Key Committing Security Analysis of AEGIS

Takanori Isobe and Mostafizar Rahman

University of Hyogo, Japan
takanori.isobe@ai.u-hyogo.ac.jp
mrahman454@gmail.com

Abstract. Recently, there has been a surge of interest in the security of authenticated encryption with associated data (AEAD) within the context of key commitment frameworks. Security within this framework ensures that a ciphertext chosen by an adversary does not decrypt to two different sets of key, nonce, and associated data. Despite this increasing interest, the security of several widely deployed AEAD schemes has not been thoroughly examined within this framework. In this work, we assess the key committing security of AEGIS, which emerged as a winner in the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR). A recent assertion has been made suggesting that there are no known attacks on AEGIS in the key committing settings and AEGIS qualifies as a fully committing AEAD scheme in IETF document. However, contrary to this claim, we propose a novel $O(1)$ attack applicable to all variants of AEGIS. This demonstrates the ability to execute a key committing attack within the FROB game setting, which is known to be one of the most stringent key committing frameworks. This implies that our attacks also hold validity in other, more relaxed frameworks, such as CMT-1, CMT-4, and so forth.

Keywords: AEGIS · Key Commitment

1 Introduction

Authenticated Encryption (AE) is a cryptographic technique that combines encryption and message authentication codes (MACs) to provide both confidentiality and integrity for data. It ensures that not only is the information kept secret from unauthorized parties, but also that it has not been tampered with during transit. AEGIS, proposed by Wu and Preneel [WP13a], is one such scheme and its variant AEGIS-128 emerged as one of the winning candidate of Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [cae19] for high performance computing applications.

The traditional focus of designers in authenticated encryption with associated data (AEAD) has been on ensuring the security aspects of confidentiality and ciphertext integrity. However, in recent years it is witnessed that the previously established notions of confidentiality and integrity may not suffice in various contexts. Among the additional properties explored is the concept of authenticated encryption (AE) key commitment, an area that has received relatively less attention.

Key commitment assures that a ciphertext C can only be decrypted using the same key that was originally used to derive C from some plaintext. Schemes that allow finding a ciphertext that decrypts to valid plaintexts under two different keys do not adhere to the principle of key commitment. The issue of non-key-committing AEAD was initially highlighted in scenarios such as moderation within encrypted messaging [DGRW18, GLR17]. Subsequently, it surfaced in various applications including password-based encryption [LGR21], password-based key exchange [LGR21], key rotation schemes [ADG+22], and envelope encryption [ADG+22].

In even more recent times, there have been new propositions [CR22, BH22] introducing definitions that focus on committing to not only the key, but also the associated data and nonce. Although there have been suggestions for novel schemes [CR22, ADG+22] that align with these diverse definitions, uncertainties persist regarding which existing AEAD schemes actually implement this commitment, and in what manner. Furthermore, several crucial and widely-used AEAD schemes lack demonstrated commitment results. Recently, commitment attacks are mounted on several widely deployed AEAD schemes, like CCM, GCM, OCB3, etc [MLGR23].

In this work, we assess the key committing security of AEGIS. A recent assertion has been made suggesting that there are no known attacks on AEGIS in the key committing settings [DL23a] and AEGIS qualifies as a fully committing AEAD scheme [MST23]. The challenge of attacking the key committing security of AEGIS is also acknowledged as an open problem in [Kö22]. In [DL23a], it is claimed that finding a collision on a 128-bit tag for variants of AEGIS requires $O(2^{64})$ computations, while for a 256-bit tag, it requires $O(2^{128})$ computations. These claims are made under the assumption that AEGIS is fully committing. However, contrary to all these claims, we demonstrate the ability to execute a key committing attack within the FROB game setting [FOR17], which is known to be one of the most stringent key committing frameworks. Thus, we are able to find collisions on tags with a complexity of $O(1)$. This implies that our attacks also hold validity in other, more relaxed frameworks, such as CMT-1, CMT-4, and so forth.

We have informed our results to the authors of IETF document, Denis and Lucas. They have confirmed our results and will update the IETF document accordingly [DL23b].

2 Preliminaries

2.1 Committing Authenticated Encryption (AE) Framework

Consider a symmetric encryption scheme Σ consisting of encryption and decryption algorithms denoted by Σ_{Enc} and Σ_{Enc}, respectively where

$$\Sigma_{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \rightarrow \mathcal{C},$$

and

$$\Sigma_{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}.$$

Here, $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{C}$ refer to the key, nonce, associated data, message and ciphertext spaces, respectively. Formally, the above scheme is called as a *nonce based authenticated encryption scheme supporting associated data*, or an nAE scheme.

A committing authenticated encryption (cAE) scheme guarantees the definitive determination of the values of its constituent elements, including the key, nonce, associated data, or message, which are utilized to produce the ciphertext. In the committing AE framework, the adversary tries to construct a ciphertext which can be obtained from two different sets of keys, nonces, associated data and messages. Let, $C_i \leftarrow \Sigma_{Enc}(K_i, N_i, A_i, M_i)$ where $K_i \in \mathcal{K}$, $N_i \in \mathcal{N}$, $A_i \in \mathcal{A}$, $M_i \in \mathcal{M}$ and $C_i \in \mathcal{C}$ for $i \in \{1, 2\}$. The adversary aims to find C_1, C_2 such that $C_1 = C_2$ and $(K_1, N_1, A_1, M_1) \neq (K_2, N_2, A_2, M_2)$.

