Revisiting MAC Forgeries, Weak Keys and Provable Security of Galois/Counter Mode of Operation

Bo Zhu, Yin Tan, and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{bo.zhu,y24tan,ggong}@uwaterloo.ca

Abstract. Galois/Counter Mode (GCM) is a block cipher mode of operation widely adopted in many practical applications and standards, such as IEEE 802.1AE and IPsec. We demonstrate that to construct successful forgeries of GCM-like polynomial-based MAC schemes, hash collisions are not necessarily required and any polynomials could be used in the attacks, which removes the restrictions of attacks previously proposed by Procter and Cid. Based on these new discoveries on forgery attacks, we show that all subsets with no less than two authentication keys are weak key classes, if the final block cipher masking is computed additively. In addition, by utilizing a special structure of GCM, we turn these forgery attacks into birthday attacks, which will significantly increase their success probabilities. Furthermore, we provide a method to fix GCM in order to avoid the security proof flaw discovered by Iwata, Ohashi and Minematsu. By applying the method, the security bounds of GCM can be improved by a factor of around 2^{20}. Lastly, we show that these forgery attacks will still succeed if GCM adopts MAC-then-Enc paradigm to protect its MAC scheme as one of the options mentioned in previous papers.

Keywords: Galois/Counter Mode, GCM, MAC forgery, weak key, birthday attack, provable security, MAC-then-Enc.

1 Introduction

Information security plays an increasingly important role due to the fast growth of computer networks. How to prevent personal data from unauthorized access by third parties is one of the fundamental problems of any system design, and it highly depends on the security levels of underlying algorithms to protect confidentiality and authentication. However, in practice, system designers and software developers may have restrained time and resources to learn and understand the detailed designs and principles of sophisticated cryptographic algorithms and protocols, and may make poor decisions in their system or software development and put users' personal data in danger. Therefore, bridging

M. Abdalla, C. Nita-Rotaru, and R. Dahab (Eds.): CANS 2013, LNCS 8257, pp. 20–38, 2013.

the gap between academic research and practical developments and introducing unified interfaces for both confidentiality and authentication are very important tasks for researchers. We believe these can serve as some of the goals of the CAESAR competition calling for authenticated encryption designs [3].

Generally, block ciphers are used with various modes of operation, such as CCM, GCM and OCB, to compute ciphertexts and message authentication codes to provide confidentiality and authentication respectively. It would be very important to better investigate and understand existing designs of modes of operation when designing new authenticated encryption schemes. Galois/Counter Mode (GCM) [4,12] is an Authenticated Encryption with Associated Data (AEAD) mode [18] for block ciphers, which possesses many excellent features. GCM can be easily and efficiently implemented in both software and hardware. The computations of GCM can be done in parallel, and only small portions need to be recomputed if one block of input is changed. The theoretical proofs of GCM are given by its designers McGrew and Viega in the paper [13]. GCM is included in NSA Suite B Cryptography [15], and is widely adopted by many standards and protocols, such as IEEE 802.1AE [7] and IPsec [21].

The design of GCM is based on Counter Mode for encryption and a polynomial-based MAC scheme for authentication. The security of GCM has been assessed by many researchers [5,6,10]. Recently, the algebraic structures of its underlying polynomial-based MAC scheme were analyzed by Saarinen [19], and by Procter and Cid [16,17]. Procter and Cid showed that almost all subsets of these kinds of polynomial-based MAC schemes are weak key classes. In 2012, Iwata *et al.* found a flaw in GCM's original security proofs, and presented new security bounds for it [8,9]. Under such circumstance, further investigation on these attacks and the security bounds would be very important for usage of GCM and future designs of authenticated ciphers.

Our Contributions. The main contributions of this paper are as follows.

- We reveal (and demonstrate by practical examples) that hash collisions are not necessarily required for forgeries of GCM-like polynomial-based MAC schemes, and polynomials with non-zero constant terms can be used for the attacks. These remove certain restrictions of MAC forgery attacks proposed by Procter and Cid.
- Based on the above discoveries on MAC forgeries, we show that all non-singleton subsets (i.e. with more than one element) of authentication keys are weak key classes, if the final masking by block ciphers is computed additively. This is an extension to previous analysis of Procter and Cid.
- Based on a special structure of GCM, we show how to turn these forgery attacks into birthday-bound based attacks by attacking the encryption oracle instead of the verification or decryption oracle. This can significantly increase success probabilities and avoid certain countermeasures.
- We provide a method to fix GCM in order to avoid the security proofs' flaw discovered by Iwata *et al.* By applying this method, the security bounds of GCM can be improved by a factor of around 2^{20}.

– We indicate that even if GCM is changed to MAC-then-Enc paradigm to make adversaries more difficult to attack MAC schemes (one of the options mentioned in [16,17]), these forgery attacks can still work.

The rest of this paper is organized as follows. The next section gives the background knowledge and notation used throughout the paper. Section 3 presents our improved forgery attacks on polynomial-based MAC schemes, and studies weak key classes of GCM-like schemes. Section 4 shows how to turn these forgery attacks on GCM into birthday attacks to improve the success probabilities. The method to fix GCM and the new security bounds are given in Section 5. The attacks on the revised version of GCM in MAC-then-Enc paradigm are discussed in Section 6. The last section concludes the paper and mentions potential future work. The appendix provides several computational examples to demonstrate the MAC forgery attacks proposed in this paper.

2 Preliminaries

This section firstly clarifies the notation that will be used throughout the paper. Secondly, the design of GCM and adversarial models will be briefly introduced.

2.1 Notation

Following the notation in [8], $\mathsf{str}_n(x)$ denotes the n-bit binary representation of the integer x, where the leftmost bits are interpreted as the most significant bits (MSB) of x, and $\mathsf{int}(s)$ returns the integer converted from the bit-string s.

The operator $||$ concatenates two bit-strings, e.g. $s_1||s_2$. $\mathsf{len}(s)$ returns the bit-length of s. $\mathsf{msb}_n(s)$ represents the leftmost n bits of s, and $\mathsf{lsb}_n(s)$ is the rightmost n bits. 0^l is used to denote a bit-string with l-bit 0's, and $0^{31}1$ is the concatenation of 0^{31} with one 1. For a set $\mathcal{S}$, the number of elements in $\mathcal{S}$ is denoted as $|\mathcal{S}|$.

