IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 4, pp. 1–38. DOI:10.46586/tosc.v2020.i4.1-38

LM-DAE: Low-Memory Deterministic Authenticated Encryption for 128-bit Security

Yusuke Naito[1], Yu Sasaki[2] and Takeshi Sugawara[3]

[1] Mitsubishi Electric Corporation, Kanagawa, Japan,
Naito.Yusuke@ce.MitsubishiElectric.co.jp
[2] NTT Secure Platform Laboratories, Tokyo, Japan, yu.sasaki.sk@hco.ntt.co.jp
[3] The University of Electro-Communications, Tokyo, Japan, sugawara@uec.ac.jp

Abstract. This paper proposes a new lightweight deterministic authenticated encryption (DAE) scheme providing 128-bit security. Lightweight DAE schemes are practically important because resource-restricted devices sometimes cannot afford to manage a nonce properly. For this purpose, we first design a new mode LM-DAE that has a minimal state size and uses a tweakable block cipher (TBC). The design can be implemented with low memory and is advantageous in threshold implementations (TI) as a side-channel attack countermeasure. LM-DAE further reduces the implementation cost by eliminating the inverse tweak schedule needed in the previous TBC-based DAE modes. LM-DAE is proven to be indistinguishable from an ideal DAE up to the $O(2^n)$ query complexity for the block size n. To achieve 128-bit security, an underlying TBC must handle a 128-bit block, 128-bit key, and 128+4-bit tweak, where the 4-bit tweak comes from the domain separation. To satisfy this requirement, we extend SKINNY-128-256 with an additional 4-bit tweak, by applying the elastic-tweak proposed by Chakraborti et al. We evaluate the hardware performances of the proposed scheme with and without TI. Our LM-DAE implementation achieves 3,717 gates, roughly 15% fewer than state-of-the-art nonce-based schemes, thanks to removing the inverse tweak schedule.

Keywords: Deterministic authenticated encryption · beyond-birthday-bound security · tweakable block cipher · lightweight · low-memory.

1 Introduction

The explosive increase in data communication through Internet of Things (IoT) devices has generated a high demand for lightweight authenticated encryption (AE) schemes that can be used comfortably in a resource-restricted environment. In particular, the National Institute of Standards and Technology (NIST) is organizing an ongoing standardization process for lightweight AE (NIST Lightweight Cryptography (LWC)) [NIS18]. Good designs for lightweight AE have been studied extensively, and there is a demand for 128-bit security to replace the conventional AES-GCM and AES-CCM with 64-bit security. NIST LWC explicitly requires better security than AES (p. 22 in [Sön19]), and many candidates have more than 64-bit security, including Romulus [IKMP20] and SKINNY-AEAD [BJK+19] with 128-bit security.

Some AE schemes need nonce, a value that must be processed only once under the same key; these schemes are called nonce-based AE. However, satisfying this requirement in implementation turned out to be difficult, and the community has encountered many security incidents caused by the inappropriate handling of nonces, e.g., a low-quality random number, a tiny nonce space, and even a constant nonce [BZD+16]. Indeed, the

Received: 2020-09-01 Accepted: 2020-11-01 Published: 2020-12-10

robustness against nonce misuse was one of the main objects in CAESAR [CAE19], which chose two algorithms for the portfolio.

Even if implementers are careful enough, extremely resource-constrained platforms lack critical components for realizing nonces. In particular, stateful nonce management (e.g., sequential counter) is necessary for 128-bit secure schemes with a 128-bit nonce because a collision in a random number degenerates it to the birthday-bound security; a random number generator is insufficient for these schemes. However, some devices do not have non-volatile memory or cannot write to it with a wireless power supply [Har08, AHM14].

As long as security relies on the nonce, the risk of misuse is inevitable. In contrast to nonce-based AE, deterministic AE (DAE) does not rely on nonces, and thus provides robustness against nonce misuse by construction. DAE is also useful for resource-constrained devices without a random number generator or a non-volatile memory. Although DAE needs a buffer for scanning the entire message twice, numerous IoT protocols limit the message length to several dozen bytes (e.g., 64 bytes for CAN FD) [ALP+19] that fit within a cheap non-volatile memory, which makes DAE a practical option. Consequently, there is an increasing number of lightweight DAE proposals, including SUNDAE [BBLT18], ANYDAE [CDD+19] and ESTATE [CDJ+19b]. Meanwhile, the NIST LWC candidates are still dominated by the nonce-based AEs because a nonce is mandatory in NIST LWC. Hence, studies on DAE and NIST LWC will complement each other.

Another line of research related to lightweight AE is an efficient countermeasure against side-channel attack (SCA). SCA is a severe concern in the main targets of lightweight cryptography, such as secure embedded devices, and the NIST's competition considers the grey-box security model with side-channel leakage, in addition to the conventional black-box security model.

Providing SCA resistance in resource-constrained devices is challenging because promising countermeasures, such as threshold implementation (TI), are based on multi-party computation that multiplies the memory size by the number of shares [ISW03, NRR06]. In particular, PFB [NS20] and PFB_Plus [NSS20] are nonce-based AEs optimized for TI: the schemes with a smaller block-length primitive, enabled by the beyond-birthday-bound (BBB) security for block size, are advantageous with TI because the number of shares can be smaller on the linearly updated state.

Our goal is to design a lightweight DAE that is 128-bit secure and suitable for TI. More specifically, we set the memory size as a primary target. Although a DAE needs a message buffer for storing the entire message for two-pass scanning, the low-memory property is still important because it is a key parameter determining the cost of a coprocessor implementation; we can use an efficient SRAM for the message buffer, while we need expensive registers for hardware implementation. The memory size must be at least 384 bits for 128-bit security; a 128-bit key for encryption and message authentication code (MAC), and 256 bits for the internal state to avoid the collision. Generally, to achieve s-bit security, the memory size must be at least $3s$ bits, which is our goal for the memory size of our DAE mode.

1.1 Choice of Primitive

Our design uses tweakable block cipher (TBC) as a primitive by considering the trade-offs summarized in Tables 1 and 2. Table 1 shows security, memory size, and speed for several parameters: the key size k, block size n, rate r, capacity c, counter size i, and tag size τ. Table 2 shows the same performances with the security level denoted by s. The memory sizes with TI in Table 1 and 2 are obtained by multiplying the target memory sizes depending on the relevant operations: the non-linearly and linearly processed states are multiplied by 3 and 2, respectively. We assume a linear key and tweak schedule considering SKINNY as a concrete example.

- Block cipher (BC): For the security level s, SUNDAE, ANYDAE, and ESTATE achieve the minimum $3s$-bit memory. However, the block size should be $2s$ bits for the birthday-bound security; these schemes need an uncommon lightweight 256-bit BC[1] to achieve 128-bit security. Moreover, a large block-size primitive has a disadvantage with TI because the 256-bit state is duplicated into several shares.