Various notions of committing security framework have been introduced [FOR17, CR22, BH22]. In the context of this work, we discuss here some of them. In CMT-1, the ciphertext commits exclusively to the key. In the attack scenario, the adversary must produce $((K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2))$ such that $K_1 \neq K_2$ and $\Sigma_{Enc}(K_1, N_1, A_1, M_1) = \Sigma_{Enc}(K_2, N_2, A_2, M_2)$. CMT-4 relaxes the constraints and allows that the commitment can encompass to any of the inputs of Σ_{Enc}, not just the key. The adversary can breach CMT-4 security by constructing a set $((K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2))$ such that, $(K_1, N_1, A_1, M_1) = (K_2, N_2, A_2, M_2)$ and $\Sigma_{Enc}(K_1, N_1, A_1, M_1) = \Sigma_{Enc}(K_2, N_2, A_2, M_2)$.

FROB ($\mathcal{A}$)

1. $(C, (K_1, N_1, A_1), (K_2, N_2, A_2)) \xleftarrow{\$} \mathcal{A}$
2. $M_1 \leftarrow \Sigma_{Dec}(K_1, N_1, A_1, C)$
3. $M_2 \leftarrow \Sigma_{Dec}(K_2, N_2, A_2, C)$
4. **If** $M_1 = \perp$ or $M_2 = \perp$ **then** Return false
5. **If** $K_1 = K_2$ or $N_1 \neq N_2$ **then** Return false
6. Return true

(a) FROB Game

CMT-1($\mathcal{A}$)

1. $(C, (K_1, N_1, A_1), (K_2, N_2, A_2)) \xleftarrow{\$} \mathcal{A}$
2. $M_1 \leftarrow \Sigma_{Dec}(K_1, N_1, A_1, C)$
3. $M_2 \leftarrow \Sigma_{Dec}(K_2, N_2, A_2, C)$
4. **If** $M_1 = \perp$ or $M_2 = \perp$ **then** Return false
5. **If** $K_1 = K_2$ **then** Return false
6. Return true

(b) CMT-1 Game

CMT-3($\mathcal{A}$)

1. $(C, (K_1, N_1, A_1), (K_2, N_2, A_2)) \xleftarrow{\$} \mathcal{A}$
2. $M_1 \leftarrow \Sigma_{Dec}(K_1, N_1, A_1, C)$
3. $M_2 \leftarrow \Sigma_{Dec}(K_2, N_2, A_2, C)$
4. **If** $M_1 = \perp$ or $M_2 = \perp$ **then** Return false
5. **If** $(K_1, N_1, A_1) = (K_2, N_2, A_2)$ **then** Return false
6. Return true

(c) CMT-3 Game

CMT-4($\mathcal{A}$)

1. $(C, (K_1, N_1, A_1), (K_2, N_2, A_2)) \xleftarrow{\$} \mathcal{A}$
2. $M_1 \leftarrow \Sigma_{Dec}(K_1, N_1, A_1, C)$
3. $M_2 \leftarrow \Sigma_{Dec}(K_2, N_2, A_2, C)$
4. **If** $M_1 = \perp$ or $M_2 = \perp$ **then** Return false
5. **If** $(K_1, N_1, A_1, M_1) = (K_2, N_2, A_2, M_2)$ **then** Return false
6. Return true

(d) CMT-4 Game

Figure 1: Different Frameworks for Key Commiting Security.

Bellare and Hoang introduced CMT-3, which is slightly more restrictive than CMT-4. They replaced the constraint $(K_1, N_1, A_1, M_1) = (K_2, N_2, A_2, M_2)$ with $(K_1, N_1, A_1) = (K_2, N_2, A_2)$. The FROB game, initially proposed by Farshim, Orlandi, and Rosie [FOR17] and later adapted to the AEAD setting by Grubbs, Lu, and Ristenpart [GLR17], is even more restrictive. It requires the condition $N_1 \neq N_2$ in addition to $K_1 = K_2$. It has been demonstrated that CMT-3 security implies CMT-1, which in turn implies the FROB game [BH22, MLGR23]. In essence, the FROB game presents the most formidable challenge for an adversary to overcome. All the related games are outlined in Fig. 1.

2.2 Description of AEGIS

The authenticated encryption scheme AEGIS was introduced in SAC 2013 [WP13a]. It encompasses three variants: AEGIS-128 (AEGIS-128 emerged as a finalist in the CAESAR competition [cae19]), AEGIS-256, and AEGIS-128L. Across all these variants, the state update function involves a single round of AES denoted as $AR(X, Y)$, where X and Y represent 16-byte states. Specifically, $AR(X, Y) = MC \circ SR \circ SB(X) \oplus Y$, where MC, SR, and SB denote the mixcolumns, shiftrows, and subbytes operations, respectively. For more details on these operations refer to [DR00, DR02].