The function $\mathsf{inc}(s)$, where $\mathsf{len}(s) = 128$, is defined as

$$\mathsf{inc}(s) = \mathsf{msb}_{96}(s)||\mathsf{str}_{32}(\mathsf{int}(\mathsf{lsb}_{32}(s)) + 1 \bmod 2^{32}),$$

and inc^n denotes applying inc for n times.

2.2 A Brief Introduction to GCM

GCM is an AEAD scheme who adopts Counter Mode for encryption, and a polynomial-based hash algorithm for message authentication. In this paper, we concentrate on the version of GCM based on a 128-bit block cipher, which is the major usage case proposed in its specification. The finite field $\mathrm{GF}(2^{128})$ adopted in GCM uses the generating polynomial $1 + x + x^2 + x^7 + x^{128}$.

The authenticated encryption of GCM requires four bit-string inputs, an initialization vector (IV, or nonce) N, a master key K, a plaintext P and an associated data A, and then produces a pair (C, T), where C is the ciphertext which has the same length as P and T is a t-bit authentication tag, where $t \leq 128$. The

authenticated decryption algorithm takes N, K, C and T, and returns P if T is valid or *FAIL* if T does not pass the verification. The lengths of these variables should meet the following requirements [13]:

$$\begin{aligned} &0 \leq \mathsf{len}(N) \leq 2^{64}, \\ &0 \leq \mathsf{len}(P) \leq 128(2^{32}-2), \\ &0 \leq \mathsf{len}(A) \leq 2^{64}. \end{aligned}$$

We use $E_K(x)$ to denote the block cipher encryption with the master key K. Suppose $\mathsf{len}(P) = 128(n-1)+m$, where $1 \leq m \leq 128$. Segment P into a sequence of message blocks $P_1||P_2||\cdots||P_n$, where $\mathsf{len}(P_i) = 128$ for $1 \leq i \leq n-1$ and $\mathsf{len}(P_n) = m$. The authentication key H is derived from the master key by computing $H = E_K(0^{128})$.

Algorithm 1 ([13]). *The steps of GCM encryption are described as follows.*

$$\begin{aligned} N_0 &= \begin{cases} N||0^{31}1 & \text{if } \mathit{len}(N) = 96, \\ \mathit{GHASH}_H(N) & \text{if } \mathit{len}(N) \neq 96, \end{cases} \\ N_i &= \mathit{inc}(N_{i-1}) \text{ for } 1 \leq i \leq n, \\ C_i &= P_i \oplus E_K(N_i) \text{ for } 1 \leq i \leq n-1, \\ C_n &= P_n \oplus \mathit{msb}_m(E_K(N_n)) \\ C &= C_1||C_2||\cdots||C_n, \end{aligned}$$

where GHASH *is a keyed hash function that will be described later.*

GCM follows the Enc-then-MAC (EtM) paradigm, i.e. computing authentication tags from ciphertexts. The authentication tag T is computed by GMAC, defined as

$$T = \mathsf{GMAC}_{H,t}(A, C) = \mathsf{msb}_t(\mathsf{GHASH}_H(A, C) \oplus E_K(N_0)). \tag{1}$$

$\mathsf{GHASH}_H(\cdot,\cdot)$ is a polynomial-based hash function defined over $GF(2^{128})$, and $\mathsf{GHASH}_H(s)$ denotes $\mathsf{GHASH}_H(0^0, s)$, i.e. the first parameter is an empty bit-string. Suppose w and v are two bit-strings, $\mathsf{len}(w) = 128(n_1-1)+m_1$ and $\mathsf{len}(v) = 128(n_2-1)+m_2$ for $1 \leq m_1, m_2 \leq 128$. Segment w and v into $w_1||w_2||\cdots||w_{m_1}$ and $v = v_1||v_2||\cdots||v_{m_2}$ respectively, where $\mathsf{len}(w_i) = 128$ for $1 \leq i \leq n_1-1$, $\mathsf{len}(v_i) = 128$ for $1 \leq i \leq n_2-1$, $\mathsf{len}(w_{n_1}) = m_1$, and $\mathsf{len}(v_{n_2}) = m_2$. By using the following notation,

$$B_i = \begin{cases} w_i & \text{for } 1 \leq i \leq n_1-1, \\ w_i||0^{128-m_1} & \text{for } i = n_1, \\ v_i & \text{for } n_1+1 \leq i \leq n_1+n_2-1, \\ v_i||0^{128-m_2} & \text{for } i = n_1+n_2, \\ \mathsf{str}_{64}(\mathsf{len}(w))||\mathsf{str}_{64}(\mathsf{len}(v)) & \text{for } i = n_1+n_2+1, \end{cases}$$

the computation of $\mathsf{GHASH}_H(w, v)$ is defined as

$$\sum_{i=1}^{n_1+n_2+1} B_i H^{n_1+n_2+2-i}.$$

One important requirement when using GCM is that nonces must be distinct. Once an IV is reused, the counter numbers N_i used in the Counter Mode of encryption will be the same, and thus exclusive-oring two ciphertexts will eliminate the key stream and get information about plaintexts. Another reason of forbidding IV reuse is well explained in Joux's *forbidden attack* [10], i.e. same nonces will result in identical $E_K(N_0)$ used in the equation (1) and by exclusive-oring two authentication tags we will get an equation on H over finite fields that may be easily solved.

For simplicity, in the following content, A, P and C are considered being multiples of 128 bits, and N is also a multiple of 128 bits if $\mathsf{len}(N) \neq 96$, such that all inputs do not need to be padded. If not stated explicitly, A is regarded as an empty bit-string. Moreover, as in [17], the indices of input blocks are reversed, e.g. $P = P_n||P_{n-1}||\cdots||P_1$ instead of $P = P_1||P_2||\cdots||P_n$, for convenience of polynomial representations.

2.3 Security Definitions

For a fixed but unknown master key K of GCM, adversaries are given two oracles, *encryption oracle* and *decryption oracle*. Adversaries can feed a tuple (N, P) to the encryption oracle to get (C, T), or query the decryption oracle with (N, C, T). The decryption oracle will return P if T passes verification, or *FAIL* otherwise. Adversaries are assumed to be nonce-respecting, i.e. no repeating nonces are queried to the encryption oracle, which is not allowed in GCM or Counter Mode.

