Universal Forgery Attack against GCM-RUP

Yanbin Li[1,2], Gaëtan Leurent[3], Meiqin Wang[1,2], Wei Wang[1,2], Guoyan Zhang[1,2], Yu Liu[1,2]

[1] School of Cyber Science and Technology, Shandong University, Jinan, China
[2] Key Laboratory of Cryptologic Technology and Information Security(Shandong University), Ministry of Education, Jinan, China
[3] Inria, Paris, France

Abstract. Authenticated encryption (AE) schemes are widely used to secure communications because they can guarantee both confidentiality and authenticity of a message. In addition to the standard AE security notion, some recent schemes offer extra robustness, i.e. they maintain security in some misuse scenarios. In particular, Ashur, Dunkelman and Luykx proposed a generic AE construction at CRYPTO'17 that is secure even when releasing unverified plaintext (the RUP setting), and a concrete instantiation, GCM-RUP. The designers proved that GCM-RUP is secure up to the birthday bound in the nonce-respecting model.

In this paper, we perform a birthday-bound universal forgery attack against GCM-RUP, matching the bound of the proof. While there are simple *distinguishing* attacks with birthday complexity on GCM-RUP, our attack is much stronger: we have a partial *key recovery* leading to universal forgeries. For reference, the best known universal forgery attack against GCM requires $2^{2n/3}$ operations, and many schemes do not have any known universal forgery attacks faster than 2^n. This suggests that GCM-RUP offers a different security trade-off than GCM: stronger protection in the RUP setting, but more fragile when the data complexity reaches the birthday bound. In order to avoid this attack, we suggest a minor modification of GCM-RUP that seems to offer better robustness at the birthday bound.

Keywords: GCM-RUP, partial key recovery, universal forgery, birthday bound.

1 Introduction

Authenticated encryption (AE) schemes aim to achieve both confidentiality and authentication of the encapsulated data. The first AE schemes were designed by combining a symmetric encryption scheme with a message authentication code (MAC). The encryption scheme provides confidentiality while the message authentication code ensures authenticity. Several generic composition schemes have been formalized and analyzed by Bellare and Namprempre [3]: Encrypt-and-MAC, MAC-then-Encrypt, and Encrypt-then-MAC. Their analysis considers black-box composition, without specific details of the underlying symmetric encryption scheme and MAC, in order to only focus on the security of the generic

composition at a high level. Their analysis shows that only the Encrypt-then-MAC composition is generically secure.

Later, new AE modes have been proposed [11,18,30] to provide confidentiality and authentication in a single scheme, which is more efficient than the generic composition of conventional mechanisms. AE schemes are now widely used in Internet protocols, and there is an ongoing effort to design and standardize new AE schemes with the recent CAESAR competition [35], and the NIST lightweight standardisation effort [38] currently running. The design and cryptanalysis of AE schemes is a very active topic in the cryptographic community today.

One of the most widely used AE schemes today is the Galois/Counter mode (GCM) [8,23], an AE scheme following the Encrypt-then-MAC paradigm. GCM has been widely deployed thanks to its excellent software performance and hardware support, and because there are no intellectual property restrictions to its use. It has been standardized in TLS [7], ISO/IEC [37], NSA Suite B [39] and IEEE 802.1 [36]. GCM encrypts data using a variation of the counter mode of operation (CTR) which requires a single block cipher encryption per message block, and does not need to perform block cipher decryption, even when decrypting the message. The ciphertext and associated data are authenticated with a Wegman-Carter-Shoup authenticator, where the keyed universal hash function is a polynomial evaluation over a binary Galois field. However, GCM is not robust against implementation errors or misuse. In particular, if a nonce is used just two times, the confidentiality and authentication for GCM are compromised with Joux's "forbidden attack" [17]. GCM also loses its security if a device releases the plaintext corresponding to invalid ciphertext before verifying the tag. Therefore, variants of GCM have been proposed to achieve some more robust security notions.

In 2015, Gueron *et al.* presented GCM-SIV [12] combining GCM's underlying components with the SIV paradigm designed by Rogaway and Shrimpton [31], to provide nonce-misuse resistance. Later, at CRYPTO'17, Ashur *et al.* introduced a generic construction of AE scheme using a tweakable block cipher (TBC), which resists attacks in the RUP setting [2] (with Release of Unverified Plaintext). Based on the generic AE scheme, an instantiation GCM-RUP with high-efficiency is put forward using AES-GCM's components. The designers proved that GCM-RUP is secure up to the birthday bound in the nonce-respecting model and RUP setting. On the other hand, no attacks are known so far against the authentication part of GCM-RUP. Therefore we do not know whether the proof is tight, and we do not know what kind of security degradation to expect after the birthday bound.

1.1 Contributions

In this paper we describe a universal forgery attack against GCM-RUP with time and data complexity close to $2^{n/2}$, where n denotes the block size of the underlying block cipher. This attack matches the security proof given in [2], showing that it is tight. However, our main result is not only about tightness of the (birthday) security bound, but rather about how badly the construction

of GCM-RUP breaks when the bound is reached: a universal forgery attack is much stronger than a distinguishing attack.

This is significant because no similar attack is known against GCM: on the one hand there are attacks with roughly $\sqrt{n} \times 2^{n/2}$ queries and time 2^n [20,22,26], and on the other hand attacks with $\sqrt{n} \times 2^{2n/3}$ queries and time $n \times 2^{2n/3}$ [20]. Our results show that universal forgery attacks against GCM-RUP are easier than against GCM, even though the security bounds from the proofs are similar, and both proofs are known to be tight (with simple distinguishing attacks).

Our attack is based on the following techniques:

- We show that inner collisions in the authentication part of GCM-RUP can be detected efficiently, and give out the output difference of the universal hash function GHASH_{K_2};
- Due to the structure of GHASH, we build a polynomial equation in K_2, which can be solved efficiently;
- Finally, when K_2 is known, we can sign arbitrary messages. This defines a universal forgery attack with complexity $2^{n/2}$ (time and data).

Since our attack points out a weakness in the structure of GCM-RUP, we also suggest a minor modification to GCM-RUP to prevent the leakage of the output of GHASH_{K_2} by using an extra block cipher call E_{K_4} to encrypt the output of GHASH_{K_2}. The objective of our variant is to achieve better security in the RUP setting and in the classical setting.

Many designs use GHASH because of its high performances. However, the output of GHASH may leak information about the key, as exploited in our attack. Therefore, the stronger GHASH variant we proposed could be applied to not only GCM like scheme but also future GHASH-based designs.

1.2 Related Works

Modes of operation are usually studied with security proofs, but there is a growing interest in generic attacks, showing how the security degrades when the proof doesn't hold. In particular, many attacks focus on (partial) key-recovery: most modes of operations have distinguishing attacks with birthday complexity $2^{n/2}$, but key-recovery and universal forgery attacks with the same complexity show that some schemes are more fragile than others when approaching the birthday bound.

