Breaking REMUS and TGIF in the light of NIST Lightweight Cryptography Standardization Project

Nilanjan Datta[1], Ashwin Jha[1], Alexandre Mège[2] and Mridul Nandi[1]

[1] Indian Statistical Institute, Kolkata, India
[2] Airbus Defence and Space, Elancourt, France
nilanjan_isi_jrf@yahoo.com,ashwin.jha1991@gmail.com,alexandre.mege@airbus.com,
mridul.nandi@gmail.com

Abstract. In this paper, we propose forgery attacks on REMUS-N1, REMUS-M1, REMUS-N3, TGIF-N1, and TGIF-M1 in $D \approx 2^{43.6}$ bytes and $T \approx 2^{97.5}$. Further we show a key recovery attack on REMUS-N3 in $D \approx 2^{43.6}$ bytes and $T \approx 2^{97.5}$.

Keywords: REMUS · TGIF · lightweight · generic attack

1 Introduction

Authenticated encryption or AEAD[1] schemes are symmetric-key primitives, which can simultaneously achieve two information security goals: confidentiality and authenticity/integrity. In recent years, the cryptographic community has shown a lot of interest in the design and analysis of AEAD schemes. Cryptographic competitions such as the recently concluded CAESAR competition [3], and the recently started NIST lightweight cryptography standardization project [18], henceforth referred as "NIST LwC", have given new impetus to the research and development of highly efficient and secure AEAD schemes. While CAESAR was aimed to identify a portfolio of AEAD schemes designed for general purpose usages, the NIST LwC is focused on standardizing lightweight AEAD schemes for resource constrained environments such as sensor networks and IoT devices.

The first round of NIST LwC has 56 candidates, including REMUS [12] and TGIF [13]. REMUS offers three variants for nonce-respecting scenario, namely, REMUS-N1, REMUS-N2, and REMUS-N3, and two variants for nonce-misusing scenario, namely, REMUS-M1 and REMUS-M2. Of these five variants, REMUS-N1 is the primary submission to NIST LwC. Similarly, TGIF offers two variants for nonce-respecting scenario, namely, TGIF-N1, and TGIF-N2, and two variants for nonce-msiusing scenario, namely, TGIF-M1, and TGIF-M2. Of these TGIF-N1 is the primary submission to NIST LwC.

1.1 NIST LwC Security Requirements from AE Schemes

In the call for submission document [19], it is clearly mentioned that

> Cryptanalytic attacks on the AEAD algorithm shall require at least 2^{112} computations on a classical computer in a single-key setting.

Further, the document puts a restriction on the limit on maximum input size (plaintext, associated data, and the amount of data processed under one key). It states that

[1] Here AD represents Associated Data. Traditional AE setting requires integrity security for AD.

> The limits on the input sizes (plaintext, associated data, and the amount of data that can be processed under one key) for this member shall not be smaller than $2^{50} - 1$ bytes.

Let T and D denote the number of computations and the maximum amount of data processed under one key. One can clearly infer that the minimum security requirements from an AEAD scheme is as follows:

$$\text{If } D < 2^{50} \text{ bytes and } T < 2^{112}, \text{ then } \quad \text{is } \textit{secure}.$$

There are several notions of security for AEAD, including key recovery, message recovery, privacy, and authenticity. For each such notion there is a well-defined notion of attacker's *advantage*, which encapsulates the success probability, and hence lies in the interval $[0, 1]$. An advantage value of at least 0.5 indicates that the attack succeeds with high probability. In general a fixed constant value , say $\epsilon \in (0, 1]$, is chosen as the desired tolerance level, and we say that a scheme is insecure if some attack succeeds with at least ϵ advantage. Although, NIST LwC has not specified any tolerance level, however for analysis purposes, $\epsilon = 0.5$ could be a very generous choice. Clearly, an attack X violates under NIST LwC requirements when:

$$D(\mathsf{X}) < 2^{50} \text{ bytes and } T(\mathsf{X}) < 2^{112}, \text{ and } \mathsf{X} \text{ attacks } \quad \text{with at least 0.5 advantage,}$$

where $T(\mathsf{X})$ and $D(\mathsf{X})$ denote the computation time and data complexity of the attack X.

1.1.1 Data Limit

The meaning of parameter D is quite clear from the document. D denotes the data complexity of the attack. This parameter quantifies the online (queries to the AEAD scheme) resource requirements, and includes the total number of blocks (among all messages/ciphertexts and associated data) processed through the underlying primitive for a fixed master key.

1.1.2 Computation Time Limit

The computation time T of an attack refers to the time complexity of the attack. This parameter quantifies the offline resource requirements, and includes the total time required to process the offline evaluations of the underlying block cipher. Since one call of the block cipher can be assumed to take a constant amount of time,[2] it is generally ignored, and the number of primitive evaluations is taken as the time complexity of evaluations.