- Permutation: In a permutation-based DAE such as HADDOC [BDP+14], the state should be larger than 256 bits because we need a 256-bit capacity for 128-bit security and the rate larger than 1. It also requires extra memory for storing a tag and a counter to achieve the sponge-based counter mode in encryption. Similarly to BC-based DAE, the large block size has a disadvantage with TI. Duplex [BDPA11] is a core component of sponge-based AEs. For deterministic schemes via Duplex, the security level is $c/2$ bits [BDPA11, ADMA15, MRV15, DMA17], where c is a capacity. Hence, achieving 128-bit security requires at least $(384 + r)$-bit memory ($(256 + r)$-bit permutation and 128-bit key), where r is a rate.[2]

- TBC: TBC provides nonce-based low-memory AEs such as PFB, Romulus, and PFB_Plus. Moreover, these AEs have BBB-security for block size, which ensures low-memory in TI. In contrast, the conventional TBC-based DAEs such as ZAE [IMPS17] aim at speed instead of memory size, and none achieves the minimum memory size. We thus design a TBC-based DAE with minimum memory and with BBB-security for block size.

1.2 Technical Challenges

Low-Memory Domain Separation A BC-based mode SUNDAE achieves the minimal state size but requires additional primitive calls and the doubling operation for the domain separation, which is not preferable with respect to the memory size. TBC-based modes can embed the domain separation inside the tweak, which increases a tweak size by just a few bits from the minimal requirement. However, existing TBC designs cannot handle these extra tweak bits efficiently. For example, in Romulus, the tweak size except for the domain separation is 128 bits, while it instantiates a 256-bit tweak version instead of a 128-bit tweak version of the underlying TBC in order to process the extra 4 bits of tweak. Although there is an implementation trick to ignore the unused part in tweakey for SKINNY [IKMP20], it cannot be used for any TBC, and the trick makes the implementation more complicated for the necessary simulation of the missing part. ESTATE presents a method called *elastic tweak framework* [CDJ+19a] to convert a BC into a TBC with a few tweak bits, and uses such TBC in the SUNDAE mode. This modifies the primitive and requires a new security evaluation.

Inverse Tweak and Key Schedule BC- and TBC-based schemes demand the same secret key for each primitive call, so we should recover the original states after making on-the-fly tweak and key schedules. To achieve this without using extra memory for preserving the original key and tweak, the conventional Romulus and PFB implementations used an additional circuit for reverting the final tweakey state to the original one at the end of a BC/TBC call. For further optimization, TGIF [IKM+19] adopted a tweakey schedule that becomes the identity map after the encryption by 72 rounds. However, TGIF's schedule

[1] There are several choices. Saturnin [CDL+19] aims at post-quantum security and supports a 256-bit block. However, the key size is 256 bits. Rijndael [DR00] has a 256-bit block and 128-bit key version, but we aim to be smaller than AES.

[2] Transform-then-Permute construction [CJN20], which is a sponge-based AE mode, achieves $\min\{(r + c)/2, c/\log r\}$-security but is a nonce-based AE.

Table 1: Security, memory size with and without TI, and speed, where k: key size; n: block size of (T)BC; t: tweak size of TBC; r: rate; c: capacity; τ: tag size; i: counter size. Note that r, c, τ and i are the parameters of HADDOC, and $r + c$ is the permutation size. This table assumes that $t \geq n$. The entries for speed show block sizes in bits for each primitive call in hashing phases.

Name	Security	Memory Size w/o TI	Memory Size w/ TI	Speed [bit/call]	Ref.
LM-DAE	n	$n + t + k$	$3n + 2t + 2k$	n	Ours
SUNDAE	$n/2$	$n + k$	$3n + 2k$	n	[BBLT18]
ESTATE	$n/2$	$n + k$	$3n + 2k$	n	[CDJ+19b]
ZAE	n	$4n + 2t + k$	$9n + 4t + 2k$	$n + t$	[IMPS17]
HADDOC	$c/2$	$r + c + \tau + i + k$	$3(r + c) + \tau + i + 2k$	$r + c$	[BDP+14]

Table 2: Memory size with and without TI, and speed, when the security levels are fixed to s. We set the parameters to $k = t = n = s$ for TBC-based DAEs; $k = s$ and $n = 2s$ for BC-based DAEs; $k = s$, $c = 2s$ and $\tau = 2s$.

Name	Memory Size w/o TI	Memory Size w/ TI	Speed [bit/call]	Ref.
LM-DAE	$3s$	$7s$	s	Ours
SUNDAE	$3s$	$8s$	$2s$	[BBLT18]
ESTATE	$3s$	$8s$	$2s$	[CDJ+19b]
ZAE	$7s$	$15s$	$2s$	[IMPS17]
HADDOC	$5s + r + i$	$10s + 3r + i$	$r + 2s$	[BDP+14]

forms a cycle in every 24 rounds, which may yield a security concern, e.g. against slide attacks[3].

1.3 Our Contribution

Novelty of the Modes. As discussed above, TBC-based design seems to be a promising approach for designing a low-memory 128-bit secure DAE because the BC-based DAEs require a 256-bit block while the existing BCs are not low memory or support only a 256-bit key, and the permutation-based DAE HADDOC requires a large state. In addition, if the design structure follows SPN, a 4-bit S-box is suitable to be lightweight. In general, a 4-bit S-box based 256-bit block BC/permutation that is lightweight particularly for hardware implementations is more difficult to design than a 4-bit S-box based 128-bit block TBC under the same goal. Hence, we design a DAE mode on the basis of a TBC.

We first define a state update function that is a building block for low-memory DAE modes and that updates a $2n$-bit state by using a TBC, which is shown in Fig. 1. The function takes as input a data block (associated data or plaintext) of up to $2n$ bits $(D_{i,1}, D_{i,2})$, which are xored to the $2n$-bit state. Then, the state is updated by a TBC and a linear function, where a TBC takes an input block X_i and a tweak (d_i, Y_i) where d_i is for the domain separation. By including the tweak schedule of $\tilde{E}$ in the linear function, i.e., $L_{1,2} \cdot Y_i$ and $L_{2,2} \cdot Y_i$ are defined to include the result of the tweak schedule with Y_i, the subsequent linear function can start immediately without computing the inverse of the tweak schedule.

In the state update function, the size of the input data block can be up to $2n$ bits. However, the maximum size, or maximum rate, to achieve n-bit security is unknown. In Section 4, we demonstrate an attack with a complexity of $O(2^{n/2})$ when the size of the input data block is at least $n + 1$ bits. That is, the size of the input is at most n bits to achieve n-bit security.

[3]Besides the issue of the tweakey-schedule structure, TGIF is not suitable for LM-DAE that requires a tweakey of at least 256 bits for 128-bit security.

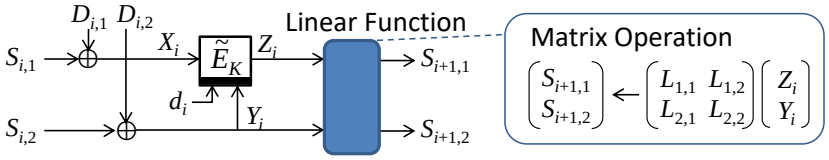

Figure 1: State Update Function SUF.

