary $\mathcal{A}$ that runs in time θ and makes q encryption queries consisting of q_1 nonce-respecting and $q_2 = q - q_1$ nonce-misusing encryption queries, and v decryption queries, each having input-data of length at most ℓ blocks and a total of at most σ blocks of input-data across all queries, there exists a PRP adversary $\mathcal{B}$ that runs in time $O(\sigma\theta)$ and makes $(\sigma + 3(q + v))$ encryption queries such that*

$$\mathbf{Adv}^{\text{nml-auth}}_{\mathsf{GCM\text{-}SST}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prp}}_{E}(\mathcal{B}) + \frac{0.5(\sigma + 3(q+v))^2}{2^n} + \frac{v\ell}{2^n} + \frac{v}{2^t}.$$

Proof. First, using the PRP-PRF switch [8], we have

$$\mathbf{Adv}^{\text{nml-auth}}_{\mathsf{GCM\text{-}SST}[E]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prp}}_{E}(\mathcal{B}) + \frac{(\sigma + 3(q+v)))^2}{2^{n+1}} + \mathbf{Adv}^{\text{nml-auth}}_{\mathsf{GCM\text{-}SST}[F]}(\mathcal{C}), \quad (13)$$

where F denotes a uniform random function from $\mathsf{F}_{n,n}$, and $\mathcal{C}$ is a computationally unbounded and deterministic adversary making at most q_1 nonce-respecting and q_2 nonce-misusing encryption queries, and at most v decryption queries, each of length at most ℓ blocks and a total of at most σ blocks across queries.

Using a similar argumentation as in the proof of Theorem 3, we have that the combined sequence of nonce-respecting encryption and decryption keystreams

$$(Z_j^i \;:\; i \in [q_1],\, j \in \{0, m_i+2\}) \cup (\widetilde{Z}_{j'}^{i'} \;:\; i' \in [v],\, j' \in \{0, \ldots, \widetilde{m}_{i'}+2\})$$

is independent of $(\overline{Z}_{j''}^{i''} \;:\; i'' \in [q_2],\, j'' \in \{0, \ldots, \overline{m}_{i''}+2\})$, the nonce-misusing encryption keystream. In addition, Z^i and $\widetilde{Z}^{i'}$ are uniformly distributed for all $i \in [q_1]$ and $i' \in [v]$, and

- Z^i is independent of Z^j for all $j \neq i \in [q_1]$;
- $\widetilde{Z}^{i'}$ is independent of Z^i for all $i \in [q_1]$, such that $\widetilde{N}^{i'} \neq N^i$.

Thus, we can completely ignore the nonce-misusing encryption queries, whence the game reduces to the usual nonce-respecting authentication game, albeit with at most q_1 nonce-respecting encryption queries and v decryption attempts. For the rest of this proof, we consider this modified game.

Let $\mathcal{T}$ denote the set of all transcripts realizable by the interaction of $\mathcal{C}$ with the encryption and decryption oracles, and $\mathcal{T}_{\mathtt{WIN}} \subseteq \mathcal{T}$ denote the subset of transcripts in which $\mathcal{C}$ succeeds in valid forgery. Recall that $\mathcal{C}$ stops as soon as it finds a valid forgery or it runs out of queries. Therefore, each $\tau \in \mathcal{T}_{\mathtt{WIN}}$ contains at most $q_1 + v$ entries and must end with a valid decryption query. Moreover, the actual output on the final decryption query is inconsequential as long as it succeeds. Indeed, without loss of generality, we assume that any decryption query output is either $\perp$ or a fixed non-$\perp$ symbol.

For $i \in [v]$, let $\mathcal{T}_{\mathtt{WIN}_i}$ denote the set of transcripts where $\mathcal{C}$ succeeds on the ith decryption attempt. To each transcript $\tau \in \mathcal{T}$, we associate a corresponding *encryption-only tuple*, denoted τ_e, that consists of all the encryption queries in τ and nothing else. Thus, the mapping $\tau \mapsto \tau_e$ is well defined. For each $i \in [v]$, let $\mathcal{E}(\mathcal{T}_{\mathtt{WIN}_i}) := \{\tau_e : \tau \in \mathcal{T}_{\mathtt{WIN}_i}\}$. By extending notations, let $\mathcal{E}(\mathcal{T}) := \{\tau_e : \tau \in \mathcal{T}\}$. By virtue of $\mathcal{C}$'s deterministic nature, there is a one-to-one correspondence between $\mathcal{T}_{\mathtt{WIN}_i}$ and $\mathcal{E}(\mathcal{T}_{\mathtt{WIN}_i})$ for each $i \in [v]$.

Let X denote the transcript random variable generated by the interaction of $\mathcal{C}$ with the encryption and decryption queries. Further, let X_e denote the corresponding encryption-only tuple. Now, by definition, we have

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{GCM\text{-}SST}[F]}^{\text{nml-auth}}(\mathcal{C}) &= \Pr(X \in \mathcal{T}_{\mathtt{WIN}}) \\
&= \sum_{\tau \in \mathcal{T}_{\mathtt{WIN}}} \Pr(X = \tau) \\
&= \sum_{i=1}^{v} \sum_{\tau \in \mathcal{T}_{\mathtt{WIN}_i}} \Pr(X = \tau) \\
&= \sum_{i=1}^{v} \sum_{\tau \in \mathcal{T}_{\mathtt{WIN}_i}} \Pr(X_e = \tau_e) \times \Pr(X = \tau \mid X_e = \tau_e). \qquad (14)
\end{aligned}$$

For $i \in [v]$, let E_i denote the event that the ith decryption query succeeds, i.e., $\mathsf{GCM\text{-}SST.Dec}(\widetilde{N}^i, \widetilde{A}^i, \widetilde{C}^i, \widetilde{T}^i) \neq \bot$. Then, for the conditional probability in (14), we have

$$\begin{aligned}\Pr(X = \tau \mid X_e = \tau_e) &\leq \Pr(\mathsf{E}_i \mid X_e = \tau_e)\\ &= \Pr\left(\mathsf{GCM\text{-}SST.Dec}(\widetilde{N}^i, \widetilde{A}^i, \widetilde{C}^i, \widetilde{T}^i) \neq \bot \mid X_e = \tau_e\right).\end{aligned}$$

Now, for the ith decryption query, we compute this conditional probability in two mutually exclusive cases:

– *Case A: $\widetilde{N}^i$ does not appear in any encryption queries.* Thus, $\widetilde{Z}_2^i$ is uniform at random and independent of all the keystream blocks appearing in the encryption queries, whence the aforementioned probability is exactly $1/2^t$;
– *Case B: $\widetilde{N}^i$ appears in an encryption query.* Let $(N^j, A^j, M^j, C^j, T^j)$ denote the unique encryption query-response tuple such that $N^j = \widetilde{N}^i$. Then, using Corollary 1 appropriately, the probability is bounded by

$$\Pr\left(\mathsf{SST}^t_{\widetilde{Z}_0^i, \widetilde{Z}_1^i}(\widetilde{A}^i, \widetilde{C}^i) \oplus \mathsf{SST}^t_{Z_0^j, Z_1^j}(A^j, C^j) = \widetilde{T}^i \oplus T^j \mid X_e = \tau_e\right) \leq \frac{\ell}{2^n} + \frac{1}{2^t}.$$

For a fixed query, one of the aforementioned two cases holds with certainty. So, we can substitute the maximum of the two probabilities in (14):

$$\begin{aligned}\mathbf{Adv}^{\text{nml-auth}}_{\mathsf{GCM\text{-}SST}[F]}(\mathcal{C}) &\leq \sum_{i=1}^{v}\left(\frac{\ell}{2^n} + \frac{1}{2^t}\right)\sum_{\tau \in \mathcal{T}_{\mathtt{WIN}_i}} \Pr(X_e = \tau_e)\\ &= \sum_{i=1}^{v}\left(\frac{\ell}{2^n} + \frac{1}{2^t}\right)\sum_{\tau_e \in \mathcal{E}(\mathcal{T}_{\mathtt{WIN}_i})} \Pr(X_e = \tau_e)\\ &\leq \sum_{i=1}^{v}\left(\frac{\ell}{2^n} + \frac{1}{2^t}\right)\sum_{\tau_e \in \mathcal{E}(\mathcal{T})} \Pr(X_e = \tau_e)\\ &\leq \frac{v\ell}{2^n} + \frac{v}{2^t},\end{aligned} \tag{15}$$

where the equality follows from the one-to-one correspondence between $\mathcal{T}_{\mathtt{WIN}_i}$ and $\mathcal{E}(\mathcal{T}_{\mathtt{WIN}_i})$, the second inequality follows from $\mathcal{E}(\mathcal{T}_{\mathtt{WIN}_i}) \subseteq \mathcal{E}(\mathcal{T})$, and the final inequality follows from the fact that X_e is an $\mathcal{E}(\mathcal{T})$-valued random variable. The result then follows from (13) and (15). □

5 Discussions

We perform a comparison between GCM and GCM-SST, mostly focusing on authenticity (as the whole point of GCM-SST is to guarantee some level of security in case of shorter tags), in Section 5.1. A discussion on the possibility to minimize the amount of mask material in GCM-SST is given in Section 5.2. Our improved universal forgery attack is described in Section 5.3.

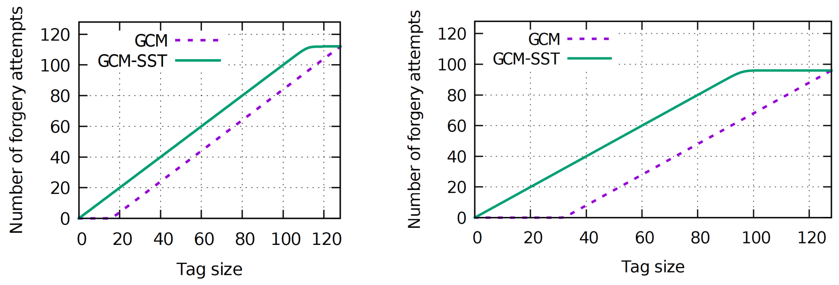


Fig. 3. The number of forgery attempts in the log scale required to break GCM and GCM-SST for each tag size, with $\ell = 2^{16}$ (left) and $\ell = 2^{32}$ (right).

5.1 Comparison of GCM and GCM-SST

We reemphasize that the results of Section 4 are also applicable in the special case of nonce-respecting security games. As a consequence, when we compare the bound in Theorem 4 for GCM-SST with the bound in Theorem 2 for GCM, the gains are significant. The primary point in comparison is authenticity degradation with respect to the tag length and the maximum input length: for GCM it is $v\ell/2^t$ (Theorem 2), while we proved it is $v\ell/2^n + v/2^t$ in the stronger security model (Theorem 4). This verifies the claim of the designers of GCM-SST in the provable security framework. For instance, consider tag size $t = 32$ and a moderate input length of $\ell = 2^{16}$ blocks. In this scenario, GCM-SST guarantees security up to 2^{32} forgery attempts, whereas GCM can guarantee security only up to 2^{16} forgery attempts. As evident from Figure 3, this gap further widens as we increase the message length or decrease the tag size.

5.2 Mask Minimization

We will investigate the possibilities to improve GCM-SST by removing the masking of Z_2 in Algorithm 3. We denote by $\overline{\mathsf{GCM\text{-}SST}}$ the AE Algorithm 3 but with Z_2 removed (thus CTR^r generates one less blocks than the original one). As the hashing subkeys Z_0 and Z_1 in $\overline{\mathsf{GCM\text{-}SST}}$ are nonce-dependent, it sounds intuitive that the resulting mode should at least guarantee privacy, and the AXU property of the hash function should also guarantee some level of authenticity.

All-zero Message Attacks on $\overline{\mathsf{GCM\text{-}SST}}$. However, it turns out that this intuition is wrong. Let N be a nonce and (Z_0, Z_1) the corresponding nonce-based hashing subkeys. Then, the SST hash function as defined in Section 4.3, and as used in $\overline{\mathsf{GCM\text{-}SST}}$, is defined as follows:

$$\mathsf{SST}^t_{Z_0,Z_1}(A,C) = [\![\mathsf{POLYVAL}_{Z_1}(\mathsf{POLYVAL}_{Z_0}(\mathsf{pad}_n(A,C)) \oplus \mathrm{len}(A,C))]\!]_t,$$

where $\mathrm{len}(A,C) := \langle |A| \rangle_{\frac{n}{2}} \parallel \langle |C| \rangle_{\frac{n}{2}}$. This is also precisely the tag value corresponding to any (N, A, M) such that $(C, T) = \overline{\mathsf{GCM\text{-}SST}}(K, N, A, M)$, noting that we dropped the masking by Z_2.