For instance, in 1996, Preneel and Van Oorshot gave a full key recovery attack against the Envelope MAC with complexity $2^{n/2}$ [29]. In 2003, Mitchell studied several variants of CBC-MAC and compared their security against key-recovery attacks [25]; for some schemes the best attack reported requires an exhaustive search over an n-bit key, but attacks with birthday complexity can recover a partial key for TMAC and OMAC [33], leading to stronger forgery attacks. More recently, a series of works has shown birthday attacks against HMAC, with full key recovery when the hash function uses an internal checksum [19] and universal forgeries [27] in general. During the CAESAR competition, it was pointed out

that the security of AEZ [14] collapses at the birthday bound, with a full key recovery [10]. The scheme was modified to avoid the attack, but a variant is still applicable [6].

Besides MAC algorithms, there has also been work on message-recovery attacks on encryption modes, with a stronger impact than distinguishers. The well-known collision attack against CBC has been shown to be usable in practice with 64-bit block ciphers [4], and message-recovery attacks have also been shown against the CTR mode [20], even though the well-known distinguisher is much weaker.

All these results clearly show the importance of cryptanalysis work against modes of operation, even when the attacks do not contradict the proofs. In addition, this type of work sometimes detects mistakes in the proofs, as shown with GCM [16] and OCB2 [15].

1.3 Organization

The remainder of this paper is organized as follows. Section 2 gives the preliminaries. Section 3 briefly describes the generic construction and its instantiation GCM-RUP. We recover the authentication key in Section 4, and a universal forgery is provided in Section 5. Section 6 recommends a minor modification to GCM-RUP to resist our forgery attack. Finally, Section 7 concludes this paper.

2 Preliminaries

This section will show notations, operations, some cryptographic schemes and security definitions used in this paper.

2.1 Notations and Operations

- n: The block size of the block cipher (for GCM-RUP, $n = 128$).
- $\{0,1\}^{\leq x}$: The set of strings with length no greater than x bits.
- $\{0,1\}^*$: The set of strings with arbitrary length.
- $|X|$: Length of X, if $X \in \{0,1\}^*$.
- $X \oplus Y$: Bit-wise exclusive OR of X and Y, if $X, Y \in \{0,1\}^*$.
- $X \cdot Y$: Galois field multiplication of X and Y, if $X, Y \in \{0,1\}^n$.
- $X \| Y$ or XY: Concatenation of X and Y, if $X, Y \in \{0,1\}^*$.
- ε: The empty string.
- 0^n: n-bit string consisting of only zeros.
- $len_n(X)$: Length of X modulo 2^n as an n-bit string.
- $X0^{*n}$: X padded on the right with 0-bits to get a string of length a multiple of n.
- $|X|_n$: X's length in n-bit blocks $|X|_n = \lceil |X|/n \rceil$.
- $X[1]X[2]\ldots X[x] \xleftarrow{n} X$: Split X into substrings such that $|X[i]| = n$ for $i = 1, \ldots, x-1$, $0 < |X[x]| \leq n$, and $X[1] \| \ldots \| X[x] = X$.

- $int(Y)$: Map the j bits string $Y = a_{j-1} \dots a_1 a_0$ to the integer $i = a_{j-1}2^{j-1} + \dots + a_1 2 + a_0$.
- $str_j(i)$: Map the integer $i = a_{j-1}2^{j-1} + \dots + a_1 2 + a_0 < 2^j$ to the j-bit string $a_{j-1} \dots a_1 a_0$.
- $inc_m(X)$: The function which adds one modulo 2^m to X when viewed as an integer: $inc_m(X) := str_m(int(X) + 1 \ mod \ 2^m)$.
- $\mathrm{msb}_j(X)$: j most significant bits of X: $\mathrm{msb}_j(a_{i-1} \dots a_1 a_0) := a_{i-1} \dots a_{a-j}$.
- $\mathrm{lsb}_j(X)$: j least significant bits of X: $\mathrm{lsb}_j(a_{i-1} \dots a_1 a_0) := a_{j-1} \dots a_0$.
- $F \leftarrow E(C\|\cdot)$: Define $F(X) = E(C\|X)$ where C is fixed as constant.
- $a \stackrel{?}{=} b$: Evaluate to $\top$ if a equals b, and $\bot$ otherwise.

2.2 AE, Separated AE and TBC

An authenticated encryption scheme is a symmetric key algorithm that provides both confidentiality and authenticity. Bellare and Namprempre [3] defined the formal notion of authenticated encryption as follows:

Definition 1 (AE [3]). *An AE scheme consists of a pair of functions, the encryption function* Enc *and the decryption function* Dec*,*

$$\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C},$$
$$\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\bot\},$$

with $\mathcal{K}$ the key space, $\mathcal{N}$ the nonce space, $\mathcal{A}$ the associated data space, $\mathcal{M}$ the message space, $\mathcal{C}$ the ciphertext space, and $\bot$ an error symbol not contained in $\mathcal{M}$, which represents verification failure. It must be the case that for all $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{A}$ and $M \in \mathcal{M}$,

$$\mathsf{Dec}_K^N(A, \mathsf{Enc}_K^N(A, M)) = M.$$

The decryption process typically has two phases: plaintext computation and verification; the plaintext obtained from decryption is only given out after successful verification. However, keeping the full plaintext in memory can be an issue for constrained devices, and side-channel attacks can potentially recover information about the plaintext while it is decrypted. Therefore, new models have been introduced to take into account the effect of releasing unverified plaintext. In particular, Andreeva *et al.* [1] defined *separated* AE schemes where the plaintext computation is disconnected from verification; in this model the decryption function always releases the plaintext, without verifying it. Formally, a separated AE scheme is defined as:

Definition 2 (separated AE [1]). *A separated AE scheme consists of a triplet of functions, the encryption function* SEnc*, the decryption function* SDec*, and the verification function* SVer*, where*

$$\mathsf{SEnc} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C},$$
$$\mathsf{SDec} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M},$$
$$\mathsf{SVer} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \{\top, \bot\},$$

with $\mathcal{K}$ the key space, $\mathcal{N}$ the nonce space, $\mathcal{A}$ the associated data space, $\mathcal{M}$ the message space, $\mathcal{C}$ the ciphertext space. The special symbols $\top$ and $\bot$ indicate the success and failure of the verification, respectively. It must be the case that for all $K \in \mathcal{K}$, $N \in \mathcal{N}$, $A \in \mathcal{A}$ and $M \in \mathcal{M}$,

$$\mathsf{SDec}_K^N(A, \mathsf{SEnc}_K^N(A, M)) = M \quad \textit{and} \quad \mathsf{SVer}_K^N(A, \mathsf{SEnc}_K^N(A, M)) = \top.$$

Finally, we need to introduce the notion of tweakable block cipher (TBC), which is used in GCM-RUP. A tweakable block cipher is a generalization of a block cipher with an additional tweak input, generating a family of independent block ciphers [21]:

Definition 3 (TBC [21]). *A TBC could be regarded as a pair of functions* (E,D)*, with*

$$\mathsf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X},$$
$$\mathsf{D} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X},$$

where $\mathcal{K}$ is the key space, $\mathcal{T}$ is the tweak space, and $\mathcal{X}$ is the domain. For all $K \in \mathcal{K}$, $T \in \mathcal{T}$ and $X \in \mathcal{X}$, E_K^T is a permutation of $\mathcal{X}$ with D_K^T as inverse and

$$\mathsf{D}_K^T(\mathsf{E}_K^T(X)) = X.$$

3 Brief Description of GCM-RUP [2]

Ashur, Dunkelman and Luykx proposed a generic construction of an efficient separated AE scheme at CRYPTO'17 [2]. Their construction uses an encryption scheme and a TBC, and achieves security in the RUP setting, assuming that the encryption scheme is strongly indistinguishable-from-random-bits (SRND) [13, 32], and the TBC is a strong pseudorandom permutation (SPRP) [32]. Based on the generic construction, a dedicated instantiation GCM-RUP is built using AES-GCM's primitives. This section will describe this construction and GCM-RUP.

3.1 Generic Construction with RUP Security [2]

Let $(\mathsf{Enc},\mathsf{Dec})$ be an encryption scheme (without authentication), with $\mathcal{K}$ the key space, $\mathcal{N}$ the nonce space, $\mathcal{M}$ the message space, and $\mathcal{C}$ the ciphertext space. Let (E,D) denote a TBC with key space $\mathcal{L}$, tweak space $\mathcal{T} = \mathcal{C}$, and domain $\mathcal{X} = \mathcal{N}$. Then define the separated AE scheme $(\mathsf{SEnc},\mathsf{SDec},\mathsf{SVer})$ as follows,

$$\mathsf{SEnc}_{K,L}^N(A, M) := \big(S = \mathsf{E}_L^{A,C}(N), C = \mathsf{Enc}_K^N(\alpha \| M)\big),$$
$$\mathsf{SDec}_{K,L}(A, S, C) := \mathrm{lsb}_{|C|-\tau}(\mathsf{Dec}_K^{\mathsf{D}_L^{A,C}(S)}(C)),$$
$$\mathsf{SVer}_{K,L}(A, S, C) := \mathrm{msb}_\tau(\mathsf{Dec}_K^{\mathsf{D}_L^{A,C}(S)}(C)) \stackrel{?}{=} \alpha,$$

where $(K, L) \in \mathcal{K} \times \mathcal{L}$ is the key, $\mathcal{N}$ is the nonce space, $\mathcal{A}$ is the associated data space, $\mathcal{M}$ is the message space, $\mathcal{N} \times \mathcal{C}$ is the ciphertext space, and $\alpha \in \{0,1\}^\tau$ is some pre-defined constant. The construction is depicted in Fig. 1. The procedures of encryption, decryption and verification are illustrated in Fig. 1 (a), (b) and (c), respectively.

The novelty of the generic construction is that the nonce is encrypted using the ciphertext as a tweak. This provides security in the RUP setting, because if an attacker modifies the ciphertext or the encrypted nonce, the decryption oracle will output a random plaintext. The authentication security comes from the redundancy in the plaintext, with the pre-defined constant α (known by both sides); the length of α determines the security level. In order to maintain security up to the birthday bound on the block size, the size of α and the nonce size are fixed to be the same as the block size n.

3.2 GCM-RUP [2]

GCM-RUP is an instantiation of the generic construction using the counter mode (CTR) for encryption and the XTX construction [24] with GHASH for the tweakable block cipher. It reuses the component of GCM in order to benefit from the efficient implementations available, while offering more robustness with security in the RUP setting. Before describing GCM-RUP itself, we first define the primitives borrowed from GCM. Let n denote the block length of the available block cipher, in this case $n = 128$.

The first one is the universal hash function GHASH, which takes a key H and two strings M and M' as input (in GCM, GHASH is used in the Wegman-Carter construction to build a MAC [34]). The core of GHASH is defined with a single string M constituted of full blocks, and evaluates a polynomial defined from M at H as follows,

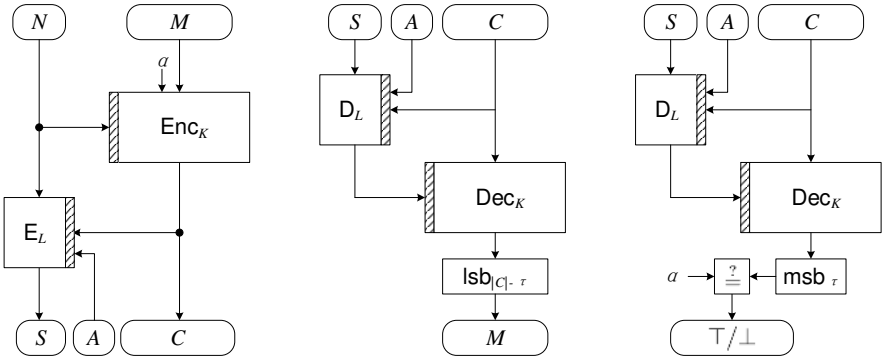
$$\mathsf{GHASHcore}_H(M) = \bigoplus_{i=0}^{|M|_n - 1} M[i] \cdot H^{|M|_n - i}. \tag{1}$$

The symbol "$\cdot$" represents multiplication in the Galois field $GF(2^n)$. All the computations are performed by the rule of operations defined in finite field. GHASH is defined from GHASHcore; it takes two strings M and M' as input, zero-pads and concatenates them, and adds the binary representation of the lengths of M and M' before processing the result through GHASHcore,

$$\mathsf{GHASH}_H(M, M') = \mathsf{GHASHcore}_H(M0^{*n} \| M'0^{*n} \| str_{n/2}(|M|) \| str_{n/2}(|M'|)),$$

where the function $str_j(i)$ maps the integer $i = a_{j-1}2^{j-1} + \cdots + a_1 2 + a_0 < 2^j$ to the j-bit string $a_{j-1} \ldots a_1 a_0$. Algorithm 1 describes the procedure of the function.