In Section 5, we present our low-memory DAE mode called LM-DAE by iteratively applying the state update function with $L_{1,1} = 1$, $L_{1,2} = 0$ and $L_{2,1} = 1$ ($L_{2,2}$ is defined to include the result of the tweak schedule), which only requires a minimal state size of $2n$ bits without the inverse of the tweak schedule. The tweak size for the domain separation in LM-DAE is 4 bits. We represent the state update by the tweak schedule by an n-bit permutation π and then prove that for any permutation π (that is, the proof covers non-linear tweak schedules as well as linear tweak schedules), LM-DAE is an n-bit secure DAE, which is indistinguishable from an ideal DAE (random-bit and reject oracles).

Although the structure of LM-DAE is more complex than SUNDAE, it is a result of the rigorous optimization across the boundaries between the mode, primitive, and implementation. We believe that LM-DAE's several benefits are worth the cost, in particular, TI and block size.

Novelty of TBC. Our scheme needs a lightweight TBC with 128-bit block, a 128-bit key, and 132-bit tweak. In this work, we use SKINNY-128-256 as a base and modify it by applying the elastic-tweak framework to handle an additional 4-bit tweak efficiently. The design is called TweSKINNY-128-256.

We first improve the elastic-tweak framework that will be explained in Section 8. The improved framework is then applied to SKINNY-128-256. We tested several design choices by evaluating the number of active S-boxes using Mixed Integer Linear Programming (MILP). Besides, the number of rounds is chosen independently of SKINNY-128-256 to minimize the hardware footprint. Our idea of π that removes the inverse of the tweak schedule cannot be applied to the key, because the key needs to be the same for all TBC invocations. We observe that the tweakey schedule of SKINNY makes a cycle in every 16 rounds about the byte position. Hence, the hardware footprint is optimized when the number of rounds is a multiple of 16. Considering the security margin, we choose 48 rounds. This happens to match the number of rounds of SKINNY-128-256 that is chosen purely for the security margin.

Tweakey-schedule Optimization. We eliminate the inverse tweakey schedule to improve the implementation performance. We achieve this goal without resorting to a cyclic tweakey schedule by instantiating the proposed mode with a particular π and TBC: we address one half of the problem by a primitive and the other half by a mode. We observed that, with an appropriate round number, SKINNY's TK1 comes back to the original value while TK2 does not because of the linear feedback shift register (LFSR). Thus, we assign the key into TK1 and let LM-DAE handle the modified tweak in TK2: LM-DAE integrates the TK2 schedule into the linear function in Fig. 1 so that we can carry the final TK2 state to the next block processing without losing the provable security.

Hardware Performance Evaluation. We implement LM-DAE with TweSKINNY in hardware with and without TI and compare them with state-of-the-art nonce-based AEs, namely PFB and PFB_Plus. Without TI, LM-DAE achieved 3,717 gates, which is roughly 15% fewer than PFB and PFB_Plus with the same $3s$ memory thanks to the tweakey-schedule

optimization. With TI, this advantage is preserved, and LM-DAE achieved 8,358 gates comparable to PFB with a smaller register size.

1.4 Related Works

AE with Higher Data Rate. As LM-DAE prioritizes security and memory size in the design trade-off, the other DAE schemes have advantages in terms of speed, i.e., rate, as summarized in Table 1. By fixing security level s, SUNDAE and ESTATE consume a $2s$-bit block for each TBC call, which is higher than LM-DAE's s bits. ZAE accepts a $(s+t)$ block for MAC wherein t is a tweak size, and HADDOC uses its entire $(r+2s)$-bit state for consuming input data[4].

AE with Nonce-Misuse Resistance. There are nonce-based AEs, such as SCT [PS16] and Romulus-M1 [IKMP20], that become DAEs when a nonce is fixed. However, they are inefficient as a DAE because of the security level degenerated by the misuse. SCT and Romulus-M1 are secure up to the birthday bound, requiring memory sizes greater than $3s$ bits to achieve s-bit security.

1.5 Outline

The rest of this paper is organized as follows. We define basic notations in Section 2. We introduce the state-update function and its security analysis in Sections 3 and 4. Then, we describe the proposed LM-DAE in Section 5, followed by its security proofs in Sections 6 and 7. Section 8 is devoted to the new primitive TweSKINNY-128-256. Finally, Section 9 describes hardware implementation.

2 Preliminaries

Notation. Let ε be an empty string and $\{0,1\}^*$ be the set of all bit strings. For an integer $i \geq 0$, let $\{0,1\}^i$ be the set of all i-bit strings, $\{0,1\}^0 := \{\varepsilon\}$, and $\{0,1\}^{\leq i} := \{0,1\}^1 \cup \{0,1\}^2 \cup \cdots \cup \{0,1\}^i$ be the set of all bit strings of length at most i, except for ε. Let 0^i resp. 1^i be the bit string of i-bit zeros resp. ones. For integers $0 \leq i \leq j$, let $[i,j] := \{i, i+1, \ldots, j\}$, $(j] := [0,j]$ and $[j] := [1,j]$. For integers $0 \leq i \leq x$, let $(x)_i := x(x-1)\cdots(x-j+1)$ be the falling factorial. For a non-empty set $\mathcal{T}$, $T \xleftarrow{\$} \mathcal{T}$ means that an element is chosen uniformly at random from $\mathcal{T}$ and is assigned to T. The concatenation of two bit strings X and Y is written as $X\|Y$ or XY when no confusion is possible. For integers $0 \leq i \leq j$ and $X \in \{0,1\}^j$, let $\mathsf{msb}_i(X)$ (resp. $\mathsf{lsb}_i(X)$) be the most (resp. least) significant i bits of X, and $|X|$ be the bit length of X, i.e., $|X| = j$. For an integer $n \geq 0$ and a bit string X, we denote the parsing into fixed-length n-bit strings as $(X_1, \ldots, X_\ell) \xleftarrow{n} X$, where if $X \neq \varepsilon$ then $X = X_1\| \cdots \|X_\ell$, $|X_i| = n$ for $i \in [\ell-1]$, and $0 < |X_\ell| \leq n$; if $X = \varepsilon$ then $\ell = 1$ and $X_1 = \varepsilon$. For an integer $n > 0$, let $\mathsf{ozp} : \{0,1\}^{\leq n} \to \{0,1\}^n$ be a one-zero padding function: for $X \in \{0,1\}^{\leq n}$, $\mathsf{ozp}(X) = X$ if $|X| = n$; $\mathsf{ozp}(X) = X\|10^{n-1-|X|}$ if $|X| < n$. For non-empty sets $\mathcal{X}, \mathcal{Y}$, let $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ be a set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

A TBC is a set of permutations indexed by a key and a public input called tweak, and a tweakable permutation (TP) is a TBC such that the key space is empty. Let $\mathcal{K}$ be the key space, $\mathcal{TW}$ the tweak space, and n the input/output-block size. A TBC (encryption) is denoted by $\widetilde{E} : \mathcal{K} \times \mathcal{TW} \times \{0,1\}^n \to \{0,1\}^n$, $\widetilde{E}$ taking a key $K \in \mathcal{K}$ is denoted by $\widetilde{E}_K$, and $\widetilde{E}_K$ taking a tweak $TW \in \mathcal{TW}$ is denoted by $\widetilde{E}_K^{TW}$. A TP (encryption) is denoted by $\widetilde{P} : \mathcal{TW} \times \{0,1\}^n \to \{0,1\}^n$. Let $\widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^n)$ be the set of all TPs.