One of adversaries' goals is to construct *MAC forgeries*. In this case, adversaries aim to create a valid authentication tag T for (N, C), which has not been queried yet. Adversaries can make any queries except (N, C) to the encryption and decryption oracles. If adversaries target only MAC schemes, they can be given two oracles, *authentication oracle* and *verification oracle*. The authentication oracle produces T for queried (N, C); while the verification oracle returns *FAIL* if T is not valid for (N, C), or returns *PASS* otherwise.

Analysis of a cryptographic algorithm's *weak keys* is a very important assessment. Handschuh and Preneel give a theoretical definition of weak keys for symmetric cryptosystems in [6]: "A class of keys is called *weak* if for members of the class the algorithm behaves in an unexpected way and if it is easy to detect whether a particular key belongs to this class." For example, for a MAC scheme, the unexpected behavior may be that MAC forgeries can be made in a very high probability. Moreover, to determine whether a key is in the class $\mathcal{K}$, the number of queries has to be fewer than exhaustive search's, i.e. $|\mathcal{K}|$.

3 Revisiting Weak Keys of Polynomial-Based MACs

In [16,17], Procter and Cid study the weak keys and MAC forgeries of polynomial-based MAC schemes, including the one used in GCM. This is a more general model upon Saarinen's cycling attack [19].

The main framework of MACs, in which they are interested, is based on *evaluation hash* [20]. Let $\mathbb{F}$ be a finite field of characteristic 2, $H \in \mathbb{F}$ be the authentication key, and $M = M_m||M_{m-1}||\cdots||M_1$ be a message to be authenticated, where $M_i \in \mathbb{F}$. Define a polynomial $g_M(x) \in \mathbb{F}[x]$ as

$$g_M(x) = \sum_{i=1}^{m} M_i x^i.$$

Then the function $h_H(M) = g_M(H)$ is called *evaluation hash*. The hash function outputs are masked by block cipher encryptions to produce the authentication tags, such as $E_K(N) \oplus h_H(M)$ and $E_K(h_H(M))$. Poly1305-AES [2], and the MAC schemes in GCM and SGCM [19] are all within this framework.

We summarize the main observation by Procter and Cid in [17] as follows. For the convenience of the readers, we include a short proof of their result.

Result 1 ([17]). *With the same notation as above, if there exists a polynomial $f(x) \in \mathbb{F}[x]$ without a constant term, such that $f(H) = 0$, then forgeries of MAC schemes based on the evaluation hash $h_H(x)$ can be made.*

Proof. Assume

$$f(x) = \sum_{i=1}^{n} F_i x^i,$$

and $F = F_n||F_{n-1}||\cdots||F_1$. Given a message M, we have

$$h_H(M \oplus F) = g_{M\oplus F}(H) = g_M(H) \oplus f(H) = g_M(H) = h_H(M),$$

where the shorter one of M and F in $M \oplus F$ is padded with zeros. We obtain a collision on the evaluation hash, and thus a MAC forgery of the MAC scheme. □

After obtaining a valid tuple (N, C, T) by eavesdropping or active querying, the adversaries query the verification oracle about $(N, C \oplus F, T)$. If the result is not *FAIL*, then a valid MAC is forged. Please note that the polynomial $f(x)$ always has x as its factor, and is in the ideal $\langle x^2 \oplus Hx\rangle$.

For an unknown H, the success probability of MAC forgery is directly related to the choice of $f(x)$. Procter and Cid propose three ways to select $f(x)$: (1) The first way is to use $f(x) = x\prod_i(x \oplus H_i)$ to involve as many H_i as desired; (2) The second way is based on irreducible factors of $x^{2^{128}} \oplus x$, which includes Saarinen's cycling attack as a special case; (3) The third is just using random polynomials.

In the next section, we will show that, a MAC forgery can also be made for any polynomial $f(x) \in \mathbb{F}[x]$, which is an extension of Result 1.

Moreover, based on these analyses, Procter and Cid point out that almost any subset of the key space of these polynomial-based MAC schemes is a weak key class.

Result 2 ([17]). *Let $\mathcal{H}$ be a subset of the authentication key space of the MAC scheme based on evaluation hash. If $0 \in \mathcal{H}$ and $|\mathcal{H}| \geq 2$, or $|\mathcal{H}| \geq 3$, then $\mathcal{H}$ is weak.*

Proof. If $|\mathcal{H}| \geq 2$ and $0 \in \mathcal{H}$, one query forged by $f(x) = x\prod_i(x \oplus H_i)$ can be fed into the verification oracle, where $H_i \in \mathcal{H}$. To further determine whether 0 is in the set $\mathcal{H}$, two queries by distinct $f(x) \in \langle x^2 \oplus Hx\rangle$ have to be made, so all elements in a subset $|\mathcal{H}| \geq 3$ can be detected by using two queries. □

3.1 New Improved MAC Forgery Attacks

The MAC forgery attacks proposed by Procter and Cid are constructed upon hash collisions, and one of the attacks' restrictions is that the chosen polynomial $f(x)$ should always have x as a factor, or equivalently do not have a constant term. We will demonstrate below how to create MAC forgeries not based on hash collisions, and without the zero constant term restriction.

For the MAC schemes as in GCM and SGCM, whose final masking by block ciphers is computed additively, we give the following theorem, where the notation is the same as above.