We remark that, the direct evaluations of the primitives have been considered within time complexity in multiple papers and in different scenarios: For instance, the time-memory trade-off attack by Hellman [11] and related-key attacks on AES-256 [2], attacks on hash functions [15, 14, 9, 1], attacks on HMAC and NMAC [20, 16, 21, 10, 6], attacks on Even-Mansour ciphers [7, 4, 5, 8], and multi-key attacks on Even-Mansour cipher [17]. In fact, this also makes sense in real scenario, where the adversary can actually make block cipher evaluations on its own by devoting sufficient time.

In this regard, we note that REMUS-N1, REMUS-N3, REMUS-M1, TGIF-N1, and TGIF-M1 restrict the number of offline evaluations of the underlying block cipher (the primitive) to less than 2^{64}. In light of the above discussion, it is clear that this violates the NIST LwC requirements as stated above, as the adversary is allowed make beyond 2^{64} (anything below 2^{112} is valid) block cipher evaluations. This is especially required from REMUS-N1 and TGIF-N1, which are the primary variants in their respective submissions.

[2] A rigorous analysis assumes that the time complexity of one evaluation of the primitive is $O(n^k)$, where n is the input size of the primitive, and $k \in \mathbb{N}$ is a constant.

1.2 Our Contributions

In this paper, we propose forgery attacks on REMUS-N1, REMUS-M1, REMUS-N3, TGIF-N1, and TGIF-M1, where $D \approx 2^{50}$ bytes and $T < 2^{100}$ and the attack advantage is close to 1. We further show a key recovery attack on REMUS-N3 with $D < 2^{50}$ bytes and $T < 2^{100}$, and the attack advantage is close to 1.

2 Forgery and Key Recovery against REMUS and TGIF

Throughout we use $n = 128$ as block size and key size of the underlying block cipher. We denote nonce size by r. For N1 and M1 variants $r = 128$, and for the N3 variant $r = 96$. Our forgery and key recovery utilize an underlying state recovery attack. We first describe the state recovery attack and then give the forgery and key recovery attacks.

2.1 Recovering the Nonce-based Key L

The key derivation function KDF_K takes a nonce value N as input and outputs a nonce-based key L. In case of the N1 and M1 variants, $\mathsf{KDF}_K(N) := E_K(N)$, and in case of the N3 variant, $\mathsf{KDF}_K(N) := K \oplus N \| 0^{32}$.
We employ the following algorithm to find the nonce-based key corresponding to one nonce value:
Algorithm 1:

1. Let $t \geq 32$ be a parameter of the algorithm, and $d = n - t$.
2. For $i = 0$ to $2^t - 1$:
   - Set $L^i = 0^d \| \langle i \rangle_t$, where $\langle i \rangle_t$ denotes the t-bit representation of integer i.
   - Simulate the encryption of (A, M) using L^i as the nonce-based key, where $|A| = |M| = n$. Response: (C^i, τ^i). Store (L^i, C^i, τ^i) in a list H.
3. Sort entries in H on second and third coordinates, i.e. (C, τ).
4. For $j = 0$ to $2^d - 1$:
   - Set $\hat{N}^j = \langle j \rangle_d \| 0^{r-d}$. Note that $r - d \geq 0$ due to $t \geq 32$.
   - Query $(\hat{N}^j, A, M)$ to the encryption oracle of AEAD. Response: $(\hat{C}^j, \hat{\tau}^j)$.
   - Search $(\hat{C}^j, \hat{\tau}^j)$ in H (binary search would suffice). Suppose there exist index $i \in H$ such that $(\hat{C}^j, \hat{\tau}^j) = (C^i, \tau^i)$ then it would mean that $\hat{L}^j = L^i$ with very high probability. Matching both ciphertext and tag helps in avoiding false positives. This is because such matchings happen in the ideal world with probability $\approx 2^{-2n}$, whereas for the real schemes this happens with probability 1 when $\hat{L}^j = L^i$.

Note that, we could have mounted a slightly more simple attack on N1 and M1 (choose 2^t L^i's in without replacement manner and use 2^d distinct nonces in encryption queries). But, to give a combined attack on N1, M1, and N3, we have to use the above strategy.

2.2 Key Recovery Attack against REMUS-N3

In REMUS-N3, the key can be directly recovered from a valid nonce and nonce-based key pair (N', L') obtained using Algorithm 1 of subsection 2.1. The master key K is computed as

$$K = L' \oplus N' \| 0^{32}.$$

Once the master key K is recovered any other attack is easily possible.