Using this observation, one can attack $\overline{\mathsf{GCM\text{-}SST}}$ in two ways:

1. The simplest and most extreme case is when one takes empty associated data and empty plaintext. In this case, the tag output is guaranteed to be 0^t, which gives an obvious privacy attack;
2. More generally, set the associated data to be a non-empty zero string, i.e., 0^a for some non-zero a, and take an empty plaintext. In this case, the resulting tag will be $T = \llbracket Z_1 \cdot \delta_a \rrbracket_t$, where $\delta_a := \langle a \rangle_{\frac{n}{2}} \parallel 0^{\frac{n}{2}}$. This tag leaks approximately t bits of Z_1. In particular, the set of valid choices for Z_1 is reduced to the set $\mathcal{L} := \{Y = (T \parallel x)/\delta_a \mid x \in \mathbb{F}_{2^{n-t}}\}$. Thus, the number of valid choices of Z_1 is reduced to at most 2^{n-t}. For a sufficiently large tag size t, the adversary can proceed to exhibit a forgery by making decryption queries of the form $(N, 0^{a'}, \varepsilon, T')$, where $a' \neq a$ and $T' := \llbracket Y \cdot \delta_{a'} \rrbracket_t$ for all $Y \in \mathcal{L}$. In particular, for $t = n$, the construction is completely broken.

A Secure Variant of $\mathsf{GCM\text{-}SST}$. The crux behind above attacks is that the adversary can render the inner hashing subkey Z_0 completely moot by choosing an arbitrary length all-zero input, and that this leaks the outer hash key Z_1. This issue is caused by the fact that $\mathsf{GCM\text{-}SST}$ employs a simple non-injective zero-padding, i.e., $\mathsf{pad}_n(\cdot)$ of Section 2. It turns out that if we would replace this padding with any injective non-zero padding, e.g., $\mathsf{pad1}_n(x) := \mathsf{pad}_n(x \parallel 1)$, the above attacks no longer work. We write $\mathsf{pad1}_n(x, y) := \mathsf{pad}_n(x) \parallel \mathsf{pad1}_n(y)$ for any tuple of strings (x, y). Denote by $\mathsf{GCM\text{-}SST}^*$ the $\overline{\mathsf{GCM\text{-}SST}}$ construction, but with $\mathsf{pad}_n(A, C)$ replaced by $\mathsf{pad1}_n(A, C)$ for any $A \in \mathcal{H}$ and $C \in \mathcal{D}$. With this padding, $\mathsf{GCM\text{-}SST}^*$ improves over $\mathsf{GCM\text{-}SST}$ in terms of the number of block cipher calls.[8]

A formal proof of the security of $\mathsf{GCM\text{-}SST}^*$ would in fact be very similar to the proofs of Theorems 3 and 4. We will nevertheless add a brief argument below. In this reasoning, we already perform the PRP-PRF switch [8] and focus on the security of $\mathsf{GCM\text{-}SST}^*[F]$ for $F \twoheadleftarrow \mathsf{F}_{n,n}$ against an adversary $\mathcal{C}$.

– *NML-PRIV Security of* $\mathsf{GCM\text{-}SST}^*$. Looking back at the proof of Theorem 3, we observe that it is sufficient to show that the tag output for each challenge encryption query is close to a uniformly random t-bit string. For any $i \in [q_1]$, consider the following two events from the event space of (Z_0^i, Z_1^i):

$$\mathsf{E}_1 : \llbracket \mathsf{POLYVAL}_{Z_1^i}(\mathsf{POLYVAL}_{Z_0^i}(\mathsf{pad1}_n(A^i, C^i)) \oplus \operatorname{len}(A^i, C^i)) \rrbracket_t = T^i,$$
$$\mathsf{E}_2 : \mathsf{POLYVAL}_{Z_0^i}(\mathsf{pad1}_n(A^i, C^i)) = \operatorname{len}(A^i, C^i).$$

Then, one has

$$(1 - \Pr(\mathsf{E}_2)) \Pr(\mathsf{E}_1 \mid \neg\mathsf{E}_2) \leq \Pr(\mathsf{E}_1) \leq \Pr(\mathsf{E}_2) + \Pr(\mathsf{E}_1 \mid \neg\mathsf{E}_2)$$
$$\left(1 - \frac{a_i + m_i + 2}{2^n}\right) \frac{1}{2^t} \leq \Pr(\mathsf{E}_1) \leq \frac{a_i + m_i + 2}{2^n} + \frac{1}{2^t}. \qquad (16)$$

[8] More precisely, if $|\mathsf{pad1}_n(A, C)|_n = |\mathsf{pad}_n(A, C)|_n$, $\mathsf{GCM\text{-}SST}^*$ reduces the number of calls by one; otherwise both need the identical number of calls.

where Proposition 2 gives $\Pr(\mathsf{E}_2) \leq (a_i+m_i+2)/2^n$, and $\Pr(\mathsf{E}_1 \mid \neg\mathsf{E}_2) = 1/2^t$ follows from the uniformity of Z_1 and 2^{n-t}-regularity[9] of $[\![\cdot]\!]_t$. Noticing that any pair of queries has independent and uniformly distributed hash keys, we infer that (16) implies that the statistical distance is bounded by $(\sigma+2q)/2^n$, whence for any adversary $\mathcal{C}$ we now have

$$\mathbf{Adv}^{\text{nml-priv}}_{\mathsf{GCM\text{-}SST}^*[F]}(\mathcal{C}) \leq \frac{\sigma + 2q}{2^n}; \tag{17}$$

– *NML-AUTH Security of* GCM-SST**.* The NML-AUTH security analysis is identical to the one for GCM-SST of Theorem 4, up to an additional factor of $(\sigma+2q)/2^n$ on account of the drop in privacy of tag output. In particular, for any adversary $\mathcal{C}$, we now have

$$\mathbf{Adv}^{\text{nml-auth}}_{\mathsf{GCM\text{-}SST}^*[F]}(\mathcal{C}) \leq \frac{\sigma + 2q}{2^n} + \frac{v\ell}{2^n} + \frac{v}{2^t}. \tag{18}$$

The final bounds will be obtained by adding the terms resulting from the PRP-PRF switch and the PRP-security of E to (17) and (18), showing that the bounds are effectively equivalent to the original ones.