The state update function of AEGIS-128 and AEGIS-256 involves updating the 16-byte state S_i with a 16-byte message block m_i to yield the state S_{i+1}. This operation is expressed as:

$$
\begin{aligned}
S_{i+1,0} &= AR(S_{i,b-1}, S_{i,0} \oplus m_i) \\
S_{i+1,1} &= AR(S_{i,0}, S_{i,1}) \\
&\vdots \\
S_{i+1,b-1} &= AR(S_{i,b-2}, S_{i,b-1}).
\end{aligned}
$$

For AEGIS-128 and AEGIS-256, the value of b is 5 and 6, respectively, resulting in state sizes of 80 bytes and 96 bytes, respectively.

The state update function of AEGIS-256 differs slightly from the other two, using two 16-byte message blocks $m_{i,0}$ and $m_{i,1}$ instead of one. The computation is as follows:

$$
\begin{aligned}
S_{i+1,0} &= AR(S_{i,7}, S_{i,0} \oplus m_{i,0}) \\
S_{i+1,1} &= AR(S_{i,0}, S_{i,1}) \\
S_{i+1,2} &= AR(S_{i,1}, S_{i,2}) \\
S_{i+1,3} &= AR(S_{i,2}, S_{i,3}) \\
S_{i+1,4} &= AR(S_{i,3}, S_{i,4} \oplus m_{i,1}) \\
S_{i+1,5} &= AR(S_{i,4}, S_{i,5}) \\
S_{i+1,6} &= AR(S_{i,5}, S_{i,6}) \\
S_{i+1,7} &= AR(S_{i,6}, S_{i,7}).
\end{aligned}
$$

In the initialization phase, the state of AEGIS is loaded with a 128-bit key K, a 128-bit initialization vector IV, and some constants. For AEGIS-128 and AEGIS-128L, the sizes of K and IV are 128 bits, while for AEGIS-256, they are 256 bits. The state update function is iterated 10 times for AEGIS-128 and AEGIS-128L, and 16 times for AEGIS-256.

Following this, based on the lengths of the associated data and plaintext, the states undergo further updates. The associated data and plaintext are encrypted concurrently with the state update function. After each step of the state update function, a 128-bit block of associated data/plaintext is encrypted for AEGIS-128 and AEGIS-256 (for AEGIS-128L, two 128-bit blocks are encrypted at each step).

Finally, during tag generation, the state update function is iterated for 7 rounds. The message bit depends on the lengths of the plaintext and associated data, encoded as 64-bit strings, along with a portion of the previous state. All the 128-bit substates of the final state are XOR-ed to obtain the tag. For more comprehensive details on AEGIS, please refer to [WP13a, WP13b, WP16].

3 Attacks

3.1 Attack Overview

Initially, let's introduce an alternative perspective on the state updating process of AEGIS. Since the state update relies on the key, IV, associated data (AD), and plaintext at various stages, we can view the entire process as illustrated in Fig.2. As explained in Section2.2, the initialization phase is contingent on the key K and the initialization vector IV. Therefore, the complete state update process during this phase can be denoted as $\mathcal{U}_{K,IV}$, which transforms the initial state IS_0 into IS_1. Subsequently, $\mathcal{U}_A$ and $\mathcal{U}P$ alter the internal states IS_1 and IS_2 into IS_2 and IS_3 respectively, based on the associated data A and plaintext P. Finally, contingent on the lengths of A and P, $\mathcal{U}_{|P|,|A|}$ transforms IS_3 into IS_4. The tag is generated based on IS_4.

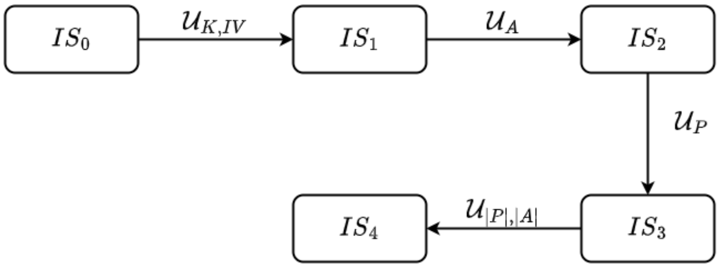

Figure 2: State updatation as a function of key, initialization vector, associated data and plaintext.

We are specifically interested in analyzing the FROB security of AEGIS. As outlined in Section 2.1, the adversary is required to generate a ciphertext (ciphertext and tag pair) which decrypts to valid plaintexts using two different sets of keys and same IV. Let's consider two sets of key, IV, AD, and plaintext, denoted as (K_1, IV_1, A_1, P_1) and (K_2, IV_2, A_2, P_2). These sets are used to create ciphertext-tag pairs $C_1||T_1$ and $C_2||T_2$ respectively. Note that, $K_1 \neq K_2$ and $IV_1 = IV_2$.