Theorem 1. *Given any polynomial $q(x) \in \mathbb{F}[x]$ such that $q(H) = 0$, for the evaluation hash based MAC scheme $T = E_K(N) \oplus h_H(M)$, a MAC forgery can be constructed.*

Proof. Let Q^* be the concatenation of coefficients $Q_n||Q_{n-1}||\cdots||Q_1$ without Q_0, and $q(x) = q^*(x) \oplus Q_0$. Since $q(H) = 0$, we have

$$T = h_H(M) \oplus E_k(N) = h_H(M) \oplus E_k(N) \oplus q(H),$$

which implies

$$\begin{aligned} T \oplus Q_0 &= E_k(N) \oplus h_H(M) \oplus q^*(H) \\ &= E_k(N) \oplus g_M(H) \oplus q^*(H) \\ &= E_k(N) \oplus g_{M \oplus Q^*}(H). \end{aligned}$$

This means if we know a polynomial $q(x)$ such that $q(H) = 0$, we can exclusive-or coefficients of $q(x)$'s non-constant terms with the captured message, to obtain a valid tuple as $(N, M \oplus Q^*, T \oplus Q_0)$, if the authentication tag T is computed as $E_k(N) \oplus h_H(M)$. □

Please note that the method in the above proof does not rely on a hash collision, and the constant term Q_0 is not required to be zero. We also want to mention that Theorem 1 leads us to an extension to the original analysis of Procter and Cid on weak keys, which will be discussed in the next subsection.

A practical attack example on GCM, by using the method in Theorem 1 (along with a length extension technique), is given in Appendix A.1.

For the sake of completeness, we also give the following theorem, which works for both $E_K(N) \oplus h_H(M)$ and $E_K(h_H(M))$.

Theorem 2. *Given any polynomial $q(x) \in \mathbb{F}[x]$ such that $q(H) = 0$, a forgery can be made on the MAC schemes based on evaluation hash by using $\alpha(x)q(x)$, where $\alpha(x)$ is a polynomial without a constant term.*

Proof. Since $q(H) = 0$, we have $\alpha(H)q(H) = 0$. Because $\alpha(0) = 0$, $\alpha(0)q(0) = 0$. Therefore, we can apply the same method in Result 1 to construct hash collisions and thus MAC forgeries. □

Theorem 2 can be seen as covered by the analysis of Procter and Cid, since $\alpha(x)q(x)$ is still in the ideal $\langle x^2 \oplus Hx \rangle$. However, Theorem 2 is insufficient to deduce the result about weak key classes (Theorem 3 in the next subsection) supported by Theorem 1.

3.2 All Non-singleton Subsets of Keys are Weak

To detect whether an authentication key H is in a subset $\mathcal{H}$ of the key space, the number of queries should be less than $|\mathcal{H}|$. If $|\mathcal{H}| = 2$, only one query can be made, and thus whether zero is in $\mathcal{H}$ cannot be determined by using polynomials in $\langle x^2 \oplus Hx \rangle$, since it will need at least two queries. However, based on the analysis of Theorem 1, we may use polynomials in $\langle x \oplus H \rangle$ instead of $\langle x^2 \oplus Hx \rangle$ to make one query and determine whether the authentication key is in $\mathcal{H}$.

Theorem 3. *For an evaluation hash based MAC scheme, $T = E_K(N) \oplus h_H(M)$, if given a valid tuple (N, M, T), then making one query to the verification oracle is enough to determine whether the authentication key $H \in \mathbb{F}$ in use is in a subset of keys $\mathcal{H} = \{H_1, H_2, \cdots, H_n\} \subseteq \mathbb{F}$.*

Proof. First define a polynomial

$$q(x) = \sum_{i=0}^{n} Q_i x^i = \prod_{i=1}^{n} (x \oplus H_i),$$

where $Q_i \in \mathbb{F}$ for $0 \leq i \leq n$. Let $M' = M \oplus Q^*$ and $T' = T \oplus Q_0$ with zero pre-padding for shorter strings, where $Q^* = Q_n || Q_{n-1} || \cdots || Q_1$. Query the verification oracle with the tuple (N, M', T'). If the verification oracle does not return *FAIL*, the authentication key H in use is known to be in $\mathcal{H}$. H is not in $\mathcal{H}$ if *FAIL* is returned.

It is easy to see H is in $\mathcal{H}$ if and only if (N, M', T') passes. If H is in $\mathcal{H}$, then $q(H) = 0$, and thus (N, M', T') is valid. On the other hand, the validity of (N, M', T') implies $q(H) = 0$, so H must be a root of $q(x) = 0$, which is among all the elements of $\mathcal{H}$. □

The steps in Theorem 3 are similar to those in [17], except the absence of the steps to determine whether 0 is in $\mathcal{H}$.

Based on Theorem 3, we have the following corollary about weak key classes.

Corollary 1. *For an evaluation hash based MAC scheme, $T = E_K(N) \oplus h_H(M)$, any subset of authentication key space, $\mathcal{H}$, is weak if $|\mathcal{H}| \geq 2$.*

Proof. Due to Theorem 3, after obtaining a valid tuple (N, M, T) by passive eavesdropping, whether the authentication key H in use is the subset $\mathcal{H}$ can

be determined by only one query, which is efficient compared to the size of the subset, i.e. $1 < |\mathcal{H}|$.

On the other hand, once H is known to be in the subset $\mathcal{H}$, H is a solution for $q(x) = \prod_{i=1}^{n}(x \oplus H_i) = 0$, where H_i's are all elements of $\mathcal{H}$. Then the polynomial $\alpha(x)q(x)$ with an arbitrary non-zero $\alpha(x)$ can be used to construct more MAC forgeries. □

4 Turning MAC Forgeries into Birthday Attacks

In [8], Iwata *et al.* find a flaw in the security proofs of GCM given by McGrew and Viega in [13]. The main problem is that inc may be translated to multiple distinct forms in terms of exclusive-ors, such that the equation

$$\mathsf{inc}^{r_1}(\mathsf{GHASH}_H(N^a)) = \mathsf{inc}^{r_2}(\mathsf{GHASH}_H(N^b)) \tag{2}$$

may have many more solutions than the desired $l_N + 1$ for any given r_1, r_2, N^a and N^b, where $0 \le r_1, r_2 \le 2^{32} - 2$, $N^a \ne N^b$, and l_N is the maximum number of blocks for nonces.

Result 3 ([8]). *For a randomly chosen H, the probability for the equation (2) to hold is at most*

$$2^{22}(l_N + 1)/2^{128}.$$

Furthermore, for n queries to the encryption oracle with the nonces N^i's, where $1 \le i \le n$, the probability of having a collision on counter numbers, i.e. $N^a_{r_1} = N^b_{r_2}$ for certain r_1, r_2, a and b, is at most

$$\frac{2^{22}(n-1)(\sigma + n)(l_N + 1)}{2^{128}}, \tag{3}$$

where $0 \le r_1, r_2 \le 2^{32} - 2$, $1 \le a, b \le n$, the total length of plaintexts is at most σ blocks, and N^a and N^b are the corresponding nonces for the counter numbers $N^a_{r_1}$ and $N^b_{r_2}$ respectively.