5.3 Universal Forgery

Given the security results of Section 4, it also makes sense to investigate the insecurity of GCM-SST. In fact, Lindell [39] recently briefly described a universal forgery against GCM-SST with $t = 32$-bit tags (recall $t = 32$ is specified by the designers [18]). This attack recovers the subkeys Z_0 and Z_1 in around 2^{40} decryption queries, and then creates a forgery for any freely chosen (N, A, M) (hence a universal forgery attack). We present a general, improved universal forgery attack applicable to any t. When $t = 32$, our attack makes around $2^{33.6}$ decryption queries to recover Z_0 and Z_1. In general, our attack costs $3 \cdot 2^t + 4(n-t) + 1$ decryption queries. For simplicity, we will not describe the attack for the actual GCM-SST with POLYVAL, but rather for a simplified POLY that encodes an n-bit string $X = (a_0 a_1 \dots a_{n-1})$ for $a_i \in \{0,1\}$ to an $\mathbb{F}_{2^n}$-element, $a_0 x^{n-1} + a_1 x^{n-2} + a_{n-2} x + a_{n-1}$, where $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(p(x))$. The attack straightforwardly generalizes to the case one uses POLYVAL, but with some extra administration and bit manipulation, and it depends on whether t is a multiple of 8. This version of the attack is included in Appendix A for completeness.

1. Let (N, A, M) be a target tuple of a universal forgery attack, and fix any n-bit A_1 and any n-bit C_1. Denote $S = \langle |A_1| \rangle_{\frac{n}{2}} \parallel \langle |C_1| \rangle_{\frac{n}{2}}$. Find a forgery by making decryption queries (N, A_1, C_1, T_1) for varying T_1. We can write

$$T_1 = [\![Z_2 \oplus Z_1(Z_0^2 A_1 \oplus Z_0 C_1 \oplus S)]\!]_t.$$

   The complexity of this step is at most 2^t decryption queries;

[9] A surjective $f : \mathcal{D} \to \mathcal{R}$ is k-regular if and only if for each $y \in \mathcal{R}$, $|f^{-1}(y)| = k$.

2. Set $C_2 = C_1 \oplus 0^{n-1}1$. Find a forgery by making decryption queries (N, A_1, C_2, T_2) for varying T_2. The complexity of this step is also 2^t decryption queries;
3. Define $V = T_1 \oplus T_2$. Note that
$$V = \lceil\!\lceil Z_1 Z_0 (C_1 \oplus C_2) \rceil\!\rceil_t = \lceil\!\lceil Z_1 Z_0 \rceil\!\rceil_t .$$
The following steps will target the recovery of the entire value $Z_1 Z_0$;
4. We start with the $(t+1)$th bit of $Z_1 Z_0$. Set $C_3 = C_1 \oplus 0^{n-2}10$. The valid tag T_3 corresponding to (N, A_1, C_3) would satisfy
$$\begin{aligned} T_3 &= \lceil\!\lceil Z_2 \oplus Z_1 (Z_0^2 A_1 \oplus Z_0 (C_1 \oplus 0^{n-2}10) \oplus S) \rceil\!\rceil_t \\ &= \lceil\!\lceil Z_2 \oplus Z_1 (Z_0^2 A_1 \oplus Z_0 C_1 \oplus S) \oplus 2 Z_1 Z_0 \rceil\!\rceil_t = T_1 \oplus \lceil\!\lceil 2 Z_1 Z_0 \rceil\!\rceil_t . \end{aligned}$$
From this equation and from step 3 of the attack, we learn that
   - If $\lceil\!\lceil V \rceil\!\rceil_1 = 0$, then $T_3 = T_1 \oplus (\lceil\!\lceil (V \ll 1) \rceil\!\rceil_{t-1} \| b)$ for $b \in \{0,1\}$;
   - If $\lceil\!\lceil V \rceil\!\rceil_1 = 1$, then $T_3 = T_1 \oplus (\lceil\!\lceil (V \ll 1) \rceil\!\rceil_{t-1} \| b) \oplus \lceil\!\lceil 110^4 10^{120} 1 \rceil\!\rceil_t$ for $b \in \{0,1\}$.

   Find a forgery by making decryption queries (N, A_1, C_3, T_3) for $b \in \{0,1\}$. The correct value of b is then equal to the $(t+1)$th bit of $Z_1 Z_0$. The complexity of this step is 2 decryption queries;
5. Step 4 is repeated to recover the rest of $Z_1 Z_0$, bit by bit. In detail, for $s \in \{3, 4, \ldots, n-t+1\}$, assume that we have already recovered $\lceil\!\lceil Z_1 Z_0 \rceil\!\rceil_{t+s-2}$ and $\lceil\!\lceil 2^{s-2} Z_1 Z_0 \rceil\!\rceil_t$. Denote $V^* = \lceil\!\lceil 2^{s-2} Z_1 Z_0 \rceil\!\rceil_t$, and set $C^* = C_1 \oplus 0^{n-s} 10^{s-1}$. As before, the valid tag T^* corresponding to (N, A_1, C^*) would satisfy $T^* = T_1 \oplus \lceil\!\lceil 2^{s-1} Z_1 Z_0 \rceil\!\rceil_t$, and we learn that
   - If $\lceil\!\lceil V^* \rceil\!\rceil_1 = 0$, then $T^* = T_1 \oplus (\lceil\!\lceil (V^* \ll 1) \rceil\!\rceil_{t-1} \| b^*)$ for $b^* \in \{0,1\}$;
   - If $\lceil\!\lceil V^* \rceil\!\rceil_1 = 1$, then $T^* = T_1 \oplus (\lceil\!\lceil (V^* \ll 1) \rceil\!\rceil_{t-1} \| b^*) \oplus \lceil\!\lceil 110^4 10^{120} 1 \rceil\!\rceil_t$ for $b^* \in \{0,1\}$.