[4] ZAE (resp. HADDOC) consume s (resp. r) bits in the encryptions.

For a security goal xxx, a target scheme Π, an adversary $\mathbf{A}$, and the advantage function $\mathsf{Adv}^{\mathsf{xxx}}_{\Pi}(\mathbf{A})$, the maximum advantage over all adversaries, running in time at most t and having parameters for an adversary's queries $\mathcal{Q}$ (such as the number of queries, query lengths, etc.), is denoted by $\mathsf{Adv}^{\mathsf{xxx}}_{\Pi}(\mathcal{Q},t) := \max_{\mathbf{A}} \mathsf{Adv}^{\mathsf{xxx}}_{\Pi}(\mathbf{A})$. When an adversary is a computationally unbounded algorithm, the time t is disregarded. For an adversary $\mathbf{A}$ with access to $\mathcal{O}$ and after the interaction returning a decision bit, the output of $\mathbf{A}$ is denoted by $\mathbf{A}^{\mathcal{O}}$.

TPRP. Though this paper, a keyed TBC is assumed to be a secure tweakable-pseudo-random permutation (TPRP). In the tprp-security game, an adversary $\mathbf{A}$ has access to either the keyed TBC $\widetilde{E}_K$ or a TRP $\widetilde{P}$, where $K \xleftarrow{\$} \mathcal{K}$ and $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW},\{0,1\}^n)$, and after the interaction, $\mathbf{A}$ returns a decision bit $y \in \{0,1\}$. The advantage function is defined as

$$\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\mathbf{A}) := \Pr[\mathbf{A}^{\widetilde{E}_K} = 1] - \Pr[\mathbf{A}^{\widetilde{P}} = 1] \ ,$$

where the probabilities are taken over K, $\widetilde{P}$, and $\mathbf{A}$.

TBC-based DAE. A DAE scheme using a keyed TBC $\widetilde{E}_K$, denoted by $\Pi[\widetilde{E}_K]$, is a pair of encryption and decryption algorithms $(\Pi.\mathsf{Enc}[\widetilde{E}_K], \Pi.\mathsf{Dec}[\widetilde{E}_K])$. $\mathcal{M},\mathcal{C},\mathcal{A}$ and $\mathcal{T}$ are sets of plaintexts, ciphertexts, associated data (AD), and tags of $\Pi[\widetilde{E}_K]$, respectively. The encryption algorithm takes AD $A \in \mathcal{A}$ and a plaintext $M \in \mathcal{M}$ and returns, deterministically, a pair of a ciphertext $C \in \mathcal{C}$ and a tag $T \in \mathcal{T}$. The decryption algorithm takes a tuple $(A,C,\hat{T}) \in \mathcal{A}\times\mathcal{C}\times\mathcal{T}$, and deterministically returns either the distinguished invalid symbol $\mathbf{reject} \notin \mathcal{M}$ or a plaintext $M \in \mathcal{M}$. We require $|\Pi.\mathsf{Enc}[\widetilde{E}_K](A,M)| = |\Pi.\mathsf{Enc}[\widetilde{E}_K](A,M')|$ when these outputs are strings and $|M| = |M'|$. Through this paper, we call queries to the encryption resp. decryption oracle "encryption queries" resp. "decryption queries."

We follow the security definition given by Rogaway and Shrimpton [RS06] that is the indistinguishability between $\Pi[\widetilde{E}_K] = (\Pi.\mathsf{Enc}[\widetilde{E}_K], \Pi.\mathsf{Dec}[\widetilde{E}_K])$ and $(\$,\bot)$, where $\$$ is a random-bits oracle that has the same interface as $\Pi.\mathsf{Enc}[\widetilde{E}_K]$ and for a query (A,M) returns a random bit string of length $|\Pi.\mathsf{Enc}[\widetilde{E}_K](A,M)|$; $\bot$ is an oracle that returns **reject** for any query. In the dae-security game, first an adversary $\mathbf{A}$ interacts with either $\Pi[\widetilde{E}_K]$ or $(\$,\bot)$ where $K \xleftarrow{\$} \mathcal{K}$ and then returns a decision bit $b \in \{0,1\}$. The advantage function is defined as

$$\mathbf{Adv}^{\mathsf{dae}}_{\Pi[\widetilde{E}_K]}(\mathbf{A}) = \Pr[\mathbf{A}^{\Pi[\widetilde{E}_K]} = 1] - \Pr[\mathbf{A}^{\$,\bot} = 1] \ ,$$

where the probabilities are taken over $K,\$$ and $\mathbf{A}$. We demand that $\mathbf{A}$ never asks a trivial decryption query (A,C,T), i.e., there is a prior encryption query (A,M) with $(C,T) = \Pi.\mathsf{Enc}[\widetilde{\mathbf{E}}_K](A,M)$, and that $\mathbf{A}$ never repeats a query.

PRF. Let $\mathcal{X}$ and $\mathcal{Y}$ be non-empty sets, and $F[\widetilde{E}_K] : \mathcal{X} \to \mathcal{Y}$ a function using a keyed TBC $\widetilde{E}_K$. The prf-(pseudo-random-function) security of $F[\widetilde{E}_K]$ is indistinguishability between real $(F[\widetilde{E}_K])$ and ideal (a random function $\mathcal{R}$) worlds. In the prf-security game, first an adversary $\mathbf{A}$ interacts with either $F[\widetilde{E}_K]$ or $\mathcal{R}$ where $K \xleftarrow{\$} \mathcal{K}$ and $\mathcal{R} \xleftarrow{\$} \mathsf{Func}(\mathcal{X},\mathcal{Y})$, and then returns a decision bit $y \in \{0,1\}$. The advantage function is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{F[\widetilde{E}_K]}(\mathbf{A}) := \Pr[\mathbf{A}^{F[\widetilde{E}_K]} = 1] - \Pr[\mathbf{A}^{\mathcal{R}} = 1] \ ,$$

where the probabilities are taken over $K,\mathcal{R}$ and $\mathbf{A}$.

3 A Tool for Low-Memory TBC-based DAE

In this section, we propose a state update function based on a TBC that is a building block of low-memory DAE modes. The state update function is defined to take a data block up to $2n$ bits long for a TBC with n-bit blocks. Before designing a DAE using the state update function, we need to find the maximum length of a data block for achieving n-bit security. In Section 4, we show that the length is at most n bits. Then, in Section 5, following the result, we define our DAE mode, which iteratively uses the state update function taking a data block of n bits, and show the security bound: our DAE mode achieves n-bit security. The security proof is given in Section 6.

3.1 Internal State and Tweak Sizes

We first specify an internal state and tweak sizes for a low-memory n-bit secure DAE, because the memory size of DAE largely depends on these sizes.