4.1 New Birthday-Bound-Based MAC Forgery Attacks on GCM

The original forgery attacks on polynomial-based MAC schemes, including our attacks described in Section 3.1, are targeting algebraic properties of underlying evaluation hash functions, e.g., GHASH in the case of GCM. The forged queries cannot be fed to the encryption oracle directly because two queries with identical nonces are forbidden.

The work by Iwata *et al.* reminds us that GCM has a very special design, in which GHASH is reused for generating initial counter numbers if $\mathsf{len}(N) \ne 96$. This makes GHASH attackable in the encryption oracle. Precisely, assuming $H \ne 0$, the attack consists of the following three steps:

1. Either passively or actively obtain a valid tuple (N, P, C), where $\mathsf{len}(N) \neq 96$. Please note that we do not need the authentication tag T here.
2. Construct a polynomial $q(x)$, and properly apply $x^d q(x)$ to N to derive N', where $d \geq 1$. Feed the pair (N', P) to the encryption oracle, and get the corresponding ciphertext C'. If $C' = C$, we know that $q(H) = 0$.
3. Apply $q(x)$ to other captured messages and tags to construct more forgeries, or recover the authentication key by binary search or solving $q(x) = 0$.

If $H = 0$, the outputs of GMAC will be the same, and thus it can be easily detected.

One advantage of targeting the encryption oracle is that we can collect all query results into a set to perform birthday attacks. For any query to the encryption oracle, we can always get corresponding ciphertext and tag as long as the nonce is not previously queried. Using the same notation in the specification of GCM in Algorithm 1, collect $E_K(N_1)$'s, which are derived from exclusive-oring P_1's with C_1's, into a set $\mathcal{S}$. If a collision occurs in $\mathcal{S}$, e.g. $E_K(N_1^a) = E_K(N_1^b)$, where N_1^a and N_1^b are the corresponding first counter numbers for the nonces N^a and N^b, then we have $N_1^a = N_1^b$ as well. Hence a collision $\mathsf{GHASH}_H(N^a) = \mathsf{GHASH}_H(N^b)$ is found. This birthday collision attack can have a significantly higher success probability than the original attacks on the verification or decryption oracle.

Assume the polynomial $q(x)$ is chosen randomly and independently, and $H \neq 0$. The success probability for the original trial-and-error method on the verification or decryption oracle is

$$n(l_N + 1)/2^{128}, \tag{4}$$

where n is the number of queries that have been made; while the upper bound for the probability of the birthday attack is (see Lemma A.9 in Section A.4 of [11])

$$0.5 \cdot n^2(l_N + 1)/2^{128}. \tag{5}$$

In addition to the first encrypted counter blocks, we can also collect the following blocks into $\mathcal{S}$, in which way we may achieve even larger collision probabilities. For example, $E_K(N_i^a)$ may be equal to $E_K(N_j^b)$ for certain i and j. The collision probability for this case can be obtained from the equation (3) in Result 3. Although the success probability of this case is higher than the previous methods of trial-and-error and birthday attacks, the collision $N_i^a = N_j^b$ may need more time complexity to be utilized for MAC forgery attacks. One naive way is to try every polynomial over the finite field that can be converted from inc^r with the specific r, and this will cost 2^{22} time at most.

Moreover, if certain countermeasures on the decryption or verification oracle are carried out, such as forbidding nonce reuse, the original attacks would fail or be detected, but the attacks on the encryption oracle will be unaffected.

A practical attack example on non-96-bit nonces is given in Appendix A.2.

5 Revisiting Provable Security of GCM

After pointing out the flaw in GCM's original security proofs, Iwata *et al.* give new security bounds, which are characterized by *privacy advantage* and *authenticity advantage*. Please refer to [13,8,9] for the detailed definitions of privacy and authenticity advantages.

Result 4 ([16,17]). *The privacy advantage of GCM is at most*

$$\frac{0.5(\sigma+q+1)^2}{2^{128}} + \frac{2^{22}q(\sigma+q)(l_N+1)}{2^{128}}, \tag{6}$$

and the upper bound for the authenticity advantage is

$$\frac{0.5(\sigma+q+q'+1)^2}{2^{128}} + \frac{2^{22}(q+q'+1)(\sigma+q)(l_N+1)}{2^{128}} + \frac{q'(l_A+1)}{2^t}, \tag{7}$$

where the total length of plaintexts is at most σ blocks, q and q' are numbers of encryption and decryption queries respectively, and l_N and l_A are the maximum numbers of blocks for nonces and inputs respectively.

Generally, the values of the equations (6) and (7) are dominated by their second terms, since they have a large constant 2^{22}.

5.1 Repairing GCM and Its Security Bounds

Here we propose a method to fix the design of GCM such that the large constant 2^{22} in the equations (6) and (7) can be reduced to 2^2. Since the flaw of the GCM's security proofs originates from the operation inc as explained in the previous section, we aim to replace the functionality of inc with operations in the finite field.

Consider $w \cdot x$, where w is a primitive element of $\mathbb{F}_{2^n}$. It is clear that the outputs of $w \cdot x$ consist of two cycles, namely (0) and $(1, w, \ldots, w^{2^n-2})$. Now define a new function L_w as

$$L_w(x) = \begin{cases} w \cdot x & \text{if } x = w^i,\ 0 \le i \le 2^n - 3, \\ 0 & \text{if } x = w^{2^n-2}, \\ 1 & \text{if } x = 0. \end{cases} \tag{8}$$

The following theorem is important for our discussions in this subsection.

Theorem 4. *Let L_w be the function defined above, and f, g be two functions defined on $\mathbb{F}_{2^n}$ with $f(0), g(0) \neq 0$. Denoting $\deg(f) = d_1, \deg(g) = d_2$ and $d = \max(d_1, d_2)$, we have*

$$\max_{0 \le r \le 2^n-1} |\{x : x \in \mathbb{F}_{2^n} | L_w^r(f(x)) + g(x) = 0\}| \le 4d.$$

Proof. Now we consider the number of solutions of the equation

$$L_w^r(f(x)) + g(x) = 0, \tag{9}$$

where $0 \le r \le 2^n - 1$. The equation (9) can be divided into the following cases.