   Find a forgery by making decryption queries (N, A_1, C^*, T^*) for $b^* \in \{0,1\}$. The correct value of b^* is then equal to the $(t+1)$th bit of $2^{s-2} Z_1 Z_0$, and then we can recover the $(t+s-1)$th bit of $Z_1 Z_0$. The complexity of this step is 2 decryption queries, for each of $s \in \{3, 4, \ldots, n-t+1\}$. In total, the complexity of this step is $2(n-t-1)$ decryption queries;
6. Recover $Z_1 Z_0^2$ in the same manner as in steps 2–5, but now with keeping C_1 constant and varying A_2. In total, the complexity of this step is $2^t + 2(n-t)$ decryption queries;
7. Recover (Z_1, Z_0) by division: $Z_1 Z_0^2 / Z_1 Z_0 = Z_0$ (note that $Z_0 Z_1 = 0$ only holds with a negligible probability). Once this tuple is obtained, $\lceil\!\lceil Z_2 \rceil\!\rceil_t$ is determined for free from any earlier valid forgery, such as (N, A_1, C_1, T_1) from step 1;
8. When $|M| \leq |C_1|$, a forgery for the target tuple (N, A, M) can be obtained without any new query, which becomes $(N, A, M \oplus Z^\star, T)$, where $Z^\star$ is a keystream of CTR for N obtained by the forgery in step 1, and T is a valid tag for $(N, A, M \oplus Z^\star)$ that can be computed offline using the subkeys obtained in step 7. When $|M| > |C_1|$, one additional decryption query is required to recover the corresponding keystream. First, take a dummy ciphertext $C^\star$ of

the same size as M. Compute the tag $T^\star$ corresponding to $(N, A, C^\star)$ offline and make a valid forgery $(N, A, C^\star, T^\star)$ to obtain the message $M^\star$ and thus the keystream $Z^\star$ coming from CTR mode. The forgery for the target tuple (N, A, M) can then be easily obtained as $(N, A, M \oplus Z^\star, T)$, where T is a valid tag for $(N, A, M \oplus Z^\star)$.

Note that, in total, the attack indeed requires $3 \cdot 2^t + 4(n - t) + 1$ decryption queries. For $t = 32$, this value is approximately $2^{33.6}$, which improves over the complexity of 2^{40} of [39]. We also mention that step 1 can be substituted by a single encryption query, in which case the attack costs a total of 1 encryption query and approximately $2^{33.0}$ decryption queries.

Acknowledgements. We are grateful to the designers of GCM-SST (Matt Campagna, Alexander Maximov, and John Preuß Mattsson) for their valuable feedback and suggestions. We thank the anonymous reviewers for their insightful comments. The work was initiated at the Lorentz workshop “Beating Real-Time Crypto: Solutions and Analysis” held in April 2024. We thank Mustafa Khairallah, Nathalie Lang, Charlotte Lefevre, and Martijn Stam for being part of the initial discussion at the Lorentz workshop. Bart Mennink is supported by the Netherlands Organization for Scientific Research (NWO) under grant VI.Vidi.203.099. Ashwin Jha is supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa – 39078197.

References

1. New WID on Addition of 256-bit security Algorithms. 3GPP TSG-SA Meeting #103 (2023), https://www.3gpp.org/ftp/tsg_sa/TSG_SA/TSGS_103_Maastricht_2024-03/Docs/SP-240476.zip
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 168–186. Springer (2014)
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 8873, pp. 105–125. Springer (2014)
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and Authenticated Online Ciphers. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 8269, pp. 424–443. Springer (2013)
5. Ashur, T., Dunkelman, O., Luykx, A.: Boosting Authenticated Encryption Robustness with Minimal Modifications. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 10403, pp. 3–33. Springer (2017)
6. Aumasson, J., Jovanovic, P., Neves, S.: NORX: Parallel and Scalable AEAD. In: ESORICS (2). Lecture Notes in Computer Science, vol. 8713, pp. 19–36. Springer (2014)

7. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer (2000)
8. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
9. Bellare, M., Tackmann, B.: The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9814, pp. 247–276. Springer (2016)
10. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F., Udvarhelyi, B., Wiemer, F.: Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. IACR Trans. Symmetric Cryptol. 2020(S1), 295–349 (2020)
11. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.: TEDT, a Leakage-Resist AEAD Mode for High Physical Security Applications. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(1), 256–320 (2020)
12. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011)
13. Bierbrauer, J., Johansson, T., Kabatianskii, G., Smeets, B.J.M.: On Families of Hash Functions via Geometric Codes and Concatenation. In: CRYPTO. Lecture Notes in Computer Science, vol. 773, pp. 331–342. Springer (1993)
14. Böck, H., Zauner, A., Devlin, S., Somorovsky, J., Jovanovic, P.: Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. In: WOOT. USENIX Association (2016)
15. Brassard, G.: On Computationally Secure Authentication Tags Requiring Short Secret Shared Keys. In: CRYPTO. pp. 79–86. Plenum Press, New York (1982)
16. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness (2014), https://competitions.cr.yp.to/caesar.html
17. Campagna, M., Maximov, A., Mattsson, J.P.: Galois Counter Mode with Secure Short Tags (GCM-SST). Internet-Draft draft-mattsson-cfrg-aes-gcm-sst-03, Internet Engineering Task Force (2024), https://datatracker.ietf.org/doc/draft-mattsson-cfrg-aes-gcm-sst/03 (Work in Progress)
18. Campagna, M., Maximov, A., Mattsson, J.P.: Galois counter mode with secure short tags (GCM-SST). Third NIST Workshop on Block Cipher Modes of Operation 2023 (2023), https://www.amazon.science/publications/galois-counter-mode-with-secure-short-tags-gcm-sst
19. Campagna, M., Maximov, A., Mattsson, J.P.: Galois counter mode with secure short tags (GCM-SST). Presentation Slides at Third NIST Workshop on Block Cipher Modes of Operation 2023 (2023)
20. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight Authenticated Encryption and Hashing. J. Cryptol. 34(3), 33 (2021)
21. Dobraunig, C., Mennink, B.: Generalized Initialization of the Duplex Construction. In: ACNS (2). Lecture Notes in Computer Science, vol. 14584, pp. 460–484. Springer (2024)
22. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A, National Institute of Standards and Technology, U.S. Department of Commerce (2001), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

23. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, National Institute of Standards and Technology, U.S. Department of Commerce (2007), https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
24. Ekdahl, P.: Specification of the 256-bit air interface algorithms. ETSI SAGE LS to 3GPP SA3 (2022), https://www.3gpp.org/ftp/inbox/LSs_from_external_bodies/SAGE/SAGE-22-01%20LS%20to%20SA3%20on%20256-bit%20AICI.zip
25. Ekdahl, P.: Recent attack on polynomial based MACs with short tag. ETSI SAGE LS to 3GPP SA3 (2024), https://www.3gpp.org/ftp/tsg_sa/WG3_Security/TSGS3_116_Jeju/Docs/S3-242363.zip
26. Ekdahl, P.: Reply LS on request to change integrity algorithm of 256-NIA3 / 256-NCA3. ETSI SAGE LS to 3GPP SA3 (2024), https://www.3gpp.org/ftp/inbox/LSs_from_external_bodies/SAGE/SAGE-24-02%20LS%20TO%20SA3%20on%20ZUC%20MAC%20algorithm.zip
27. Ferguson, N.: Authentication Weaknesses in GCM. Public Comment to NIST (2005), http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html
28. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. RFC 8452 (Apr 2019), https://www.rfc-editor.org/info/rfc8452
29. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust Authenticated-Encryption AEZ and the Problem That It Solves. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 9056, pp. 15–44. Springer (2015)
30. IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Security. IEEE Std 802.1AE-2006 (2006)
31. Igoe, K., Solinas, J.: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. RFC 5647 (Aug 2009), https://www.rfc-editor.org/info/rfc5647
32. Inoue, A., Guo, C., Minematsu, K.: Nonce-misuse resilience of Romulus-N and GIFT-COFB. IET Inf. Secur. 17(3), 468–484 (2023)
33. Information Technology — Security Techniques — Authenticated Encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009)
34. Iwata, T., Minematsu, K., Guo, J., Morioka, S.: CLOC: Authenticated Encryption for Short Input. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 149–167. Springer (2014)
35. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and Repairing GCM Security Proofs. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 31–49. Springer (2012)
36. Joux, A.: Authentication Failures in NIST version of GCM. Public Comment to NIST (2006), http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html
37. Jutla, C.S.: Encryption Modes with Almost Free Message Integrity. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 2045, pp. 529–544. Springer (2001)
38. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: FSE. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011)
39. Lindell, Y.: Comment on AES-GCM-SST (2024), https://mailarchive.ietf.org/arch/browse/cfrg/?gbt=1&q=GCM-SST
40. Mattsson, J.P.: Private communication (2024)

41. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer (2004)
42. Mennink, B.: Understanding the Duplex and Its Security. IACR Trans. Symmetric Cryptol. 2023(2), 1–46 (2023)
43. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 8441, pp. 275–292. Springer (2014)
44. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 8441, pp. 257–274. Springer (2014)
45. NIST: Lightweight Cryptography (2019), https://csrc.nist.gov/Projects/Lightweight-Cryptography
46. Niwa, Y., Ohashi, K., Minematsu, K., Iwata, T.: GCM Security Bounds Reconsidered. In: FSE. Lecture Notes in Computer Science, vol. 9054, pp. 385–407. Springer (2015)
47. National Security Agency, Internet Protocol Security (IPsec) Minimum Essential Interoperability Requirements, IPMEIR Version 1.0.0 Core (2010), http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml
48. Nyberg, K., Gilbert, H., Robshaw, M.: Galois MAC with forgery probability close to ideal. Public Comment to NIST (2005), https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/general-comments/papers/Nyberg_Gilbert_and_Robshaw.pdf
49. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018), https://www.rfc-editor.org/info/rfc8446
50. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)
51. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: CCS. pp. 196–205. ACM (2001)
52. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer (2006)
53. Salowey, J.A., McGrew, D., Choudhury, A.: AES Galois Counter Mode (GCM) Cipher Suites for TLS. RFC 5288 (Aug 2008), https://www.rfc-editor.org/info/rfc5288
54. Shen, Y., Standaert, F., Wang, L.: Forgery Attacks on Several Beyond-Birthday-Bound Secure MACs. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 14440, pp. 169–189. Springer (2023)
55. Shoup, V.: On Fast and Provably Secure Message Authentication Based on Universal Hashing. In: CRYPTO. Lecture Notes in Computer Science, vol. 1109, pp. 313–328. Springer (1996)
56. Stinson, D.R.: Universal Hashing and Authentication Codes. In: CRYPTO. Lecture Notes in Computer Science, vol. 576, pp. 74–85. Springer (1991)
57. Taylor, R.: An Integrity Check Value Algorithm for Stream Ciphers. In: CRYPTO. Lecture Notes in Computer Science, vol. 773, pp. 40–48. Springer (1993)
58. Viega, J., McGrew, D.: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106 (Jun 2005), https://www.rfc-editor.org/info/rfc4106
59. Wegman, M.N., Carter, L.: New Hash Functions and Their Use in Authentication and Set Equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981)

60. WPA3 Specification Version 3.3. Wi-Fi Alliance (2024), https://www.wi-fi.org/system/files/WPA3%20Specification%20v3.3.pdf

A Universal forgery against the original GCM-SST

We show a complete description of a universal forgery attack against GCM-SST with POLYVAL, i.e., the original version. The strategy of the attack is the same as that in Section 5.3; after obtaining t bits of the product of Z_0 and Z_1 using 2^{t+1} decryption queries, we can recover the unknown $n-t$ bits of it bit by bit using shifting. The attack complexity does not change; $3 \cdot 2^t + 4(n-t) + 1$ decryption queries and around $2^{33.6}$ when $t = 32$.

Before going into the details of the attack, we describe how POLYVAL encodes an n-bit string because it affects how the attack actually works. For an n-bit string $X = (a_0 a_1 \ldots a_{n-1})$ for $a_i \in \{0,1\}$, POLYVAL encodes X to an $\mathbb{F}_{2^n}$-element

$$\begin{aligned}
&(a_0x^7 + a_1x^6 + a_2x^5 + \cdots + a_6x + a_7) \\
&+ (a_8x^{15} + a_9x^{14} + a_{10}x^{13} + \cdots + a_{14}x^9 + a_{15}x^8) + \cdots \\
&+ (a_{n-7}x^{n-1} + a_{n-6}x^{n-2} + a_{n-5}x^{n-2} + \cdots + a_{n-2}x^{n-6} + a_{n-1}x^{n-7}),
\end{aligned}$$

where $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(p(x))$. Thus, recalling that $\overline{X}$ denotes the byte-reversed copy of X, then we can write

$$2X = \begin{cases} \overline{\overline{X} \ll 1} & (\lceil\lceil \overline{X} \rceil\rceil_1 = a_{n-7} = 0), \\ \overline{\overline{X} \ll 1} \oplus \overline{110^4 10^{120} 1} & (\lceil\lceil \overline{X} \rceil\rceil_1 = a_{n-7} = 1). \end{cases}$$

A.1 Attack procedure (when $t = 0 \bmod 8$)

We show how the attack operates. For simplicity, we assume $t = 0 \mod 8$. The other case is similar and shown in Appendix A.2.