2.3 Forgery against REMUS -N1/N3/M1 and TGIF -N1/M1

One can also construct nonce-respecting[3] forgery attacks on REMUS-N1 (primary version), REMUS-N3, and REMUS-M1, given the nonce-based key recovered in Algorithm 1. On a closer inspection, we found that the same attack also works on TGIF-N1 (primary version) and TGIF-M1 as well. Basically, the attack recovers some valid nonce and nonce-based key pair (N', L') using Algorithm 1 of subsection 2.1, which can then be used trivially to construct valid forgeries of the form (N', A', C', T'), where A' and C' can be chosen arbitrarily, and the tag is computed using L', A' and C'.

2.4 Complexity of the Attack

It is easy to see that Algorithm 1 has the following complexity:

1. Data complexity, $D \approx 2^{d+5.6}$ bytes. The factor of 5.6 is due to the fact that each encryption query consists of $3 \approx 2^{1.6}$ blocks of data and each block contains 2^4 bytes.
2. Total time complexity, $T \approx 2^{t+5.6} + t \cdot 2^t + t \cdot 2^{n-t}$.

Here the first, second, and third terms in the time complexity correspond to time complexity of step 2, 3, and 4, respectively, of Algorithm 1 in subsection 2.1.

The time complexity of key recovery and forgeries can be made negligible given the nonce based key L recovered in Algorithm 1.

Algorithm 1 works for all choices of $t \geq 32$, as $d + t = 128$. Specifically, to reduce the data complexity below the NIST LwC requirement, we set $t = 90$, which gives $d = 38$. For this choice of t, we get $D \approx 2^{43.6}$ bytes and $T \approx 2^{97.5}$, which clearly falls within the NIST LwC minimum data and time limit.

Remark 1. We remark that there is a scope of improvement in time complexity of Algorithm 1 by using a hash table instead of a list. Similarly, one can improve data complexity by using empty message and empty AD. However, this may lead to some false positives which can be eliminated by making constant number of checking queries. We do not use the empty message and AD case, as such inputs seldom occur in real scenario.

2.5 Inherent Weakness of REMUS-N1/N3/M1 and TGIF-N1/M1

We would like to point out that the N1/N3/M1 variants of REMUS (and N1/M1 variants of TGIF) have an inherent weakness: insufficient randomness in the initial state (key,input). Although the key is derived using nonce for each encryption query, the adversary can easily fix a constant value as the initial input. So, to create an initial state collision the adversary just needs to collide the initial key.

In summary, it seems that REMUS-N1/N3/M1 and TGIF-N1/M1 do not satisfy the minimum security requirements of NIST LwC.

References

[1] Elena Andreeva, Charles Bouillaguet, Orr Dunkelman, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. New second-preimage attacks on hash functions. *J. Cryptology*, 29(4):657–696, 2016.

[2] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and related-key attack on the full AES-256 (extended version). *IACR Cryptology ePrint Archive*, 2009:241, 2009.

[3] Nonce is not repeated in encryption queries to the AEAD. But it can be repeated in forge queries.