1. If $f(x) = 0$,
   (a) If $L_w^r(f(x)) = 0$, then $g(x) = 0$.
   (b) If $L_w^r(f(x)) \neq 0$, then $g(x) = w^{r-1}$.
2. If $f(x) \neq 0$,
   (a) If $L_w^r(f(x)) = 0$, then $g(x) = 0$.
   (b) If $L_w^r(f(x)) \neq 0$, let $f(x) = w^{r_1}$ and $L_w^r(f(x)) = w^{r_2}$, where $0 \le r_1, r_2 < 2^n - 1$. Then we have
      i. If $r_1 \le r_2$, then $w^r f(x) = g(x)$.
      ii. If $r_1 > r_2$, then $w^{r-1} f(x) = g(x)$.

Therefore, for a given r, any solution of the equation $L_w^r(f(x)) + g(x) = 0$ must be one of the solutions of the four equations

$$\begin{cases} g(x) = 0, \\ g(x) = w^{r-1}, \\ w^r f(x) = g(x), \\ w^{r-1} f(x) = g(x). \end{cases}$$

The total number of solutions for these four equations are at most $2d_2 + 2d \le 4d$. □

It is known that the detailed design of the next counter function of Counter Mode is not important as long as counter numbers are produced uniquely [14]. If the underlying block cipher is ideal, i.e. treated as a pseudorandom permutation PRP for randomly chosen encryption key, $\mathsf{PRP}(L_w^r(s))$ is indistinguishable from $\mathsf{PRP}(\mathsf{inc}^r(s))$. Therefore, the Counter Mode encryption in GCM will have same security properties as original if inc is replaced by L_w defined over $\mathbb{F}$. We propose the following revised design of GCM.

Algorithm 2. *The encryption steps of the revised GCM, denoted by LGCM, are as follows.*

$$\begin{aligned} N_0 &= \mathit{GHASH}_H(N), \\ N_i &= L_w^i(N_0) \text{ for } 1 \le i \le n, \\ C_i &= P_i \oplus E_K(N_i) \text{ for } 1 \le i \le n-1, \\ C_n &= P_n \oplus \mathit{msb}_m(E_K(N_n)), \\ C &= C_1 || C_2 || \cdots || C_n, \end{aligned}$$

where the notation is the same as in Algorithm 1.

Please note that nonces are always processed by GHASH regardless of nonces' lengths, for simplicity of security proofs.

Based on Theorem 4, we can have the following lemma.

Lemma 1. *Randomly choosing an authentication key H, the probability to have*

$$L_w^{r_1}(\mathsf{GHASH}_H(N_1)) = L_w^{r_2}(\mathsf{GHASH}_H(N_2)) \tag{10}$$

is no more than $4(l_N+1)/2^{128}$ for any given r_1, r_2, N_1 and N_2, where $0 \le r_1, r_2 \le 2^{32}-2$, $N_1 \neq N_2$, and l_N is the maximum number of blocks for nonces.

Proof. Without loss of generality, assume $r_2 \le r_1$, then the equation (10) is equivalent to

$$L_w^{r_1-r_2}(\mathsf{GHASH}_H(N_1)) = \mathsf{GHASH}_H(N_2). \tag{11}$$

The maximum degree of $\mathsf{GHASH}_H(N_1)$ and $\mathsf{GHASH}_H(N_2)$ is l_N+1, so by applying Theorem 4 we know the probability for the equation (11) to hold is $4(l_N+1)/2^{128}$ for a randomly chosen H. □

Now we can give the security bounds of LGCM as follows.

Theorem 5. *For LGCM, the revised GCM algorithm defined in Algorithm 2, the privacy advantage is at most*

$$\frac{0.5(\sigma+q+1)^2}{2^{128}} + \frac{4q(\sigma+q)(l_N+1)}{2^{128}}, \tag{12}$$

and the new upper bound for the authenticity advantage is

$$\frac{0.5(\sigma+q+q'+1)^2}{2^{128}} + \frac{4(q+q'+1)(\sigma+q)(l_N+1)}{2^{128}} + \frac{q'(l_A+1)}{2^t}, \tag{13}$$

where the notation is the same as in Result 4.

Proof. The proofs of Theorems 1 and 2 in [9] can be carried over by using Lemma 1 in the paper to replace the original probability statement of counter number collisions. □

Implementation against Timing-Based Side-Channel Attacks

The functions defined in (8) have vulnerabilities for timing-based side-channel attacks since the computations will have inconsistent times for different inputs. To minimize such effects, we may use the following equations in practical implementations.

$$\begin{aligned} y &= w \cdot x, \\ L_w(x) &= \begin{cases} 1 & \text{if } y = 0, \\ 0 & \text{if } y = 1, \\ y & \text{otherwise.} \end{cases} \end{aligned} \tag{14}$$

The equations (14) would have very close computational time costs for different branches.

We want to make a note here that it might be possible to directly adopt $w \cdot x$ instead of $L_w(x)$ to generate counter numbers since the probability for GHASH to output zero is low, but the security proofs for GCM may require to be largely rewritten and new bounds might have different formats as existing ones. We leave this as an open problem for interested readers.

6 Attacking GCM in MAC-then-Enc Mode

GCM follows the Enc-then-MAC paradigm, i.e. authentication tag is computed based on ciphertexts. It is known that once the integrity of the system is compromised, the whole system including privacy will not be trustworthy. For GCM, if we successfully perform a MAC forgery attack described in previous sections, e.g., a forged tuple (N, C', T'), based on a valid (N, C, T), is fed to the decryption oracle and passes verification, the oracle will return P' that may have a known linear difference with P. In this way, P can be obtained even without any knowledge of the encryption key. Therefore, the message authentication algorithm must be well protected.

One potential and straightforward option, which is indicated in [16,17], is to change GCM to a MAC-then-Enc scheme (MtE GCM, thereafter). More precisely, in MtE GCM, GMAC is computed based on plaintexts instead of ciphertexts, and the authentication tag is encrypted by block ciphers in Counter Mode.