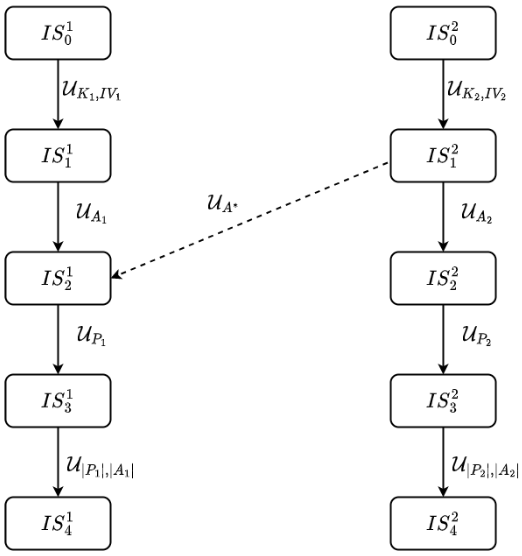

Figure 3: Overview of the attack in FROB framework

As depicted in Fig. 3, we need to find a A^* such that $\mathcal{U}_{A^*}$ transforms IS_1^2 to IS_2^1. If $|A^*| = |A_1|$ (the plaintext is P_1), the final state IS_4^1 can be obtained which results in generating the ciphertext-tag pair $C_1||T_1$. Consequently, the tuples (K_1, IV_1, A_1, P_1) and (K_2, IV_2, A^*, P_1) yield the same ciphertext-tag pair, thereby compromising the FROB security of AEGIS. Hence, the adversary is required to find an A^* such that $|A^*| = |A_1|$.

3.2 Attacks on AEGIS

In this subsection, we primarily focus on the recovery of the associated data A^* in the case of AEGIS-128. The recovery of A^* for AEGIS-256 and AEGIS-128L follows a similar strategy.

Please refer to Fig. 4 for an overview of the attack. Corresponding to the discussion in Section 3.1 and Fig. 3, the states $S_{i,0}||S_{i,1}||S_{i,2}||S_{i,3}||S_{i,4}$ and $S_{i+5,0}||S_{i+5,1}||S_{i+5,2}||S_{i+5,3}||S_{i+5,4}$ can be considered as IS_1^2 and IS_2^1, respectively.

Let $A^* = A_0^*||A_1^*||A_2^*||A_3^*||A_4^*$, where each A_j^* (for $0 \leq j \leq 4$) is a 16-byte block. Based on the values of the substates $Si, 0, \cdots, Si, 4$, some of the internal substates' values can be fixed (indicated by the red rectangles in Fig. 4).

Now, when the value of $S_{i+5,4}$ is fixed, it deterministically establishes the internal substates $S_{i+k+1,k}$ for $0 \leq k \leq 3$ (indicated by the blue rectangles in Fig. 4). The values of $S_{i,0}$, $S_{i,4}$, and $S_{i+1,0}$ deterministically determine the value of A^0. In general, by fixing the value of $S_{i+5,k}$, A_{4-k}^* can be determined. Hence, the complete A^* can be deterministically recovered.

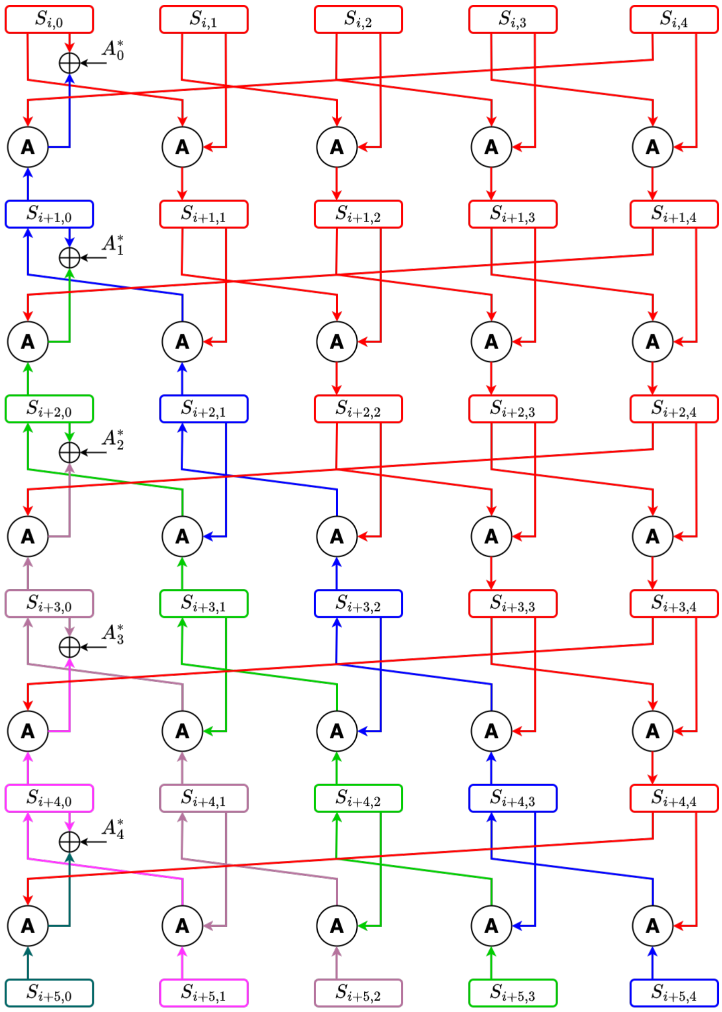

Figure 4: Attack on AEGIS-128

By following the similar strategy, A^* can be recovered for both AEGIS-256 and AEGIS-128L. The attack strategy corresponding to AEGIS-256 and AEGIS-128L are outlined in Fig. 5 and Fig. 6, respectively. The attack vectors corresponding to the attack on AEGIS-128, AEGIS-256 and AEGIS-128L are provided in Appendix A.1, A.2 and A.3, respectively.