The second important auxiliary function is the CTR mode. Given a counter value X, a positive integer m and a predefined keyed function F as input, this function $\mathsf{CTR}[F](X, m)$ outputs a string S with m blocks. Each block of S is



(a) Encryption (b) Decryption (c) Verification

Fig. 1. Generic Construction with RUP Security

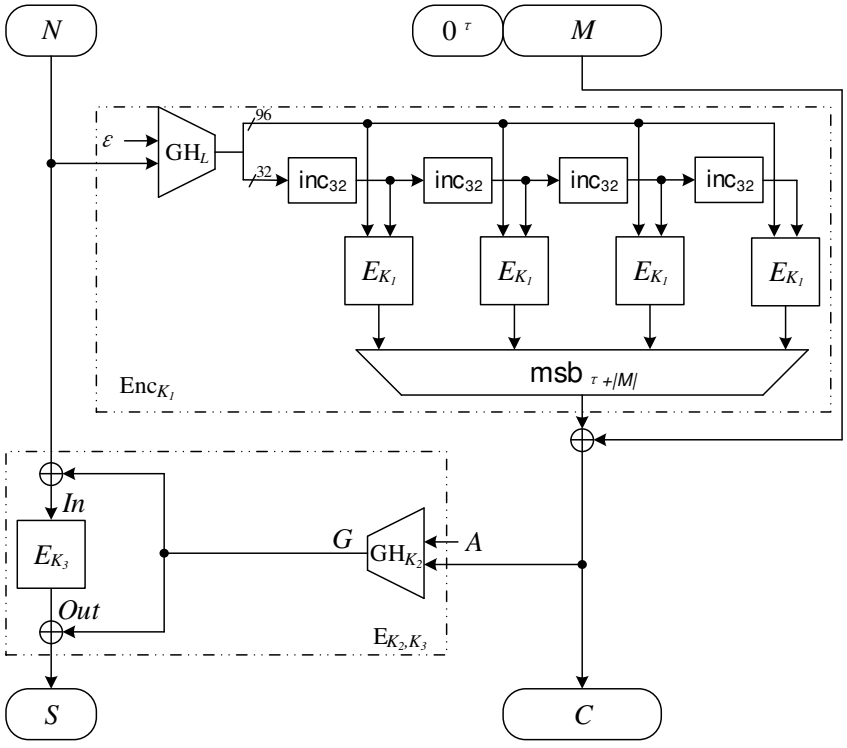


Fig. 2. GCM-RUP (Figure from [2])

Algorithm 1 $\mathsf{GHASH}_H(M, M')$

Input: $H \in \{0,1\}^n$, $M \in \{0,1\}^{\leq n(2^{n/2}-1)}$, $M' \in \{0,1\}^{\leq n(2^{n/2}-1)}$
Output: $Y \in \{0,1\}^n$
1: $X \leftarrow M0^{*n} \| M'0^{*n} \| str_{n/2}(|M|) \| str_{n/2}(|M'|)$
2: $X[1]X[2] \dots X[x] \xleftarrow{n} X$
3: $Y \leftarrow 0^n$
4: **for** $1 \leq j \leq x$ **do**
5: $\quad Y \leftarrow H \cdot (Y \oplus X[j])$
6: **end for**
7: **return** Y

computed by $S[i] = F(inc_x^i(X))$, where inc_x represents counter incrementation, adding one modulo 2^x to X, with the convention that inc_x^i represents i successive implementations. The CTR mode is defined in Algorithm 2.

Algorithm 2 $\mathsf{CTR}[F](X, m)$

Input: $F : \{0,1\}^x \to \{0,1\}^n$, $X \in \{0,1\}^x$, $m \in \mathbb{N}$
Output: $S \in \{0,1\}^{mn}$
1: $I \leftarrow X$
2: **for** $1 \leq j \leq m$ **do**
3: $\quad S[j] \leftarrow F(I)$
4: $\quad I \leftarrow inc_x(I)$
5: **end for**
6: $S \leftarrow S[1]S[2] \dots S[m]$
7: **return** S

Finally, GCM-RUP uses three keys: K_1 is used for the CTR encryption, and K_2 and K_3 are used for the TBC following the XTX construction (K_2 is used for GHASH, and K_3 is used for the underlying block cipher call). GCM-RUP encrypts a message M together with its associated data A and a nonce N, into a ciphertext C and an encrypted nonce S. The associated data, the message and the ciphertext are all seen as sequences of blocks of length n. GCM-RUP follows the generic construction given above, and is described in Fig. 2, with pseudocode in Algorithm 3 (with ε an empty string). In the figure, Enc_{K_1} corresponds to CTR mode encryption, and E_{K_2,K_3} to the TBC.

As an instantiation of the generic construction with RUP security, GCM-RUP is secure under RUP setting. More precisely, GCM-RUP can provide security up to the birthday bound on the block size (because this is the security of the underlying AE scheme and TBC).

Algorithm 3 $\mathsf{GCM\text{-}RUP}_{K_1,K_2,K_3}(N, A, M)$

Input: $K_1K_2K_3 \in \{0,1\}^{3n}$, $A \in \{0,1\}^{n2^{32}}$, $M \in \{0,1\}^{n2^{32}}$
Output: $(S, C) \in \{0,1\}^n \times \{0,1\}^{\tau+|M|}$
1: $M \leftarrow 0^\tau \| M$
2: $L \leftarrow E_{K_1}(0^n)$
3: $I \leftarrow \mathsf{GHASH}_L(\varepsilon, N)$
4: $m \leftarrow |M|_n$
5: $F \leftarrow E_{K_1}(\mathrm{msb}_{96}(I)\|\cdot)$
6: $S \leftarrow \mathsf{CTR}[F](inc_{32}(\mathrm{lsb}_{32}(I)), m)$
7: $C \leftarrow M \oplus \mathrm{msb}_{|M|}(S)$
8: $G \leftarrow \mathsf{GHASH}_{K_2}(A, C)$
9: $S \leftarrow E_{K_3}(N \oplus G) \oplus G$
10: **return** (S, C)

4 Partial Authentication Key Recovery for GCM-RUP

Our analysis focuses on the GHASH_{K_2} function, which can be written as a polynomial in K_2. In this section, we analyze properties of GHASH_{K_2} which are then used to recover K_2. After recovering K_2, it is possible to perform a forgery attack for GCM-RUP.

The main property used in our attacks is that G, the output of GHASH_{K_2} as defined in Fig. 2, is linearly dependent on the input (A, C) for fixed K_2. Therefore, the output difference ΔG of values G emerging in encryption operations of two input tuples (N_1, A, M) and (N_2, A, M) is independent of the value of (A, M), and is only a function of N_1 and N_2.

Based on this property, we retrieve K_2 with the following two steps.

- For a fixed associated data and message, we search for a pair of nonces (N_1, N_2) which produce a collision for the input of E_{K_3} using a birthday attack. For such pair of nonces (N_1, N_2), $\Delta G = N_1 \oplus N_2 = S_1 \oplus S_2$.
- With a known ΔG, a polynomial equation in K_2 is derived from the GHASH_{K_2} definition. Then K_2 can be retrieved by solving this equation.

In this section, we will give the detailed description of the recovery of K_2.

4.1 Properties of GHASH

Let $\Pi = (\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$ denote the scheme GCM-RUP. We focus on the component GHASH_{K_2} with inputs the associated data A and the ciphertext C. In order to clearly describe the attack, we rewrite GHASH_{K_2} as

$$G = \mathsf{GHASH}_{K_2}(A, C) = \mathsf{GHASHcore}_{K_2}(A\|C\|str_{n/2}(|A|)\|str_{n/2}(|C|)).$$

According to the definition of $\mathsf{GHASHcore}$ given by Equation (1), G is linear in the $\mathsf{GHASHcore}$ input $(A\|C\|str_{n/2}(|A|)\|str_{n/2}(|C|))$ for a fixed K_2. Therefore, we consider the difference ΔG in the output of GHASH_{K_2} for a pair of inputs.