However, we find that the MAC forgery attacks described in previous sections may still work on MtE GCM. These attacks are based on the linear properties of the polynomial-based MAC schemes. Assuming no length extension is needed, applying $q(x)$ directly to ciphertexts and encrypted tags may successfully result in MAC forgeries. Consider the simplified case with

$$\begin{aligned} ET &= h_H(P) \oplus E_K(N) \oplus E_K(N_t) \\ &= h_H(P) \oplus \mathsf{Mask} \\ &= h_H(C \oplus S) \oplus \mathsf{Mask}, \end{aligned}$$

where ET is the encrypted authentication tag, $E_K(N_t)$ is to encrypt the authentication tag, $\mathsf{Mask} = E_K(N) \oplus E_K(N_t)$, S is the key stream produced by Counter Mode, and the other variables are the same as in previous analyses. If we know a function $q(x)$ such that $q(H) = 0$, then

$$\begin{aligned} ET' = ET \oplus Q_0 &= h_H(C \oplus S) \oplus q^*(H) \oplus \mathsf{Mask} \\ &= g_{C \oplus S}(H) \oplus g_{Q^*}(H) \oplus \mathsf{Mask} \\ &= g_{C \oplus Q^* \oplus S}(H) \oplus \mathsf{Mask} \\ &= h_H(C \oplus Q^* \oplus S) \oplus \mathsf{Mask} \\ &= h_H(C' \oplus S) \oplus \mathsf{Mask}. \end{aligned}$$

This implies the tuple (N, C', ET'), where $C' = C \oplus Q^*$ and $ET' = ET \oplus Q_0$, will pass the verification oracle of MtE GCM. A computational example is given in Appendix A.3.

If $\mathsf{len}(Q^*) > \mathsf{len}(C)$, i.e. length extension is needed, the above attack on MtE GCM may not work. To decrypt $C \oplus Q^*$, where $\mathsf{len}(C \oplus Q^*) > \mathsf{len}(C)$, the verification oracle will produce longer key stream $S' = S||S_u$ with an unknown portion S_u, so outputs of the oracle will become unpredictable. However, adversaries may avoid this by trying to attack GHASH in the encryption oracle as discussed in Section 4.1, or simply waiting for longer ciphertexts.

Therefore, we can see that changing GCM into MAC-then-Enc paradigm would add little strength against these MAC forgery attacks.

7 Concluding Remarks

This paper revisits weak key classes of polynomial-based MAC schemes and provable security of GCM. We demonstrate that hash collisions are not necessary to construct successful MAC forgeries and any polynomials can be used in these attacks, which removes the restrictions in Procter and Cid's attacks. Based on these new discoveries on MAC forgeries, we prove that all subsets of keys with no less than two elements are weak key classes for GCM-like polynomial-based MAC schemes, which is an extension to Procter and Cid's analysis on weak keys. Moreover, we present a novel approach to transform these MAC forgery attacks into birthday attacks to increase their success probabilities. The success probabilities of these attacks are summarized in Table 1. Furthermore, we provide a method to fix GCM in order to avoid the security proof flaw discovered by Iwata *et al.* and significantly improve the security bounds. In addition, we show that these MAC forgeries attacks would still succeed if GCM is modified to MAC-then-Enc paradigm, as one of the options mentioned in [16,17], such that authentication tags are protected by Counter Mode encryptions.

Table 1. Comparisons of success probabilities of MAC forgery attacks

Method	Success Probability	Reference
Trial-and-Error	$n(l_N+1)/2^{128}$	[16,17]
Birthday Attack	$\leq 0.5 \cdot n^2(l_N+1)/2^{128}$	Section 4.1
Birthday Attack with inc	$\leq 2^{22}(n-1)(n+\sigma)(l_N+1)/2^{128}$	Section 4.1

Future work may include improving the probability analyses in Section 4.1. Certain probabilities for collisions and MAC forgeries are characterized by upper bounds rather than average estimations. If more accurate probabilities can be derived, this work may also, in return, improve the security bounds given by Iwata *et al.* on the original GCM design.

As recommended in [8,9], we further suggest that GCM may preferably be used with 96-bit nonces. For example, an altered version of GCM was introduced by Aoki and Yasuda in [1], which only accepts a fixed-length nonce. Reusing GHASH in both generating initial counter numbers and computing authentication tags may help attackers to amplify their success probabilities for MAC forgeries as we discussed in Section 4.1. For practical applications that have to use non-96-bit nonces, we suggest applying the fix to GCM proposed in Section 5.1, i.e. using LGCM defined in Algorithm 2, which could tighten the security bounds by a factor of around 2^{20}.

Acknowledgments. The authors would like to thank the anonymous reviewer for the helpful comments. This work is supported by NSERC Discovery Grant and ORF-RE Grant.

References

1. Aoki, K., Yasuda, K.: The security and performance of "GCM" when short multiplications are used instead. In: Kutyłowski, M., Yung, M. (eds.) Inscrypt 2012. LNCS, vol. 7763, pp. 225–245. Springer, Heidelberg (2013)
2. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 32–49. Springer, Heidelberg (2005)
3. CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness, `http://competitions.cr.yp.to/caesar.html`
4. Dworkin, M.J.: SP 800-38D. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Technical report, Gaithersburg, MD, United States (2007)
5. Ferguson, N.: Authentication weaknesses in GCM. Comments Submitted to NIST Modes of Operation Process (2005)
6. Handschuh, H., Preneel, B.: Key-recovery attacks on universal hash function based MAC algorithms. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 144–161. Springer, Heidelberg (2008)
7. IEEE 802.1AE. Media access control (MAC) security (2006), `http://www.ieee802.org/1/pages/802.1ae.html`
8. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 31–49. Springer, Heidelberg (2012)
9. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. Cryptology ePrint Archive, Report 2012/438 (2012), `http://eprint.iacr.org/`
10. Joux, A.: Authentication failures in NIST version of GCM. NIST Comment (2006)
11. Katz, J., Lindell, Y.: Introduction to modern cryptography. Chapman & Hall (2008)
12. McGrew, D., Viega, J.: The Galois/Counter Mode of operation (GCM). Submission to NIST Modes of Operation Process (2004)
13. McGrew, D.A., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)
14. McGrew, D.A.: Counter mode security: Analysis and recommendations (2002), `http://www.mindspring.com/~dmcgrew/ctr-security.pdf`
15. NSA. Suite B Cryptography (2005), `http://www.nsa.gov/ia/programs/suiteb_cryptography/`
16. Procter, G., Cid, C.: On weak keys and forgery attacks against polynomial-based MAC schemes. In: Fast Software Encryption.LNCS. Springer (to appear, 2013)
17. Procter, G., Cid, C.: On weak keys and forgery attacks against polynomial-based MAC schemes. Cryptology ePrint Archive, Report 2013/144 (2013), `http://eprint.iacr.org/`
18. Rogaway, P.: Authenticated-encryption with associated-data. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 98–107. ACM, New York (2002)
19. Saarinen, M.-J.O.: Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 216–225. Springer, Heidelberg (2012)