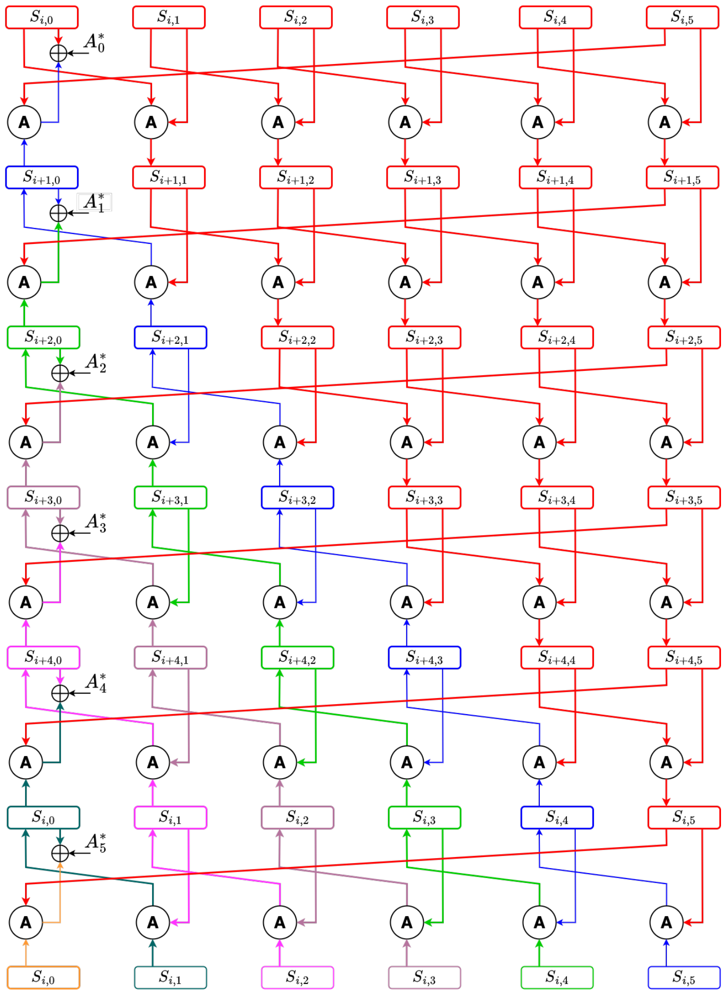

Figure 5: Attack on AEGIS-256

4 Conclusion

The issue of key commitment security in AEGIS has been a significant and persisting question. This work addresses this gap by conducting a thorough analysis of AEGIS. Our analysis, considering various existing frameworks, culminated in the development of a $O(1)$ attack applicable to all variants of AEGIS. However, in frameworks where an additional constraint of identical associated data is imposed (i. e., $A_1 = A_2$), the proposed attacks will not be effective. These findings underscore the need for continued research and evaluation in the domain of AEAD security, particularly in the context of key commitment frameworks.

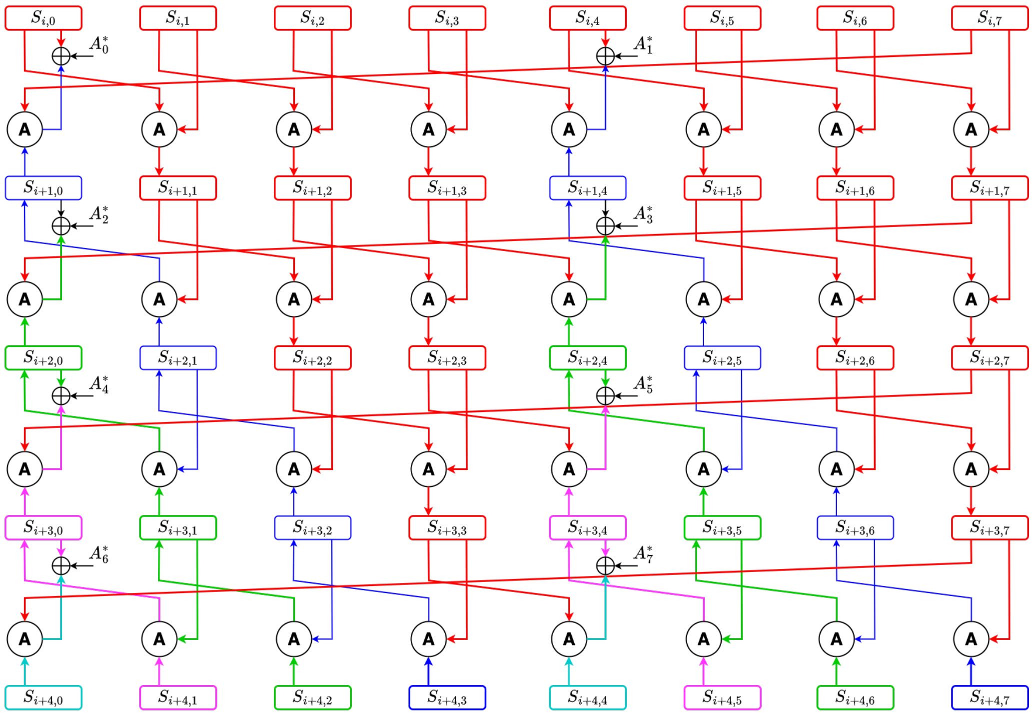

Figure 6: Attack on AEGIS-128L

References

[ADG+22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 3291–3308. USENIX Association, 2022.