1. Let (N, A, M) be a target tuple of a universal forgery attack, and fix any n-bit A_1 and any n-bit C_1. Denote $S = \langle |A_1| \rangle_{\frac{n}{2}} \parallel \langle |C_1| \rangle_{\frac{n}{2}}$. Find a forgery by making decryption queries (N, A_1, C_1, T_1) for varying T_1. Using the definition of POLYVAL in [28], we can write
$$T_1 = \lceil\lceil Z_2 \oplus (Z_1 \odot Z_0 \odot Z_0 \odot A_1) \oplus (Z_1 \odot Z_0 \odot C_1) \oplus (Z_1 \odot S) \rceil\rceil_t,$$
where $Z_1 \odot Z_0 \coloneqq 2^{-n} Z_1 Z_0$. The complexity of this step is at most 2^t decryption queries;
2. Set $C_2 = C_1 \oplus 1$. Find a forgery by making decryption queries (N, A_1, C_2, T_2) for varying T_2. The complexity of this step is at most 2^t decryption queries;
3. Define $V = T_1 \oplus T_2$. Note that
$$V = \lceil\lceil Z_1 \odot Z_0 \odot (C_1 \oplus C_2) \rceil\rceil_t = \lceil\lceil (Z_1 \odot Z_0) \odot 1 \rceil\rceil_t = \lceil\lceil 2^{-n}(Z_1 \odot Z_0) \rceil\rceil_t.$$
The following steps will recover $2^{-n}(Z_1 \odot Z_0)$ denoted by W;

4. We start with the first bit of $\overline{W}$. Set $C_3 = C_1 \oplus 2$. The valid tag T_3 corresponding to (N, A_1, C_3) would satisfy

$$\begin{aligned} T_3 &= \llbracket Z_2 \oplus (Z_1 \odot Z_0 \odot Z_0 \odot A_1) \oplus (Z_1 \odot Z_0 \odot C_1) \oplus (Z_1 \odot S) \oplus (2 \odot Z_1 \odot Z_0) \rrbracket_t \\ &= T_1 \oplus \llbracket 2 \odot (Z_1 \odot Z_0) \rrbracket_t = T_1 \oplus \llbracket 2W \rrbracket_t. \end{aligned}$$

   Because t is a multiple of 8, we obtain

$$\llbracket \overline{\overline{W} \ll 1} \rrbracket_t = \llbracket \overline{\overline{(V \| 0^{n-t})} \ll 1} \rrbracket_t. \tag{19}$$

   The right side of the above equation is known from step 3. From these equations and from step 3 of the attack, we learn that
   - If $\llbracket \overline{W} \rrbracket_1 = 0$, then $T_3 = T_1 \oplus \llbracket \overline{\overline{(V \| 0^{n-t})} \ll 1} \rrbracket_t$;
   - If $\llbracket \overline{W} \rrbracket_1 = 1$, then $T_3 = T_1 \oplus \llbracket \overline{\overline{(V \| 0^{n-t})} \ll 1} \rrbracket_t \oplus \llbracket \overline{110^4 10^{120} 1} \rrbracket_t$.

   Find a forgery by making two decryption queries (N, A_1, C_3, T_3) varying T_3 as above. The correct value of T_3 reveals the unknown $\llbracket \overline{W} \rrbracket_1$. The complexity of this step is 2 decryption queries;
5. Step 4 is repeated to recover the rest of W, bit by bit. In detail, for $s \in \{3, 4, \ldots, n-t+1\}$, assume that we have already recovered $\llbracket 2^{s-2} W \rrbracket_t$. Denote $V^* = \llbracket 2^{s-2} W \rrbracket_t$, and set $C^* = C_1 \oplus 2^{s-1}$. As before, the valid tag T^* corresponding to (N, A_1, C^*) would satisfy

$$T^* = T_1 \oplus \llbracket 2^{s-1} W \rrbracket_t,$$

   and we learn that
   - If $\llbracket \overline{2^{s-2} W} \rrbracket_1 = 0$, then $T^* = T_1 \oplus \llbracket \overline{\overline{(V^* \| 0^{n-t})} \ll 1} \rrbracket_t$;
   - If $\llbracket \overline{2^{s-2} W} \rrbracket_1 = 1$, then $T^* = T_1 \oplus \llbracket \overline{\overline{(V^* \| 0^{n-t})} \ll 1} \rrbracket_t \oplus \llbracket \overline{110^4 10^{120} 1} \rrbracket_t$.

   Find a forgery by making decryption queries (N, A_1, C^*, T^*) varying T^* as above. The correct value of T^* reveals unknown $\llbracket \overline{2^{s-2} W} \rrbracket_1$, and then we can recover the corresponding bit of W. The complexity of this step is 2 decryption queries, for each of $s \in \{3, 4, \ldots, n-t+1\}$. In total, the complexity of this step is $2(n-t-1)$ decryption queries;
6. Recover $W' = 2^{-n}(Z_1 \odot Z_0 \odot Z_0)$ in the same manner as in steps 2–5, but now with keeping C_1 constant and varying A_2. In total, the complexity of this step is $2^t + 2(n-t)$ decryption queries;
7. Recover (Z_1, Z_0) by division:

$$W'/W = (2^{-n}(Z_1 \odot Z_0 \odot Z_0))/(2^{-n}(Z_1 \odot Z_0)) = 2^{-n} Z_0$$

   (note that $2^{-n}(Z_1 \odot Z_0)$ only with negligible probability). Once (Z_1, Z_0) is obtained, $\llbracket Z_2 \rrbracket_t$ can be obtained for free from any earlier valid forgery, such as (N, A_1, C_1, T_1) from step 1;
8. When $|M| \leq |C_1|$, a forgery for the target tuple (N, A, M) can be obtained without any new query. It becomes $(N, A, M \oplus Z^\star, T)$, where $Z^\star$ is a keystream of CTR for N obtained by the forgery in step 1, and T is a valid tag for $(N, A, M \oplus Z^\star)$ that can be computed offline using the subkeys obtained

in step 7. When $|M| > |C_1|$, one additional decryption query is required to recover the corresponding keystream. First, take a dummy ciphertext $C^\star$ of the same size as M. Compute the tag $T^\star$ corresponding to $(N, A, C^\star)$ offline and make a valid forgery $(N, A, C^\star, T^\star)$ to obtain the message $M^\star$ and thus the keystream $Z^\star$ coming from CTR mode. The forgery for the target tuple (N, A, M) can then be easily obtained as $(N, A, M \oplus Z^\star, T)$, where T is a valid tag for $(N, A, M \oplus Z^\star)$.