- **Internal State Size.** An internal state collision offers an output collision for a DAE, thereby yielding a distinguishing attack from an ideal system $(\$, \perp)$. Due to the birthday paradox, the minimum size is $2n$ bits for n-bit security.
- **Tweak Size.** Basically, TBC-based DAE modes such as ZAE [IMPS17] use small bits in the tweak space for the domain separation. Hence, we define the tweak size as $t+\delta$ bits where δ is a small constant, e.g., $\delta = 4$, for the domain separation and t is a tweak size for handling an internal state. Regarding the tweak size t, to achieve n-bit security, it must be $t \geq n$, because a collision of the input to a TBC (the complexity is $O(2^{(t+n)/2})$) becomes a bad event in a security proof. On the other hand, if $t > n$, the TBC requires an additional $t-n$-bit memory beyond the memory for the $2n$-bit internal state. Hence, the tweak size should be $t = n$, which implies that the tweak size is $n+\delta$ bits.

Tweaks for domain separation. Generally, domain separations separate the domain of the following computations: (i) data block processing in a tag generation, (ii) a finalization in a tag generation, (iii) encryption (decryption), and whether or not a padding is applied to the data block for (i) and (iii). In the state update function defined in the next subsection, the δ-bit tweak space is used to separate these TBC calls.

3.2 TBC-based State Update Function

We define a state update function shown in Fig. 1 that uses a TBC with n-bit block and $n+\delta$-bit tweak spaces, takes a $2n$-bit (previous) internal state and a data block of length at most $2n$ bits, and returns the next $2n$-bit internal state. By using the state update function iteratively, one can obtain a low-memory DAE.

The state update function $\mathsf{SUF}[\widetilde{E}_K]$ is composed of a TBC and linear operations and takes data blocks $D_{i,1}$ and $D_{i,2}$. Let $\widetilde{E}_K : (\{0,1\}^\delta \times \{0,1\}^n) \times \{0,1\}^n \to \{0,1\}^n$ be the underlying TBC with a secret key K. For each $j \in [2]$, let n_j be the length in bits of a data block $D_{i,j}$ such that $0 \leq n_j \leq n$. For an internal state $(S_{i,1}, S_{i,2}) \in \{0,1\}^n \times \{0,1\}^n$ and data blocks $(D_{i,1}, D_{i,2}) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$, the next internal state $(S_{i+1,1}, S_{i+1,2}) \in \{0,1\}^n \times \{0,1\}^n$ is defined as follows.

- $\mathsf{SUF}[\widetilde{E}_K]((S_{i,1}, S_{i,2}), (D_{i,1}, D_{i,2}))$:
  1. The following operations are performed:

$$X_i \leftarrow S_{i,1} \oplus D_{i,1}; Y_i \leftarrow S_{i,2} \oplus D_{i,2}; Z_i \leftarrow \widetilde{E}_K^{d_i, Y_i}(X_i)$$

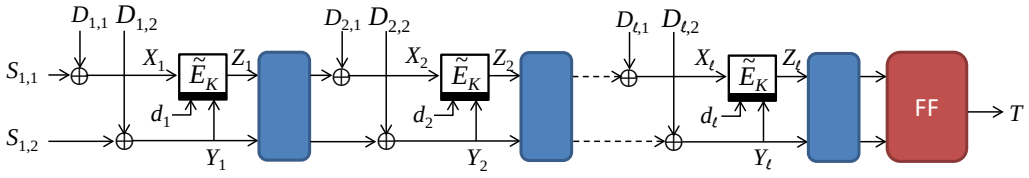

Figure 2: Tag Generation Function $\mathsf{TagGen}[\widetilde{E}_K]$.

where $d_i \in \{0,1\}^\delta$ is a tweak for domain separation, and for the XOR operation $S_{i,j} \oplus D_{i,j}$ where $j \in [2]$, $D_{i,j}$ in the n-bit space is assigned to any n_j-bit-position and $(n - n_j)$-bit zeros are padded in the remaining space.

2. The next internal state is defined as

$$\begin{pmatrix} S_{i+1,1} \\ S_{i+1,2} \end{pmatrix} \leftarrow \begin{pmatrix} L_{1,1} & L_{1,2} \\ L_{2,1} & L_{2,2} \end{pmatrix} \cdot \begin{pmatrix} Z_i \\ Y_i \end{pmatrix}$$

where $L_{i,j} \in \{0,1\}^n$ and the matrix operation is done by linear operations over $GF(2^n)$.

3. **Return** $(S_{i+1,1}, S_{i+1,2})$.

We design an n-bit secure DAE using the state update function $\mathsf{SUF}[\widetilde{E}_K]$. Before giving our DAE, in Section 4, we show an upper bound of the length $n_1 + n_2$ to achieve n-bit security. In Section 5, we define our DAE in accordance with the upper bound.

4 Upper Bound of Data Block Size for n-bit Security

We specify an upper bound of length $n_1 + n_2$ to achieve n-bit security. We show that the upper bound is n bits, i.e., we give an $O(2^{n/2})$ attack on a DAE that iteratively uses $\mathsf{SUF}[\widetilde{E}_K]$ defined in Section 3 when $n_1 + n_2 \geq n + 1$.

4.1 Target DAE

We first split the encryption algorithm into two algorithms: tag generation algorithm $\mathsf{TagGen}[\widetilde{E}_K]$, and ciphertext generation algorithm $\mathsf{CTGen}[\widetilde{E}_K]$:

$$\mathsf{TagGen}[\widetilde{E}_K] : \mathcal{A} \times \mathcal{M} \to \mathcal{T}, \qquad \mathsf{CTGen}[\widetilde{E}_K] : \mathcal{A} \times \mathcal{M} \to \mathcal{C},$$

where $\mathcal{A} = \{0,1\}^*$, $\mathcal{M} = \mathcal{C} = \{0,1\}^*$, $\tau \geq n$ and $\mathcal{T} = \{0,1\}^\tau$. Hence, the encryption algorithm of DAE is defined as follows.

- $\Pi.\mathsf{Enc}[\widetilde{E}_K](A, M)$:
  - $C \leftarrow \mathsf{CTGen}[\widetilde{E}_K](A, M)$; $T \leftarrow \mathsf{TagGen}[\widetilde{E}_K](A, M)$; **return** (C, T)

In our analysis, for DAE schemes that iteratively use $\mathsf{SUF}[\widetilde{E}_K]$, we define an adversary that makes only encryption queries with non-empty AD and empty plaintexts. Hence, the goal of the adversaries is to distinguish between $\mathsf{TagGen}[\widetilde{E}_K]$ and a random function, i.e., breaking the prf-security of $\mathsf{TagGen}[\widetilde{E}_K]$.

We define the procedure of $\mathsf{TagGen}[\widetilde{E}_K]$. For simplicity, the length of each AD is a multiple of the data block length $n_1 + n_2$. Let $\mathsf{FF} : \{0,1\}^{2n} \to \{0,1\}^\tau$ be a finalization function. Let $d_1, \ldots, d_\ell$ be tweaks for domain separation and constant values.[5] Then, the tag generation algorithm $\mathsf{TagGen}[\widetilde{E}_K]$ is defined as follows and shown in Fig. 2.