[BH22] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875. Springer, 2022.

[cae19] Caesar: Competition for authenticated encryption: Security, applicability, and robustness. https://competitions.cr.yp.to/caesar-submissions.html, 2019.

[CR22] John Chan and Phillip Rogaway. On committing authenticated-encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 275–294. Springer, 2022.

[DGRW18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186. Springer, 2018.

[DL23a] Frank Denis and Samuel Lucas. The aegis family of authenticated encryption algorithms. https://datatracker.ietf.org/doc/draft-irtf-cfrg-aegis-aead/04/, 2023.

[DL23b] Frank Denis and Samuel Lucas. Private communication, 2023. Email correspondence.

[DR00] Joan Daemen and Vincent Rijmen. Rijndael for AES. In *The Third Advanced Encryption Standard Candidate Conference, April 13-14, 2000, New York, New York, USA*, pages 343–348. National Institute of Standards and Technology,, 2000.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[FOR17] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symmetric Cryptol.*, 2017(1):449–473, 2017.

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International*

Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97. Springer, 2017.

[Kö22] Stefan Kölbl. Open questions around key committing aeads. https://frisiacrypt2022.cs.ru.nl/assets/slides/stefan-frisiacrypt2022.pdf, 2022.

[LGR21] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 195–212. USENIX Association, 2021.

[MLGR23] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break ccm, eax, siv, and more. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 379–407. Springer, 2023.

[MST23] John Preuß Mattsson, Ben Smeets, and Erik Thormarker. Proposals for standardization of encryption schemes. https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/Practical%20Challenges%20with%20AES-GCM.pdf, 2023.

[WP13a] Hongjun Wu and Bart Preneel. AEGIS: A Fast Authenticated Encryption Algorithm. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 185–201. Springer, 2013.

[WP13b] Hongjun Wu and Bart Preneel. Aegis: A fast authenticated encryption algorithm. Cryptology ePrint Archive, Paper 2013/695, 2013.

[WP16] Hongjun Wu and Bart Preneel. Aegis: A fast authenticated encryption algorithm (v1.1). https://competitions.cr.yp.to/round3/aegisv11.pdf, 2016.

A Attack Vectors

Note that, in the attack vectors, we have provided a ciphertext/tag pair. However, the tuple $((K_1, IV_1, A_1), (K_2, IV_2, A^*))$ (here $IV_1 = IV_2$) works with any plaintext, i. e., if we encrypt a plaintext with both (K_1, IV_1, A_1) and (K_2, IV_2, A^*), it generates same ciphertext/tag pair. In this way, numerous ciphertext/tag pair can be generated which can be decrypted to valid plaintexts.

In the vectors provided, the leftmost bit is the least significant bit (LSB). Consider a 16-bit string $b_0 \cdots b_{15}$ wher b_0 is the LSB and b_{15} is the most significant bit (MSB). Using the vectors, the above string is denoted as $[b_0 \cdots b_7 \quad b_8 \cdots b_{15}]$.