Property 1 *When processing a fixed associated data A and message M under two distinct nonces (N_1, N_2) with* GCM-RUP*, the output difference ΔG of GHASH_{K_2} is only dependent on N_1 and N_2, but independent on A and M. This also holds for the input difference of E_{K_3}.*

Proof. For two tuples (N_1, A, M) and (N_2, A, M), query SEnc and get

$$(S_1, C_1) \leftarrow \mathsf{GCM\text{-}RUP}(N_1, A, M),$$
$$(S_2, C_2) \leftarrow \mathsf{GCM\text{-}RUP}(N_2, A, M).$$

Let G_1 and G_2 represent the corresponding outputs of the function GHASH_{K_2} in the encryptions under nonces N_1 and N_2, respectively,

$$G_1 = \mathsf{GHASH}_{K_2}(A, C_1),$$
$$G_2 = \mathsf{GHASH}_{K_2}(A, C_2),$$

where

$$C_1 = (0^\tau \| M) \oplus \mathsf{Enc}_{K_1}(N_1),$$
$$C_2 = (0^\tau \| M) \oplus \mathsf{Enc}_{K_1}(N_2),$$

the function Enc_{K_1} is defined in the upper dotted box in Fig. 2. Hence,

$$\begin{aligned} \Delta G &= G_1 \oplus G_2 \\ &= \mathsf{GHASH}_{K_2}(A, C_1) \oplus \mathsf{GHASH}_{K_2}(A, C_2). \end{aligned} \tag{2}$$

From the definition of GHASH, we have

$$\begin{aligned} \Delta G &= \mathsf{GHASHcore}_{K_2}\big(A \oplus A \| C_1 \oplus C_2 \| \\ &\qquad (str_{n/2}(|A|) \| str_{n/2}(|C_1|)) \oplus (str_{n/2}(|A|) \| str_{n/2}(|C_2|))\big) \\ &= \mathsf{GHASHcore}_{K_2}(0^{|A|} \| \Delta C \| 0^n) \\ &= \mathsf{GHASHcore}_{K_2}(0^{|A|} \| \mathsf{Enc}_{K_1}(N_1) \oplus \mathsf{Enc}_{K_1}(N_2) \| 0^n), \end{aligned} \tag{3}$$

which shows that the output difference of the function GHASH_{K_2} depend only on N_1 and N_2 for two tuples (N_1, A, M) and (N_2, A, M). The input difference of E_{K_3} can be computed as

$$\begin{aligned} \Delta In &= N_1 \oplus N_2 \oplus \Delta G \\ &= N_1 \oplus N_2 \oplus \mathsf{GHASHcore}_{K_2}(0^{|A|} \| \Delta C \| 0^n), \end{aligned} \tag{4}$$

so it is also independent of A and M. □

In particular, if we can recover a value ΔG, we can then extract K_2 by solving a polynomial equation, given the ciphertext difference ΔC and the output difference ΔG:

$$\Delta G = \mathsf{GHASHcore}_{K_2}(0^{|A|} \| \Delta C \| 0^n).$$

For simplicity, we assume that $|M| = n$ and $\tau = n$, this implies $|C_1| = |C_2| = 2n$:

$$\Delta G = \Delta C[0] \cdot K_2^3 \oplus \Delta C[1] \cdot K_2^2.$$

This a polynomial equation in K_2 in the Galois field with 2^{128} elements. Luckily, there are efficient algorithm to factor polynomials over finite fields. For instance, the Cantor-Zassenhaus algorithm [5] requires $\mathcal{O}(n^2(\log(r)\log(q)+n))$ field operations to factor a degree-n polynomial with r irreducible factors over a field with q elements. In practice, with the parameters used here, this takes negligible time using the implementation of SageMath [40].

4.2 Recovering K_2 from Inner Collisions

As explained earlier, the first step of the attack is to identify collisions in the input of E_{K_3}, defined as $In = N \oplus G$. Following the analysis above, we start with a fixed associated data A and message M, and query SEnc for q different nonces, to receive the corresponding encrypted nonces S and ciphertexts C.

In order to simplify the description, we focus on the value In, and we consider the function mapping N, A, M to In, denoted as PEnc, and represented by Fig. 3. The output values of PEnc can not be accessed by the attacker, but collisions in PEnc can be detected. As for In and the output Out of E_{K_3},

$$In = N \oplus G,$$
$$Out = S \oplus G.$$

When the collisions happen, $\Delta In = \Delta Out = 0$, which means

$$N_1 \oplus N_2 \oplus G_1 \oplus G_2 = 0,$$
$$S_1 \oplus S_2 \oplus G_1 \oplus G_2 = 0.$$

Thus, $N_1 \oplus N_2 = S_1 \oplus S_2 = \Delta G$. If the collisions $\Delta N = \Delta S$ can be detected, the collisions in PEnc can be detected. Meanwhile, this type of collisions give out the value of ΔG, which can be used to recover K_2. Moreover, the corresponding pairs can be identified efficiently. We just build a list of all nonces indexed by $N_i \oplus S_i$, sort the list and look for collisions: each collision corresponds to a pair with $N_1 \oplus S_1 = N_2 \oplus S_2$ i.e. $N_1 \oplus N_2 = S_1 \oplus S_2$. We now consider the converse, and evaluate the probability of a collision in PEnc when $N_1 \oplus N_2 = S_1 \oplus S_2$.

We formally define the two events as X and Y:

- X ($N_1 \oplus N_2 = S_1 \oplus S_2$): the event identifying pairs of nonces (N_1, N_2) with the input difference equal to the output difference of E_{K_3}, which is called *outer collision* (equivalently, it can be defined as $\Delta In = \Delta Out$).
- Y ($\Delta In = 0$): the event identifying pairs of nonces with collision in PEnc, i.e. zero input difference for E_{K_3}, which is called *inner collision*.