[5] The reason the tweaks are fixed is that for each AD block a padding function is not applied, and in such case, (to our knowledge) in all existing TBC-based AE schemes, the tweaks are fixed.

- $\mathsf{TagGen}[\widetilde{E}_K]((D_{1,1}\|D_{1,2}\|\cdots\|D_{\ell,1}\|D_{\ell,2}),\varepsilon)$: // ε is an empty plaintext.
  1. $S_{1,1}$ and $S_{1,2}$ are initialized to n-bit constant values.
  2. **for** $i = 1,\ldots,\ell$ **do** $(S_{i+1,1}, S_{i+1,2}) \leftarrow \mathsf{SUF}[\widetilde{E}_K]((S_{i,1}, S_{i,2}), (D_{i,1}, D_{i,2}))$
  3. **return** $T \leftarrow \mathsf{FF}(S_{\ell+1,1}, S_{\ell+1,2})$

4.2 Upper Bound for n-bit Security: $n_1 + n_2 \leq n$

Theorem 1. *Let* $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^n)$ *and* FF *a random function. For the tag generation function* $\mathsf{TagGen}[\widetilde{P}]$ *using* $\widetilde{P}$ *and* FF *such that* $n_1 + n_2 \geq n + 1$*, there exist adversaries* $\mathbf{A}$ *making* $O(2^{n/2})$ *queries such that*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{TagGen}[\widetilde{P}]}(\mathbf{A}) = \Omega(1) \ .$$

4.3 Proof of Theorem 1 (Attacks when $n_1 + n_2 \geq n + 1$)

We prove Theorem 1 by demonstrating the attacks with a complexity of $O(2^{n/2})$ when the data block size $n_1 + n_2$ is at least $n + 1$ bits.

The attack goal is a forgery. Overall, we first generate a state collision: two different data D_1 and D_2 that result in the same $2n$-bit state. Then for any common suffix D_{suffix}, the tag for $D_1\|D_{\text{suffix}}$ and $D_2\|D_{\text{suffix}}$ will collide.

The attack target satisfies $n_1+n_2 \geq n+1$. From this nature, at least one of $n_1 \geq n/2+1$ or $n_2 \geq n/2+1$ holds. Besides, both of $n_1 \geq 1$ and $n_2 \geq 1$ hold. The attack procedure depends on which of n_1 and n_2 is larger than or equal to $n/2+1$ bits. We study each case.

4.3.1 Case 1: $n_1 \geq n/2 + 1$

We inject the difference from the ith block and to cancel it in the $i+1$th block. The analysis is further divided into several cases depending on whether $L_{1,1}, L_{1,2}, L_{2,1}$ and $L_{2,2}$ are zero or non-zero. To be secure, during the matrix operation, each of two n-bit inputs (resp. outputs) must contribute to at least one of the two n-bit outputs (resp. inputs). This limits the valid pattern of $L_{1,1}, L_{1,2}, L_{2,1}, L_{2,2}$ to be the seven cases depicted in Fig. 3. The analysis is further divided into three cases: $L_{2,1} = 0$, $L_{1,1} = 0$, and $L_{2,1} \neq 0 \wedge L_{1,1} \neq 0$.

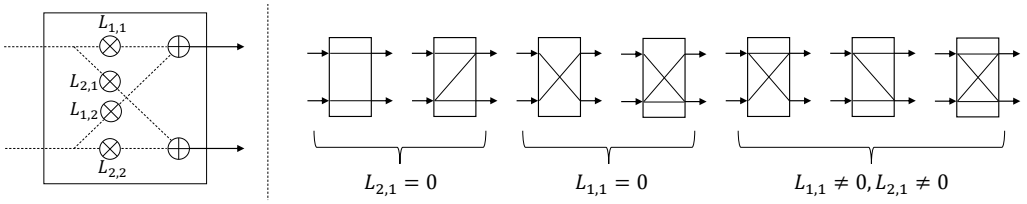

Figure 3: Seven Patterns of Linear Computations.

$\boldsymbol{L_{2,1} = 0}$**.** When $L_{2,1} = 0$, Z_i does not impact $S_{i+1,2}$. We set $\Delta D_{i,1} \neq 0$ and $\Delta D_{i,2} = 0$, then the difference is propagated as depicted in Fig. 4. It is ensured that $\Delta Z_i \neq 0$. Because of $L_{2,1} = 0$, ΔZ_i does not impact $S_{i+1,2}$, thus by setting $\Delta D_{i+1,1} \neq 0$ and $\Delta D_{i+1,2} = 0$, all the differences can be canceled probabilistically. Let $\delta := \Delta D_{i+1,1}$. Then if ΔZ_i is equal to $\delta/L_{1,1}$, the difference is canceled. Because of $n_1 \geq n/2 + 1$, the attacker can choose $2^{n/2}$ distinct values of $D_{i,1}$. By making $2^{n/2}$ queries, there should exist one pair that achieves $\Delta Z_i = \delta/L_{1,1}$, hence the construction is attacked. For the completeness, the algorithmic description of the attack is given in Alg. 1.

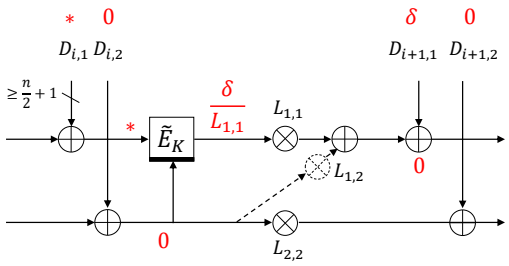

Figure 4: Differential Propagation for Case 1 with $L_{2,1} = 0$.

Algorithm 1 Attack Procedure for Case 1 with $L_{2,1} = 0$.

1: Choose any two distinct values of $D_{i+1,1}$ denoted by z and z'.
2: Choose any value for $D_{i,2}$ and $D_{i+1,2}$ denoted by y and w, respectively.
3: **for** $j = 1, 2, \ldots, 2^{n/2}$ **do**
4: Choose a value of $D_{i,1}$ denoted by x_j that is different from other x_j.
5: Query $x_j \| y \| z \| w$ and $x_j \| y \| z' \| w$ and store the corresponding output of the queries denoted by Q_j and Q'_j.
6: **end for**
7: Find a pair of two indices j_1 and j_2 such that $Q_{j_1} = Q'_{j_2}$.

$\boldsymbol{L_{1,1} = 0}$**.** The attack is almost the same as the case with $L_{2,1} = 0$. The only different point is that the difference in the $i+1$th block is injected from $D_{i+1,2}$ because ΔZ_i impacts $S_{i+1,2}$. Due to the similarity, we omit the details.

$\boldsymbol{L_{2,1} \neq 0}$ **and** $\boldsymbol{L_{1,1} \neq 0}$**.** In this case, we set $\Delta D_{i+1,1} \neq 0$ and $\Delta D_{i+1,2} \neq 0$. We expect ΔZ_i to satisfy $L_{2,1} \cdot \Delta Z_i = \Delta D_{i+1,2}$ and $L_{1,1} \cdot \Delta Z_i = \Delta D_{i+1,1}$ simultaneously. Then, all the differences are canceled. The differential propagation is depicted in Fig. 5.