A.1 Attack Vector for AEGIS-128

$C\|\|T=$	[0xA5	0xA7	0x7C	0x8D	0x8D	0xB5	0xEB	0x88	0x35	0x72
	0x71	0x78	0xDA	0x00	0x15	0xFF	0xBC	0x1D	0xB4	0xF6
	0x28	0x7B	0x96	0xEE	0x1E	0xA0	0xF8	0xEC	0x0C	0xFF
	0x32	0x4B]								
$K_1=$	[0x62	0x1F	0x61	0xFA	0x65	0x84	0x70	0xCC	0x18	0x4B
	0x39	0x45	0x3D	0xAB	0x75	0x80]				
$IV_1=$	[0xCE	0xD7	0xE2	0xF0	0xB2	0xAE	0x0D	0x0D	0x3E	0x82
	0x5F	0xFC	0xE4	0x6F	0xC7	0xCF]				
$A_1=$	[0xBE	0x17	0x84	0xAA	0x3B	0x98	0x29	0xBC	0xCC	0xF3
	0x81	0x04	0x11	0x57	0x4F	0x43	0xFB	0x86	0xA4	0xE3
	0xD6	0x34	0x1C	0x15	0xB7	0x07	0x8E	0x2C	0x91	0x75
	0x86	0xE2	0x89	0x94	0x5D	0x69	0x85	0x55	0xB0	0xEE
	0x68	0x70	0x27	0x71	0xF1	0x0A	0xF8	0x89	0x30	0xF9
	0x35	0x7B	0x8D	0xFE	0x1F	0x07	0xD1	0x6F	0x39	0xD2
	0x44	0x1D	0xC3	0x83	0x31	0x65	0xAF	0x74	0x55	0x03
	0xA6	0xB3	0xD3	0x2C	0x15	0x8C	0x86	0xA3	0xFA	0xCF]
$K_2=$	[0xFC	0xF9	0x24	0xED	0x84	0x21	0x9B	0xD8	0x24	0xEB
	0x58	0xB9	0x01	0xA8	0x08	0x82]				
$IV_2=$	[0xCE	0xD7	0xE2	0xF0	0xB2	0xAE	0x0D	0x0D	0x3E	0x82
	0x5F	0xFC	0xE4	0x6F	0xC7	0xCF]				
$A^*=$	[0x15	0x7E	0xC0	0x40	0x64	0xDB	0x40	0x47	0xDC	0xE2
	0x56	0x7D	0x41	0x6C	0x5D	0x08	0x71	0xB4	0xDB	0xD8
	0x76	0xC5	0xCC	0xD1	0x44	0xF0	0x58	0x91	0xF5	0xED
	0x22	0x91	0x3F	0xA8	0xEC	0x97	0x71	0xD5	0xD2	0x7C
	0x28	0xF7	0x53	0xBB	0xE0	0x5A	0xD1	0xBF	0x34	0xF2
	0x44	0x14	0xE7	0x37	0x88	0x61	0xB3	0x0E	0x5C	0x75
	0x61	0x84	0xBE	0x03	0x0F	0xBB	0x57	0xF1	0x3B	0x2D
	0x93	0x74	0xCB	0x70	0x57	0xFC	0x9D	0xF9	0xE4	0x2B]

A.2 Attack Vector for AEGIS-256

$C||T=$ [0x5F 0x74 0x00 0x73 0x1E 0x88 0x1D 0x84 0xAE 0x0A
0x18 0xE2 0x16 0x9B 0x6E 0x98 0xB0 0x8D 0x5C 0xB1
0x74 0x9F 0x53 0x80 0xF6 0xE0 0x9B 0x0F 0x33 0x1D
0x42 0xF0]

$K_1=$ [0x15 0x86 0x32 0x3E 0x9C 0x71 0xB4 0x9F 0x13 0x36
0xAC 0x8D 0x7D 0x37 0x1B 0x9B 0x7A 0x80 0x0D 0x63
0x7D 0x27 0x46 0xFF 0x5C 0x55 0x0E 0x5A 0xEC 0xE7
0x8C 0x81]

$IV_1=$ [0xD9 0x9D 0x22 0x35 0x4E 0xF7 0x15 0xF8 0x70 0x88
0xEF 0x8E 0x88 0xBE 0xC0 0x1C 0x6A 0xD7 0xFE 0xDF
0x43 0xF7 0x8D 0x61 0x5D 0x88 0xB9 0x00 0xCA 0x62
0x29 0xF0]

$A_1=$ [0x8E 0x15 0x9D 0xB0 0x18 0x2E 0x11 0xFC 0x46 0xE0
0x28 0xA6 0x49 0x58 0xC5 0x5E 0xFE 0x77 0x01 0xBA
0x07 0xB6 0x19 0x8C 0x3C 0x1D 0x1E 0xB7 0x63 0x5E
0x97 0xB0 0xCD 0x58 0x06 0x81 0x03 0xD5 0x64 0xDC
0x36 0xA2 0x26 0xCB 0x2B 0xC5 0xE6 0x5E 0x16 0xCB
0xB3 0x19 0xB2 0xFB 0x3C 0x39 0x3B 0x8E 0xCF 0xF1
0x79 0x06 0x61 0x4D 0x67 0xFF 0xF0 0xFB 0x86 0xC5
0x8E 0x61 0x8D 0x74 0x8F 0x52 0x7B 0x0C 0x75 0xC6
0x85 0x84 0x0D 0x09 0xC2 0xCA 0xF1 0xDB 0x18 0xC2
0x43 0x6F 0xE9 0x11 0x37 0x00]

$K_2=$ [0x5C 0xD5 0x0D 0xFB 0x4F 0x8A 0x55 0x31 0x1C 0xF3
0xCC 0xBD 0xF0 0xA4 0xD5 0x80 0x5D 0xAA 0x0B 0x2E
0x98 0xDE 0x8E 0x09 0x1F 0x82 0x04 0xBA 0x39 0x29
0x7C 0x78]

$IV_2=$ [0xD9 0x9D 0x22 0x35 0x4E 0xF7 0x15 0xF8 0x70 0x88
0xEF 0x8E 0x88 0xBE 0xC0 0x1C 0x6A 0xD7 0xFE 0xDF
0x43 0xF7 0x8D 0x61 0x5D 0x88 0xB9 0x00 0xCA 0x62
0x29 0xF0]