First, we observe that $Y \subseteq X$, because if $\Delta In = 0$, then $\Delta Out = 0$ and $N_1 \oplus N_2 = S_1 \oplus S_2 = \Delta G$. Therefore, we have

$$\Pr[Y|X] = \frac{\Pr[Y]}{\Pr[X]}.$$

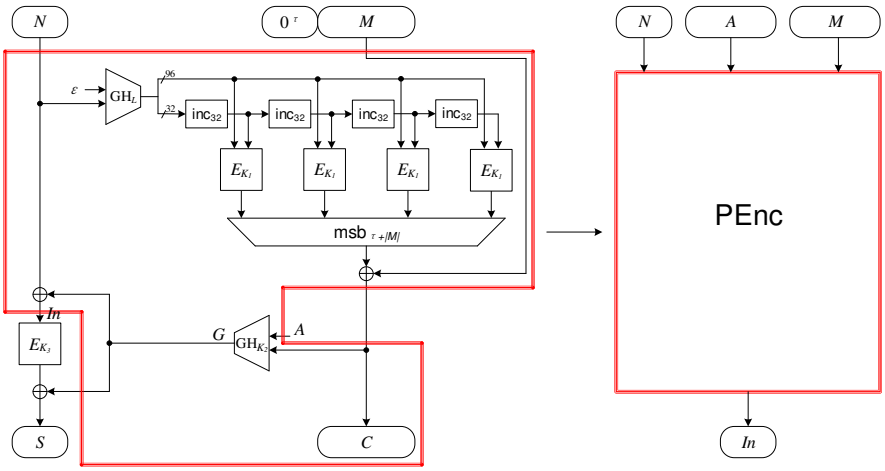


Fig. 3. Representation of the Function PEnc

Moreover, we have $\Pr[Y] = 2^{-n}$ because the output of PEnc with a fresh nonce is random, assuming that E is a PRF. In order to compute $\Pr[X]$, we consider two cases, depending on event Y:

1. $\Delta In = 0$**.** Then we have necessarily $\Delta Out = 0$, i.e. $Pr[X|\Delta In = 0] = 1$.
2. $\Delta In \neq 0$**.** A pair with non-zero input difference must produce a non-zero output difference. Assuming that E is a PRF, we have $Pr[X|\Delta In \neq 0] = \frac{1}{2^n-1}$.

Therefore,

$$\begin{aligned} Pr(X) =& Pr[\Delta In = \Delta Out] \\ =& Pr[\Delta In = \Delta Out|\Delta In = 0] \times Pr[\Delta In = 0] + \\ & Pr[\Delta In = \Delta Out|\Delta In \neq 0] \times Pr[\Delta In \neq 0] \\ =& 1 \times \frac{1}{2^n} + \frac{1}{2^n - 1} \times \frac{2^n - 1}{2^n} \\ =& \frac{1}{2^{n-1}}. \end{aligned} \tag{5}$$

Finally, we can conclude

$$\Pr[\Delta In = 0|N_1 \oplus N_2 = S_1 \oplus S_2] = \frac{2^{-n}}{2^{-n+1}} = \frac{1}{2}. \tag{6}$$

Attack Procedure. We can now give the detailed procedure to recover K_2:

1. Choose an arbitrary associated data A and a single-block message M, then query SEnc for q different nonces N and receive the corresponding encrypted

nonces S and ciphertexts C; save them in a table indexed by $N \oplus S$. With a suitable value of q (in the order of $2^{n/2}$), there are two pairs of nonces (N_1, N_2) satisfying $N_1 \oplus N_2 = S_1 \oplus S_2$, one of which is expected to further satisfy $\Delta In = 0$.
2. For each pair with $N_1 \oplus N_2 = S_1 \oplus S_2$, assuming that $\Delta In = 0$, we have $\Delta G = \Delta S$ and we obtain a cubic polynomial equation with unknown variable K_2, which can be solved with factoring tools:

$$\Delta S = \mathsf{GHASHcore}_{K_2}(0^{|A|} \| \Delta C \| 0^n) = \Delta C[0] \cdot K_2^3 \oplus \Delta C[1] \cdot K_2^2.$$

3. Identify the correct candidate for K_2 with forgery attempts.

Using two pairs of nonces, this attack suggests a small set of six key candidates. The correct key can be identified with forgery attempts, or by using more pairs and looking for a repeated key candidate.

More precisely, we will describe how to construct a forgery with known candidate of K_2 for GCM-RUP in Section 5 for a given message, which can be used to filter the correct K_2. There would be two cases:

- If the forgery is constructed under the correct candidate for K_2, it can pass the verification algorithm of GCM-RUP.
- If the forgery is constructed under the wrong candidate for K_2, it will receive a failure of the verification of GCM-RUP.

We only need to query the verification oracle SVer six times to identify the correct K_2. The cost for this step is negligible.

Complexity Estimation. As already mentioned, the probability of two random nonces N_1 and N_2 satisfying $\Delta In = 0$ is 2^{-128}. Starting from a set of q queries, we can evaluate the probability p of finding an inner collision following the analysis of the birthday paradox:

$$p \simeq 1 - e^{-q^2/(2 \times 2^{128})}.$$

Thus,

$$q \simeq \sqrt{2 \times 2^{128} \ln \frac{1}{1-p}}.$$

Table 1 shows number of nonces needed to achieve the given probability of success.

4.3 Experimental Verification with Mini-GCM-RUP

In order to verify our attack theory, we use a mini version of GCM-RUP constructed with the 16-bit block cipher 4-round Mini-AES [28] to experimentally recover K_2. This experiment identifies pairs of nonces in event X and Y from 2^9 random nonces, and recover K_2 with SageMath. We execute this experiment several times to give some results to show the validity of probabilities of event X and Y in our paper, the detail is listed in Table 2.

In this table, we see that probabilities of event X and Y conform to Equation (5) and (6), respectively. The complexity of this experiment is dominated by 2^9.

Table 1. Number of Nonces Needed to Achieve the Given Success Probability

Number of nonces to identify inner collision	Probability of finding inner collision
2^{63}	11%
2^{64}	39%
2^{65}	86%
2^{66}	99.9%

Table 2. Experimental Verification with Mini-GCM-RUP

(K_1, K_2, K_3)	Pair of nonces in X	ΔIn	$Pr(X)$	$Pr(Y\|X)$
(0x3d0e,0x2afc,0x2e91)	(0x2704,0x0889)	0	$\frac{1}{2^8}$	1
	(0x7649,0x7b0d)	0		
(0x4ef3,0x454b,0x1e9a)	(0x2323,0x602d)	0	$\frac{7}{2^9}$	$\frac{3}{7}$
	(0x11b7,0x2b2e)	0x0af7		
	(0x7bab,0x3a72)	0		
	(0x1215,0x1e05)	0xa3b5		
	(0x6593,0x093d)	0xbce8		
	(0x09bd,0x2db2)	0x03cf		
	(0x7d35,0x5e97)	0		
(0x5388,0x2641,0x7a4f)	(0x0ba9,0x46f5)	0x5393	$\frac{3}{2^8}$	$\frac{1}{2}$
	(0x684d,0x5786)	0		
	(0x334c,0x22e1)	0x0636		
	(0x4487,0x13f0)	0		
	(0x5413,0x03d8)	0		
	(0x5a91,0x179f)	0x0c06		
(0x5691,0x2ee9,0x5a68)	(0x3874,0x7546)	0x3fcb	$\frac{1}{2^8}$	$\frac{1}{2}$
	(0x44b0,0x4323)	0		

5 Universal Forgery Attack of GCM-RUP

In this section, we will construct forgeries for GCM-RUP given a candidate for K_2. We consider a challenge message M^* (and possibly a challenge associated data A^*), and our goal is to construct a valid ciphertext for M^*.