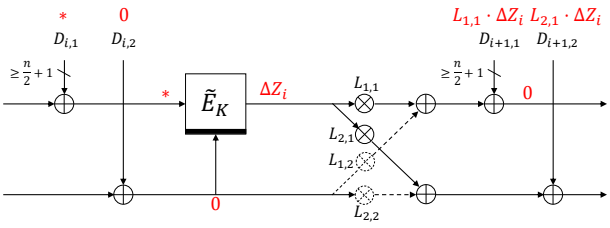

Figure 5: Differential Propagation for Case 1 with $L_{2,1} \neq 0 \wedge L_{1,1} \neq 0$.

Those imply that $\frac{\Delta D_{i+1,2}}{L_{2,1}} = \frac{\Delta D_{i+1,1}}{L_{1,1}}$, hence $\Delta D_{i+1,1} = \frac{L_{1,1}}{L_{2,1}} \cdot \Delta D_{i+1,2}$. The left-hand side is the linear space of dimension n_1 and the right-hand side is the linear space of dimension n_2, where $n_1 + n_2 \geq n + 1$, hence there always exists a particular choice of $\Delta D_{i+1,1}$ and $\Delta D_{i+1,2}$ that causes the difference cancellation. After appropriate $\Delta D_{i+1,1}$ and $\Delta D_{i+1,2}$ are chosen, the attack procedure is almost the same as the previous one. Due to the similarity, we omit the details.

4.3.2 Case 2: $n_2 \geq n/2 + 1$

Unlike in Case 1, $2^{n/2}$ different X_i may be impossible to prepare because n_1 can be very small, e.g., only 1 bit. Hence, a different attack strategy is required. The analysis again

depends on whether $L_{1,1}, L_{1,2}, L_{2,1}$, and $L_{2,2}$ are zero or non-zero. We do the case analysis for three cases: $L_{1,1} = 0$, $L_{1,1} \neq 0 \wedge L_{1,2} \neq 0$, and $L_{1,2} = 0$.

$L_{1,1} = 0$. In this case, the attacker injects a difference from $D_{i,1}$ and cancels it with a difference from $D_{i+1,2}$. The differential propagation is shown in Fig. 6, and the attack procedure is shown in Alg. 2. The attacker chooses $2^{n_1/2}$ values of $D_{i,1}$ and $2^{n_2/2}$ values of $D_{i+1,2}$ and makes queries of all combinations, thus the data complexity is $2^{(n_1+n_2)/2}$, which is $O(2^{n/2})$. Then we expect a state collision by the following analysis.

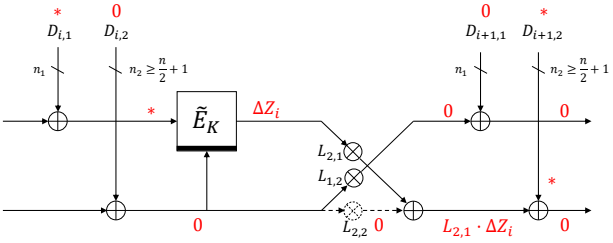

Figure 6: Differential Propagation for Case 2 with $L_{1,1} = 0$.

Without loss of generality, assume that the data for $D_{i+1,2}$ is xored to n_2 least significant bits (LSBs). Then $\Delta D_{i+1,2}$ cannot be added to $n - n_2$ most significant bits (MSBs). From $n_1 + n_2 \geq n + 1$, we have $n - n_2 \leq n_1 - 1$, i.e., the number of bits that cannot be canceled by $\Delta D_{i+1,2}$ is at most $n_1 - 1$ bits. By choosing $2^{n_1/2}$ values of $D_{i,1}$, we expect one pair with no difference in n_1 MSBs of $L_{2,1} \cdot Z_i$. For this pair, $2^{n_2/2}$ values of $D_{i+1,2}$ are queried, thus the difference in n_2 LSBs will be canceled in one pair. For the completeness, the algorithmic description of the attack is given in Alg. 2.

Algorithm 2 Attack Procedure for Case 2 with $L_{1,1} = 0$.

1: Choose any value for $D_{i,2}$ and $D_{i+1,1}$ denoted by y and z, respectively.
2: **for** $i = 1, 2, \ldots, 2^{n_1/2}$ **do**
3: Choose a value of $D_{i,1}$ denoted by x_i that is different from other x_i.
4: **for** $j = 1, 2, \ldots, 2^{n_2/2}$ **do**
5: Choose a value of $D_{i+1,2}$ denoted by w_j that is different from other w_j.
6: Query $x_i \| y \| z \| w_j$ and store the corresponding output of the query $Q_{i,j}$.
7: **end for**
8: **end for**
9: Find a quartet of four indices i_1, i_2, j_1 and j_2 such that $Q_{i_1,j_1} = Q'_{i_2,j_2}$.

$L_{1,1} \neq 0$ and $L_{1,2} \neq 0$. We first explain the most complex case in which all of $L_{1,1}$, $L_{1,2}$, $L_{2,1}$, and $L_{2,2}$ are non-zero. Let $\Delta D_{i,1}$ be any non-zero difference and let $\delta := \Delta D_{i,2}$. We cancel the difference with $\Delta D_{i+1,1} = 0$ and $\Delta D_{i+1,2} = \delta'$. The differential propagation is depicted in Fig. 7.

ΔY_i is δ. We expect that ΔZ_i will cancel the impact of ΔY_i to $S_{i+1,1}$, namely $L_{1,1} \cdot \Delta Z_i = L_{1,2} \cdot \delta$. This occurs with probability 2^{-n}. Then, the difference is canceled in the $i+1$th block if the following holds.

$$\left(L_{2,2} \oplus \frac{L_{1,2} \cdot L_{2,1}}{L_{1,1}}\right) \cdot \delta = \delta'. \tag{1}$$

For simplicity, we denote the coefficient in Eq. (1) by L.

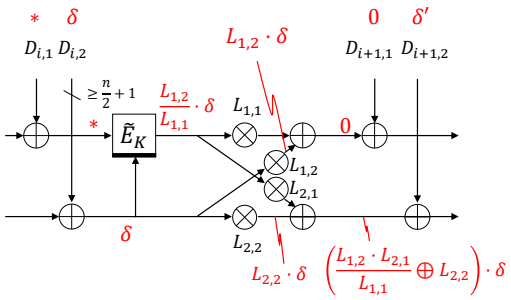

Figure 7: Attack against Case 2 where all of $L_{1,3}, L_{1,4}, L_{2,3}$ and $L_{2,4}$ are non-zero.

The number of choices of such δ and δ' depends on the parameter n_2. Considering that $n_2 \geq n/2+1$, the number of choices is at least $2^{2 \cdot n_2 - n}$. Thus, we can exploit the birthday-paradox like effect of size at most $2^{2 \cdot n_2 - n}$. Unlike in the previous cases, δ and δ' cannot be fixed in advance. To overcome this issue, whenever we choose $D_{i,2}$, we compute $D_{i+1,2}$ by $L \cdot D_{i,2}$. Then for any paired values $D_{i,2}$ and $D'_{i,2}$ with a difference $D_{i,2} \oplus D'_{i,2}$, the corresponding paired values of $D_{i+1,2}$ have a difference $L \cdot D_{i,2} \oplus L \cdot D'_{i,2} = L \cdot (D_{i,2} \oplus D'_{i,2})$. Hence, the coefficient L is preserved for any paired values of $D_{i,2}$.