$A^*=$ [0xB9 0x55 0xF7 0x5C 0xB9 0x91 0xC3 0x17 0xD1 0xC4
0x2A 0x7D 0x7C 0x3A 0xC8 0x1E 0x84 0x62 0xF4 0x03
0x69 0x44 0x7F 0x20 0x6E 0xFB 0xF3 0x0E 0xD1 0x47
0x8A 0xD0 0xA4 0xA0 0x0C 0x00 0xA4 0x6B 0x84 0x71
0x14 0x14 0x70 0xF3 0xD3 0x4E 0x88 0xD7 0xF8 0xC3
0xFD 0xAE 0xAA 0x2A 0xA1 0x98 0xFC 0x07 0x87 0x74
0x7C 0x7D 0xBB 0x06 0x5E 0x56 0x1C 0x41 0x67 0x54
0x54 0xDF 0x1F 0x49 0x0A 0x1D 0x9B 0xE0 0x7E 0x05
0xF1 0x41 0xE9 0x2A 0x11 0x0E 0x91 0x87 0xB7 0xBA
0xA8 0x2F 0xBC 0x67 0x2B 0xEF]

A.3 Attack Vector for AEGIS-128L

$C\|\|T=$	[0xE2	0xF5	0x27	0xF6	0x7D	0xD5	0xC9	0x77	0x5C	0x0C
	0x0A	0x09	0x0C	0x06	0x71	0x5A	0x4F	0x78	0x84	0xF1
	0x2F	0x08	0xB8	0xF6	0x05	0xD4	0xED	0x86	0x89	0x52
	0x37	0xA0]								
$K_1=$	[0x09	0xAA	0x5D	0x16	0x70	0x62	0x2E	0xED	0xFB	0x18
	0x8E	0x9D	0x17	0xA9	0x71	0x18]				
$IV_1=$	[0x24	0xF2	0xEA	0xAF	0xAE	0xCA	0x95	0xFF	0xC8	0x4A
	0x3B	0x94	0x36	0x8C	0xD2	0xC1]				
$A_1=$	[0x92	0x9D	0xBF	0xD2	0x4E	0xAE	0x0A	0x2E	0xAC	0xB1
	0x1E	0x0F	0x82	0x28	0x1A	0x2D	0x4B	0x7F	0x15	0xF2
	0x32	0x53	0x7B	0xFC	0x00	0xDC	0x98	0x08	0xA8	0xF7
	0x57	0xA9	0xB5	0x38	0xF6	0x4E	0x0F	0xD1	0x6F	0x88
	0xD1	0x10	0x7D	0xE9	0x11	0x35	0x8C	0x27	0x24	0xDE
	0x8E	0x14	0xF5	0x51	0x21	0x0E	0xEB	0x90	0x95	0xB6
	0x4A	0xAC	0x7D	0x1D	0xF9	0xAE	0xC5	0xEA	0x99	0x06
	0xF5	0x0E	0x57	0x8A	0x8B	0xB5	0x64	0x3C	0x15	0x4C
	0xD0	0xC2	0xE3	0xE6	0x76	0x82	0xE6	0xDF	0x63	0xB4
	0x30	0x27	0xAE	0x13	0x94	0xD8	0x5D	0x16	0x6A	0x2E
	0x3B	0x7C	0x0B	0xB6	0xAA	0xB9	0x98	0x2C	0x03	0x44
	0xF0	0x98	0x54	0xB5	0x1A	0xBA	0x37	0xB6	0x51	0x70
	0xCC	0xDB	0x91	0xCA	0x36	0x65	0x45	0x08]		
$K_2=$	[0x1A	0x69	0x72	0xD1	0x60	0x38	0x0B	0xA9	0xD6	0x0D
	0x6A	0xF6	0x1E	0xCB	0xEA	0x75]				
$IV_2=$	[0x24	0xF2	0xEA	0xAF	0xAE	0xCA	0x95	0xFF	0xC8	0x4A
	0x3B	0x94	0x36	0x8C	0xD2	0xC1]				
$A^*=$	[0xB6	0x58	0x24	0xE0	0x6F	0x0E	0xA4	0x06	0x42	0x5A
	0xF9	0x9F	0x84	0x1D	0xBA	0x19	0x3A	0xAA	0x11	0xA5
	0xA1	0x09	0x72	0x02	0x85	0x9A	0x58	0xA2	0xDA	0x54
	0xED	0x2A	0x57	0xF3	0x7F	0x00	0xBD	0xB0	0x31	0x0B
	0x75	0xD5	0xCA	0xD0	0x3A	0x09	0x34	0x30	0x51	0xB9
	0xF1	0x74	0x80	0xF8	0x79	0x8A	0x10	0xA1	0x16	0x89
	0x40	0xCF	0xFC	0xDD	0x11	0x68	0xC2	0x22	0xF6	0xB5
	0xFB	0xA3	0xED	0x44	0x81	0x1B	0xDA	0xBC	0xB4	0x2E
	0xE3	0x52	0xA4	0x49	0x21	0xFD	0x9C	0x9F	0x41	0xF0
	0xB7	0xD8	0x77	0xB4	0x62	0x3D	0x79	0x61	0x69	0xE9
	0xD7	0x0A	0xA7	0x06	0x4C	0xD8	0x14	0xD8	0x9C	0xF1
	0x56	0x1A	0xA9	0x42	0x06	0xD2	0x6C	0x70	0x28	0x04
	0xE3	0xF4	0x11	0x14	0xC4	0x30	0x31	0x72]		