5.1 Almost Universal Forgery Attack

The first forgery attack makes only one query to the encryption oracle SEnc and then constructs a forgery by solving an equation over $GF(2^{128})$.

For an arbitrary nonce N, associated data A and message M (with $|M| = |M^*|$), query (N, A, M), and receive the corresponding ciphertext (S, C). Let $G = \mathsf{GHASH}_{K_2}(A, C)$, and the keystream used to XOR message is computed by

$$\mathsf{Enc}_{K_1}(N) = C \oplus (0^\tau \| M).$$

We create a valid encryption of M^* by reusing the same nonce N and the values G and S.

First, we compute C^* corresponding to M^*:

$$\begin{aligned} C^* &= 0^\tau \| M^* \oplus \mathsf{Enc}_{K_1}(N) \\ &= 0^\tau \| M^* \oplus (C \oplus 0^\tau \| M). \end{aligned} \tag{7}$$

Then we construct A' such that

$$\mathsf{GHASH}_{K_2}(A', C^*) = \mathsf{GHASH}_{K_2}(A, C),$$

where A, C, C^* and K_2 are known. This gives a linear equation over $GF(2^{128})$ which can easily be solved assuming that $|A'| \geq 128$ and $K_2 \neq 0$.

To summarize, for any chosen message M^*, we can give a successful forgery (A', M^*, S', C') satisfying $(S' = S, C' = 0^\tau \| M^* \oplus (C \oplus 0^\tau \| M))$. This is an almost universal forgery, because we can choose M^* freely but not A'.

5.2 Universal Forgery Attack

Alternatively, we can design an attack where we choose both A^* and M^*, using $2^{n/2}$ queries. First, we make $2^{n/2}$ queries (N_i, A, M), for fixed A and M with $|M| = |M^*|$, and receive the corresponding (S_i, C_i). Since K_2 is known, we can compute $G_i = \mathsf{GHASH}(A, C_i)$, and recover the corresponding inputs and outputs to E_{K_3}: $E_{K_3}(N_i \oplus G_i) = S_i \oplus G_i$.

Then, we can use the same nonces N_i to build a forgery. For each N_i, we build the corresponding C_i' from M^* and C_i as above, and we check whether $N_i \oplus \mathsf{GHASH}(A^*, C_i')$ is in the set of known inputs to E_{K_3}. With high probability, one of the nonces will result in a match $N_i \oplus \mathsf{GHASH}(A^*, C_i') = N_j \oplus G_j$, and we deduce a forgery using $S' = S_j \oplus G_j \oplus \mathsf{GHASH}(A^*, C_i')$.

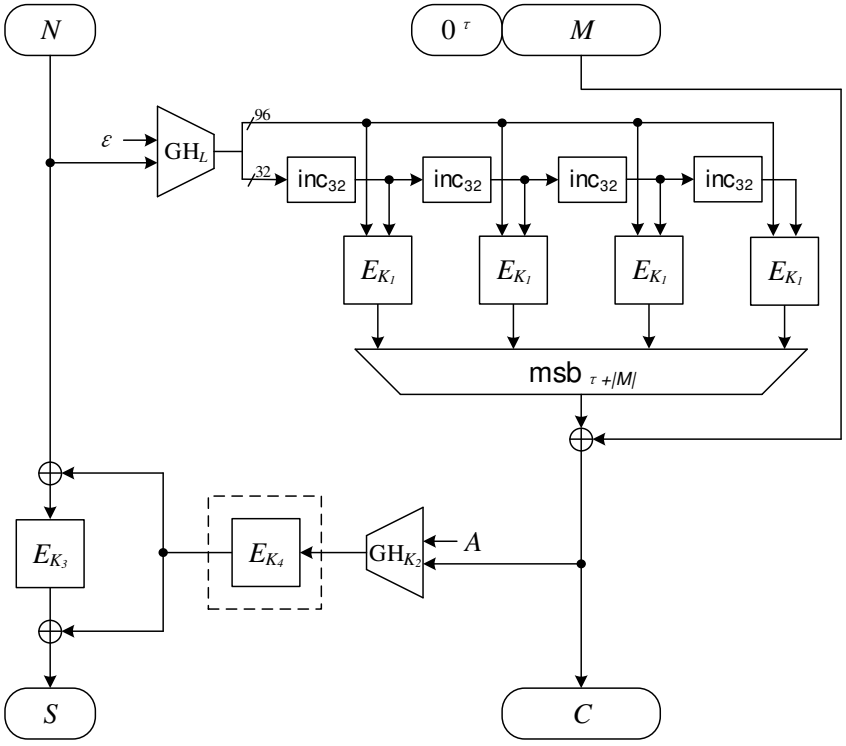


Fig. 4. A Variant for GCM-RUP

6 Variant of GCM-RUP

Our forgery attack against GCM-RUP highlights a potential weakness on the structure of GCM-RUP: the output difference of the function GHASH_{K_2} can be recovered with birthday complexity and this leads to a recovery of K_2. In order to prevent this attack, we suggest to add a block cipher call in the TBC construction used in GCM-RUP, as shown in Fig. 4, to avoid leakage of the output difference of the function GHASH_{K_2}.

This modified TBC still follows the XTX construction of Iwata and Minematsu [24], using universal hash function $E_{K_4}(\mathsf{GHASH}_{K_2}(A, C))$ instead of the original $\mathsf{GHASH}_{K_2}(A, C)$. The new universal hash function has the same security bounds, but does not leak the key from an output difference. Thus, the security proof of GCM-RUP is still applicable to this variant. But we do not provide a formal security proof. The extra block cipher has a limited impact on efficiency, and might offer better security by avoiding our attack.

More generally, the modified GHASH could replace GHASH in other designs. In particular, the corresponding modification of GCM would prevent the universal forgery attack with complexity $2^{2n/3}$ given in [20]. We believe that this construction is worth further study. Further work will be needed to determine whether this modification actually provides extra security and how much.

7 Conclusion

This paper shows a birthday-bound attack against GCM-RUP [2] using inner collisions to recover the output difference of the function GHASH_{K_2}. Hence, K_2 can be retrieved by solving a polynomial equation, and this directly leads to a universal forgery attack against GCM-RUP. This forgery attack shows that the construction of GCM-RUP breaks drastically when the security bound is reached. This is surprising because no such attack is known on GCM: the best known universal forgery attack requires $2^{2n/3}$ operations.

Finally, a minor modification of GCM-RUP is suggested to prevent this kind of attack, using an additional block cipher to protect the output of GHASH. With little performance loss, this design focusing on GHASH can be applied to all GHASH-based designs.

In a more general setting, our attack technique can be applied to the LRW construction [21] with a polynomial universal hash function, as used in OCB, for instance. Actually, the corresponding attack on OCB would match the previous attack by Ferguson [9].