20. Shoup, V.: On fast and provably secure message authentication based on universal hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 313–328. Springer, Heidelberg (1996)
21. Viega, J., McGrew, D.A.: The use of Galois/Counter Mode (GCM) in IPsec encapsulating security payload, ESP (2005), http://tools.ietf.org/html/rfc4106.html

Appendix A Practical Attack Examples

A.1 A Example for Forgeries by Polynomials with Non-zero Constant Terms

This example is for GCM with AES-128 and 128-bit authentication tags, and the associated data A is always considered as empty. We use the same representations as the test vectors in GCM's specification [12], e.g. 1 in GF(2^{128}) is represented as 80000000000000000000000000000000, and longer strings will written in multiple lines.

We take the following values for the encryption of GCM. The lengths of P and C are 128 bits, i.e. one block.

K	71eebc49c8fb773b2224eaff3ad68714
N	07e961e67784011f72faafd95b0eb640
	89c8de15ad685ec57e63d56e679d3e20
	2b18b75fcbbec3185ffc41653bc2ac4a
	e6ae8be8c85636f353a9d19a86100d0b
P	705da82292143d2c949dc4ba014f6396
H	d27430c121f14d4ddfecb38acaffec53
C	251ccc6d2c45540cac4fde8b1e36802d
T	be2da05993fbde00421c1d8eaaaea373

Suppose we have a subset of authentication keys $\mathcal{H} = \{H_1, H_2, H_3\}$, whose values are as follows.

H_1	d27430c121f14d4ddfecb38acaffec53
H_2	00000000000000000000000000000001
H_3	00000000000000000000000000000002

Construct the polynomial

$$q(x) = \sum_{i=0}^{3} Q_i x^i = \prod_{i=1}^{3} (x \oplus H_i),$$

we can get the values for Q_i's.

Q_3	80000000000000000000000000000000
Q_2	d27430c121f14d4ddfecb38acaffec50
Q_1	c488aa211ab5dccec9c440bc33fc47b3
Q_0	5bb5716dc4b4687a06f15f10d62613ee

Please note $q(x)$ is a polynomial with non-zero constant term, i.e. $Q_0 \neq 0$.

Then compute $\alpha = (1 \oplus 2)/Q_0 =$ `7ef05dd871ead7e7f8e79d7d9343a170`, such that $\alpha \cdot Q_1 \oplus 1$ will match the length of new message, i.e. 2. Construct the new ciphertext $C' = (\alpha \cdot Q_3)||(C \oplus \alpha \cdot Q_2)$, and the authentication tag $T' = T \oplus \alpha \cdot Q_0$.

C'	`7ef05dd871ead7e7f8e79d7d9343a170`
	`7ccbd8dbfca54d785f5662d48c7eef81`
T'	`8b53b318750a2e948459b204e47629b4`

(N, C', T') passes the verification, and thus we complete a MAC forgery with length extension by using a polynomial with a non-zero constant term.

A.2 MAC Forgeries by Attacking Non-96-bit Nonces of GCM

We only give a basic example for this case. The values and the polynomial $q(x)$ computed in the previous example are reused here.

Construct the polynomial $q'(x) = x^2 q(x)$, and apply $q'(x)$ to N to get a new 512-bit nonce N', i.e. $N' = (N_4 \oplus Q_3)||(N_3 \oplus Q_2)||(N_2 \oplus Q_1)||(N_1 \oplus Q_0)$.

N'	`87e961e67784011f72faafd95b0eb640`
	`5bbceed48c991388a18f66e4ad62d270`
	`ef901d7ed10b1fd6963801d9083eebf9`
	`bd1bfa850ce25e8955588e8a50361ee5`

Feeding (N', P) to the encryption oracle will result in the same ciphertext as C, so we are sure that the authentication H is the set $\mathcal{H}$, and further MAC forgeries can be carried out by using $q(x)$.

A.3 MAC Forgeries for GCM in MAC-then-Enc Mode

The same K, N, H_1, and H_2 as in the previous examples are used. In order to avoid length extension, P is chosen to be longer and H_3 is explicitly chosen to be $H_1 \cdot H_2/(H_1 \oplus H_2)$.

P	`705da82292143d2c949dc4ba014f6396`
	`705da82292143d2c949dc4ba014f6396`
C	`a51ccc6d2c45540cac4fde8b1e36802d`
	`a4bd55da5dcde1d763021d44f5fb3ab8`
ET	`5aba7c39516a4a90f738eaf61b02514a`
H_3	`6e0b0d1eaf109b0f26926be82780085c`

Constructing the polynomial $q(x)$, we can have its coefficients as follows.

Q_3	`80000000000000000000000000000000`
Q_2	`bc7f3ddf8ee1d642f97ed862ed7fe40e`
Q_1	`00000000000000000000000000000000`
Q_0	`c52222258b2614c4c6f5981c65f15acd`

Please note $Q_1 = 0$, so the length padding block in GHASH can stay unchanged.

The new ciphertext and encrypted authentication tag are $C' = (C_2 \oplus Q_3)||(C_1 \oplus Q_2)$ and $ET' = ET \oplus Q_0$.

C'	a51ccc6d2c45540cac4fde8b1e36802d
	a4bd55da5dcde1d763021d44f5fb3ab8
ET'	5aba7c39516a4a90f738eaf61b02514a

(N, C', ET') passes the verification oracle of MtE GCM.