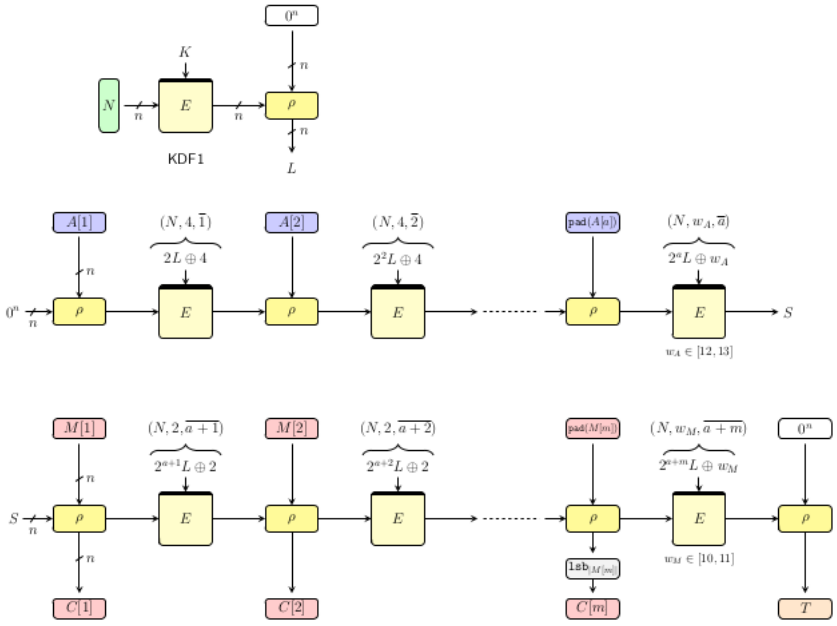

Figure 1: REMUS-N1 N1 Authenticated Encryption Mode

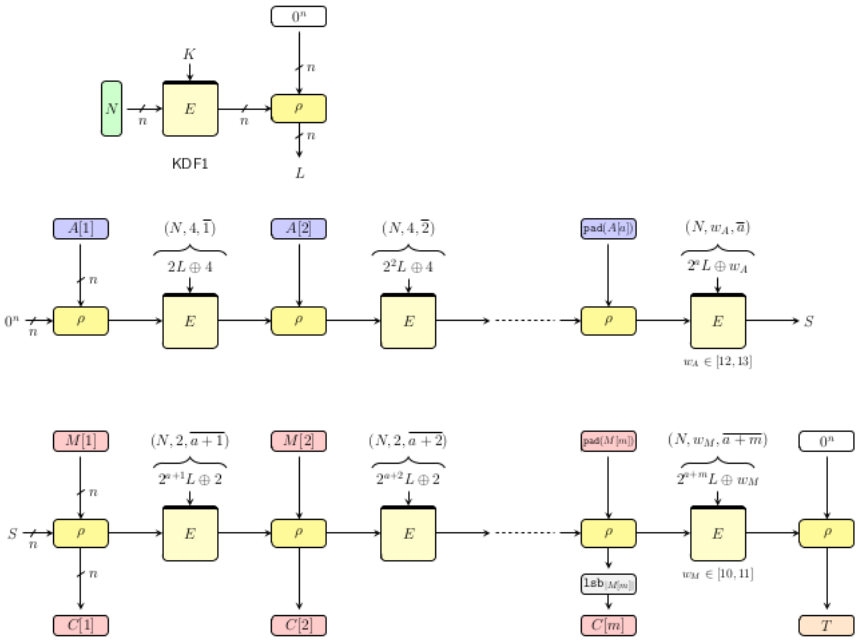

Figure 2: TGIF-N1 Authenticated Encryption Mode

[3] CAESAR. CAESAR: competition for authenticated encryption: Security, applicability, and robustness, 2015. Online: https://competitions.cr.yp.to/caesar.html. Accessed: August 01, 2019.

[4] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Key recovery attacks

on 3-round even-mansour, 8-step led-128, and full AES2. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, pages 337–356, 2013.

[5] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Cryptanalysis of iterated even-mansour schemes with two keys. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 439–457, 2014.

[6] Itai Dinur and Gaëtan Leurent. Improved generic attacks against hash-based macs and HAIFA. *Algorithmica*, 79(4):1161–1195, 2017.

[7] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The even-mansour scheme revisited. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 336–354, 2012.

[8] Orr Dunkelman, Nathan Keller, and Adi Shamir. Slidex attacks on the even-mansour encryption scheme. *J. Cryptology*, 28(1):1–28, 2015.

[9] Jian Guo, Jérémy Jean, Gaëtan Leurent, Thomas Peyrin, and Lei Wang. The usage of counter revisited: Second-preimage attack on new russian standardized hash function. In *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, pages 195–211, 2014.

[10] Jian Guo, Thomas Peyrin, Yu Sasaki, and Lei Wang. Updates on generic attacks against HMAC and NMAC. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 131–148, 2014.

[11] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, 26(4):401–406, 1980.

[12] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. REMUS v1.0. Submission to NIST LwC Standardization Process (Round 1), 2019. Online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Remus-spec.pdf. Access: August 01, 2019.

[13] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Ling Sun. Thank Goodness It's Friday (TGIF) v1.0. Submission to NIST LwC Standardization Process (Round 1), 2019. Online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TGIF-spec.pdf. Access: August 01, 2019.

[14] John Kelsey and Tadayoshi Kohno. Herding hash functions and the nostradamus attack. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 183–200, 2006.

[15] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 474–490, 2005.

[16] Gaëtan Leurent, Thomas Peyrin, and Lei Wang. New generic attacks against hash-based macs. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 1–20, 2013.

[17] Nicky Mouha and Atul Luykx. Multi-key security: The even-mansour construction revisited. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 209–223, 2015.

[18] NIST. Lightweight cryptography, 2018. Online: https://csrc.nist.gov/Projects/Lightweight-Cryptography. Accessed: August 01, 2019.

[19] NIST. Submission requirements and evaluation criteria for the lightweight cryptography standardization process, 2018. Online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf. Accessed: August 01, 2019.

[20] Thomas Peyrin, Yu Sasaki, and Lei Wang. Generic related-key attacks for HMAC. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 580–597, 2012.

[21] Thomas Peyrin and Lei Wang. Generic universal forgery attack on iterative hash-based macs. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 147–164, 2014.