8 Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61572293 and 61602276, National Cryptography Development Foundation of China under Grant No. MMJJ20170102, Major Scientific and Technological Innovation Projects of Shandong Province, China under Grant

No. 2017CXGC0704, Natural Science Foundation of Shandong Province, China under Grant No. ZR2016FM22.

References

1. Andreeva E., Bogdanov A., Luykx A., Mennink B., Mouha N., Yasuda K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: Sarkar P., Iwata T. (Eds.) ASIACRYPT 2014, PART I, LNCS 8873, pp. 105–125. Springer.
2. Ashur T., Dunkelman O., Luykx A.: Boosting Authenticated Encryption Robustness with Minimal Modifications. In: Katz J., Shacham H. (Eds.) CRYPTO 2017, Part III, LNCS 10403, pp. 3-33. Springer.
3. Bellare M., Namprempre C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto T. (Eds.) ASIACRYPT 2000, LNCS 1976, pp. 531–545. Springer.
4. Bhargavan, K., Leurent, G.: On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 456–467. ACM Press (Oct 2016)
5. Cantor, D.G., Zassenhaus, H.: A new algorithm for factoring polynomials over finite fields. Mathematics of Computation pp. 587–592 (1981)
6. Chaigneau, C., Gilbert, H.: Is AEZ v4.1 sufficiently resilient against key-recovery attacks? IACR Trans. Symm. Cryptol. 2016(1), 114–133 (2016), `http://tosc.iacr.org/index.php/ToSC/article/view/538`
7. Dierks, T., Allen, C.: RFC 2246 - The TLS Protocol Version 1.0. Internet Activities Board (Jan 1999)
8. Dworkin M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology. SP 800-38D, November 2007.
9. Ferguson, N.: Collision attacks on OCB. Comment to NIST (Feb 2002).
10. Fuhr, T., Leurent, G., Suder, V.: Collision attacks against CAESAR candidates - forgery and key-recovery against AEZ and Marble. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 510–532. Springer, Heidelberg (Nov / Dec 2015)
11. Gligor V., Donescu P.: Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In: Matsui M. (Eds.) FSE 2001, LNCS 2355, pp. 92-108. Springer.
12. Gueron S., Lindell Y.: GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In: Ray I., Li N., Kruegel C. (Eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015, pp. 109-119. ACM (2015).
13. Halevi S., Rogaway P.: A Parallelizable Enciphering Mode. In: Okamoto T. (Eds.) CT-RSA 2004. LNCS 2964, pp. 292–304. Springer.
14. Hoang V.T., Krovetz T., Rogaway P.: Robust Authenticated-Encryption AEZ and the Problem That It Solves. In: Oswald E., Fischlin M. (eds) EUROCRYPT 2015. LNCS, vol 9056, pp 15-44. Springer, Berlin, Heidelberg
15. Inoue, A., Iwata, T., Minematsu, K., Poettering, B.: Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. In: Boldyreva A., Micciancio D. (eds) CRYPTO 2019. LNCS, vol 11692. Springer, Cham

16. 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 (Aug 2012)
17. Joux A.: Comments on the Draft GCM Specification - Authentication Failures in NIST Version of GCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38Series-Drafts/GCM/Jouxcomments.pdf.
18. Jutla C.: Encryption Modes with Almost Free Message Integrity. In: Pfitzmann B. (Eds.) EUROCRYPT 2001, LNCS 2045, pp. 529-544. Springer.
19. Leurent, G., Peyrin, T., Wang, L.: New generic attacks against hash-based MACs. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 1–20. Springer, Heidelberg (Dec 2013)
20. Leurent, G., Sibleyras, F.: The missing difference problem, and its applications to counter mode encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 745–770. Springer, Heidelberg (Apr / May 2018)
21. Liskov M., Rivest R.L., Wagner D.: Tweakable Block Ciphers. In: Yung M. (Eds.) CRYPTO 2002, LNCS 2442, pp. 31-46. Springer.
22. Luykx A., Preneel B.: Optimal forgeries against polynomial-based MACs and GCM. In: Nielsen J.B., Rijmen V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 445–467. Springer, Heidelberg (Apr / May 2018)
23. McGrew D., 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, Berlin, Heidelberg
24. Minematsu, K., Iwata, T.: Tweak-length extension for tweakable blockciphers. In: Groth, J. (ed.) 15th IMA International Conference on Cryptography and Coding. LNCS, vol. 9496, pp. 77–93. Springer, Heidelberg (Dec 2015)
25. Mitchell C.J.: On the security of XCBC, TMAC and OMAC. Technical Report RHUL-MA-2003-4, 19 August, 2003. Available at http://www.rhul.ac.uk/mathematics/techreports. Also available from NIST's web page at http://csrc.nist.gov/CryptoToolkit/modes/comments/
26. Nandi, M.: Bernstein bound on WCS is tight - repairing Luykx-Preneel optimal forgeries. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 213–238. Springer, Heidelberg (Aug 2018)
27. Peyrin, T., Wang, L.: Generic universal forgery attack on iterative hash-based MACs. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 147–164. Springer, Heidelberg (May 2014)
28. Phan R. C.W.: Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students. In Cryptologia, XXVI (4), 2002. https://staff.guilan.ac.ir/staff/users/rebrahimi/fckeditor_repo/file/mini-aes-spec.pdf.
29. Preneel, B., van Oorschot, P.C.: On the security of two MAC algorithms. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 19–32. Springer, Heidelberg (May 1996)
30. Rogaway P., Bellare M., Black J.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. Transactions on Information and System Security, Vol. 6, No. 3, August 2003, pp. 365-403.
31. Rogaway P., Shrimpton T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay S. (Eds.): EUROCRYPT 2006, LNCS 4004, pp. 373-390. Springer.
32. Shrimpton T., Terashima R.S.: A Modular Framework for Building Variable-Input-Length Tweakable Ciphers. In: Sako K., Sarkar P. (Eds.) ASIACRYPT 2013, Part I, LNCS 8269, pp. 405–423. Springer.

33. Sung, J., Hong, D., Lee, S.: Key recovery attacks on the RMAC, TMAC, and IACBC. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 03. LNCS, vol. 2727, pp. 265–273. Springer, Heidelberg (Jul 2003)
34. Wegman M.N., Carter L.: New Hash Functions and Their Use in Authentication and Set Equality. Journal of Computer and System Sciences 22, 265-279 (1981)
35. The CAESAR committee: CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html
36. IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Security. IEEE Std 802.1AE-2006 (2006).
37. Information Technology - Security Techniques - Authenticated Encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009).
38. NIST: Lightweight Cryptography. https://csrc.nist.gov/Projects/Lightweight-Cryptography
39. National Security Agency, Internet Protocol Security (IPsec) Minimum Essential Interoperability Requirements, IPMEIR Version 1.0.0 Core (2010), http://www.nsa.gov/ia/programs/suitebcryptography/index.shtml.
40. Sage Documentation. SageMath Help. http://www.sagemath.org/