As in the attack in Section 5.3, we can substitute step 1 with a single encryption query, which leads to the attack complexity of 1 encryption query and around $2^{33.0}$ decryption queries when $t = 32$.

A.2 Attack procedure (when $t \neq 0$ mod 8)

In step 4 in Appendix A.1, (19) does not hold when $t \neq 0$ mod 8. Thus, we have to take a slightly different procedure for the attack. In the following, we only show the attack steps that differ from the one in Appendix A.1.

4. Set $C_3 = C_1 \oplus 2$. The valid tag T_3 corresponding to (N, A_1, C_3) would satisfy

$$T_3 = T_1 \oplus \lceil\!\lceil 2 \odot (Z_1 \odot Z_0) \rceil\!\rceil_t = T_1 \oplus \lceil\!\lceil 2W \rceil\!\rceil_t.$$

Because $t \neq 0 \mod 8$, we obtain

$$\lceil\!\lceil \overline{\overline{W} \ll 1} \rceil\!\rceil_t = \lceil\!\lceil \overline{\overline{(V \| b \| 0^{n-t-1})} \ll 1} \rceil\!\rceil_t,$$

for an unknown $b \in \{0,1\}$, which corresponds to $(t+1)$th bit of W. Then we learn that
   - If $\lceil\!\lceil \overline{W} \rceil\!\rceil_1 = 0$, then $T_3 = T_1 \oplus \lceil\!\lceil \overline{\overline{(V \| b \| 0^{n-t-1})} \ll 1} \rceil\!\rceil_t$ for $b \in \{0,1\}$;
   - If $\lceil\!\lceil \overline{W} \rceil\!\rceil_1 = 1$, then $T_3 = T_1 \oplus \lceil\!\lceil \overline{\overline{(V \| b \| 0^{n-t-1})} \ll 1} \rceil\!\rceil_t \oplus \lceil\!\lceil \overline{110^4 10^{120} 1} \rceil\!\rceil_t$ for $b \in \{0,1\}$.

   Find a forgery by making four decryption queries (N, A_1, C_3, T_3) varying T_3 as above for b. The correct value of T_3 reveals $\lceil\!\lceil \overline{W} \rceil\!\rceil_1$ and $(t+1)$th bit of W. The complexity of this step is 4 decryption queries;
5. Step 4 is repeated to recover the rest of W. Let $t = u (\neq 0) \mod 8$. This step has two substeps.
   (a) For $s \in \{3, \ldots, 8 - u + 1\}$, assume that we have already recovered $\lceil\!\lceil W \rceil\!\rceil_{t+s-2}$ and $\lceil\!\lceil 2^{s-2} W \rceil\!\rceil_t$. Denote $V^* = \lceil\!\lceil 2^{s-2} W \rceil\!\rceil_t$, and set $C^* = C_1 \oplus 2^{s-1}$. As before, the valid tag T^* corresponding to (N, A_1, C^*) would satisfy

   $$T^* = T_1 \oplus \lceil\!\lceil 2^{s-1} W \rceil\!\rceil_t,$$

   and we learn that
      - If $\lceil\!\lceil \overline{2^{s-2} W} \rceil\!\rceil_1 = 0$, then $T^* = T_1 \oplus \lceil\!\lceil \overline{\overline{(V^* \| b^* \| 0^{n-t-1})} \ll 1} \rceil\!\rceil_t$ for $b \in \{0,1\}$;
      - If $\lceil\!\lceil \overline{2^{s-2} W} \rceil\!\rceil_1 = 1$, then $T^* = T_1 \oplus \lceil\!\lceil \overline{\overline{(V^* \| b^* \| 0^{n-t-1})} \ll 1} \rceil\!\rceil_t \oplus \lceil\!\lceil \overline{110^4 10^{120} 1} \rceil\!\rceil_t$ for $b \in \{0,1\}$.

Find a forgery by making decryption queries (N, A_1, C^*, T^*) varying T^* as above for $b \in \{0,1\}$. The correct value of T^* reveals unknown $\lceil\!\lceil \overline{2^{s-2}W} \rceil\!\rceil_1$ and $(t+s-1)$th bit of W. Finishing this loop of s means that we recovered $\lceil\!\lceil W \rceil\!\rceil_{(t+8-u)/8}$ and the leftmost $8-u$ bits of the last byte.

(b) For $s \in \{8-u+2, \ldots, n-t-(8-u)+1\}$, assume that we have already recovered $\lceil\!\lceil W \rceil\!\rceil_{t+s-2}$ and $\lceil\!\lceil 2^{s-2}W \rceil\!\rceil_t$. The attack procedure of this substep is totally the same as step 5 in Appendix A.1. This is because

$$\lceil\!\lceil \overline{\overline{2^{s-2}W} \ll 1} \rceil\!\rceil_t = \lceil\!\lceil \overline{\overline{(V^* \| 0^{n-t})} \ll 1} \rceil\!\rceil_t$$

always holds, where $V^* = \lceil\!\lceil 2^{s-2}W \rceil\!\rceil_t$.

The complexity of this step is 4 decryption queries for each of s in substep (a), and 2 decryption queries for each of s in substep (b). In total, the complexity of this step is $4(7-u) + 2(n-t+2u-16) = 2(n-t-2)$ decryption queries;

The total complexity of the above steps 4 and 5 is $2(n-t)$, which is the same as the one in Appendix A.1. Thus, the attack complexity is $3 \cdot 2^t + 4(n-t) + 1$ decryption queries for any t.