We also consider all 2^{n_1} possible values of $D_{i,1}$ to exploit the birthday-paradox like effect of size 2^{n_1}. The attack procedure is given in Alg. 3. We introduce two parameters s and t where $s \leq n_1$ and $t \leq n_2$ to evaluate the number of queries.

Algorithm 3 Attack for Case 2 with $L_{2,4}$, $L_{1,4}$, $L_{1,3}$, and $L_{2,3}$ are Non-zero.

1: Compute 2^t choices of $\delta := \Delta D_{i,2}$ and $\delta' := \Delta D_{i+1,2}$ satisfying $\delta' := L \cdot \delta$.
2: Choose any value for $D_{i+1,1}$ denoted by z.
3: **for** $j = 1, 2, \ldots, 2^s$ **do**
4: Choose a value of $D_{i,1}$ denoted by x_j that is different from other x_j.
5: **for** $k = 1, 2, \ldots, 2^t$ **do**
6: Choose a value of $D_{i,2}$ denoted by y_k from δ that is different from other y_k.
7: Compute $D_{i+1,2}$ denoted by w_k by $w_k = L \cdot y_k$.
8: Query $x_j \| y_k \| z \| w_k$ and store the corresponding output denoted by $Q_{j,k}$.
9: **end for**
10: **end for**
11: Find a pair of paired indices (j_1, k_1) and (j_2, k_2) such that $Q_{j_1,k_1} = Q_{j_2,k_2}$.

We show the complexity analysis of Alg. 3. The attack makes 2^{s+t} queries to collect $Q_{j,k}$. Two distinct (j,k) achieve the difference in $D_{i,1}, D_{i,2}, D_{i+1,1}, D_{i+1,2}$ in Fig. 7. Hence if $\binom{2^{s+t}}{2} \approx 2^{2(s+t)} \geq 2^n$, we expect one pair that produces the desired ΔZ_i. From the condition $2^{2(s+t)} \geq 2^n$, the number of queries 2^{s+t} is at least $2^{n/2}$. In the following, we show that the attack complexity can be $2^{n/2}$ for any choice of n_1 and n_2 that satisfies $n_1 + n_2 \geq n+1$. The goal is to show that there always exists a choice of s and t that satisfies the following constraints.

$$s + t = n/2, \qquad s \leq n_1, \qquad t \leq 2 \cdot n_2 - n.$$

It is easy to check that when $n_1 = n/2$ and $n_2 = n/2+1$, we can set $s = n/2$ and $t = 0$, hence the attack complexity is $2^{n/2}$. On the other hand, when $n_1 = 1$ and $n_2 = n$, we can set $s = 0$ and $t = n/2$, hence the attack complexity is $2^{n/2}$. In general, the attacker can set $s = n_1$. Then t becomes $n/2 - n_1$. From $n_1 + n_2 = n+1$, we have $t = n_2 - n/2 - 1$. The

attacker can always choose such t because the constraint of the range of t is $t \geq 2 \cdot n_2 - n$. As a conclusion, there always exists the choice of s and t to achieve $n/2$ queries.

> **Example.** When $n_1 = n/4$ and $n_2 = 3n/4 + 1$, the number of (δ, δ') is about $2^{2\cdot(3n/4)-n} = 2^{n/2}$, which gives the range of t as $t \leq n/2$. We set $s = n/4$. Hence, t is $n/2 - n/4 = n/4$. This is consistent with the condition $t \leq n/2$.

The attack for $L_{2,1} = 0$ and $L_{2,2} \neq 0$ is almost the same. The coefficient L in Eq. (1) will be slightly different; the second term in Eq. (1) will not appear. The attack for $L_{2,1} \neq 0$ and $L_{2,2} = 0$ is also the same variant. The coefficient L in Eq. (1) will be slightly different; the first term in Eq. (1) will not appear. In both cases, the collision of the state is generated with a complexity of $O(2^{n/2})$.

$L_{1,2} = 0$. The attack in this case is also a variant of the attack for $L_{1,1} \neq 0$ and $L_{1,2} \neq 0$. We generate a collision on ΔZ_i. The diagram of the differential propagation is depicted in Fig. 8. Due to the similarly, we omit the details.

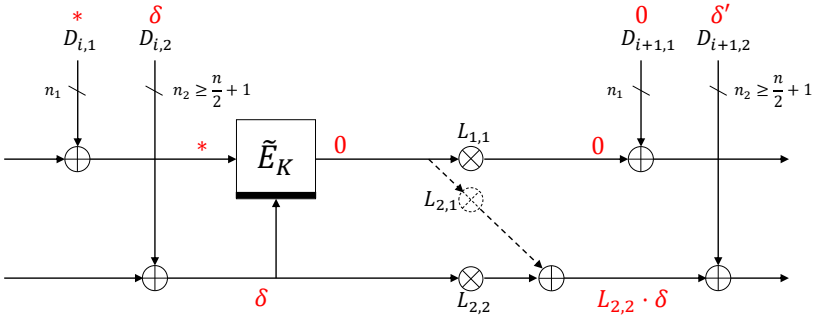

Figure 8: Differential Propagation for Case 2 with $L_{1,2} = 0$.

5 LM-DAE

5.1 Specification

We have designed a new DAE called LM-DAE that has minimum memory size. LM-DAE is designed by using the SIV paradigm [RS06], and the MAC and encryption/decryption functions iteratively use the state update function $\mathsf{SUF}[\tilde{E}_K]$ given in Section 3. In accordance with the upper bound of the data block size given in Section 4, the state update function takes an n-bit data block that is an optimal size for achieving n-bit security. In addition, to avoid the inverse tweak schedule, we introduce an n-bit permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ in the internal state, i.e., the result of the tweak schedule can be used as the next internal state. The tweak space is defined as $\mathcal{TW} = (15] \times \{0,1\}^n$, where the tweak space (15] is for the domain separation that can be achieved by a 4-bit space. The specification of LM-DAE is given in Algorithm 4 and is illustrated in Fig. 9.

Remark 1. The tweak rule for the domain separation is defined to transfer information for a length of AD to the tweaks v and w. If the information is not transferred, there exists an attack with $2^{n/2}$ complexity. The attack collects $2^{n/2}$ inputs with a distinct final AD block such that $u = 2$, and $2^{n/2}$ inputs with a distinct final AD block such that $u = 3$, then a collision of a TBC having a last AD block occurs for a pair of inputs with different domain separations, and results in a tag collision. Using the collision, one can forge a tag (or distinguish the MAC from a random function).

Algorithm 4 LM-DAE

Encryption LM-DAE.Enc$[\widetilde{E}_K](A, M)$

1: $T \leftarrow$ LM-DAE.MAC$[\widetilde{E}_K](A, M)$
2: $C \leftarrow M \oplus \mathsf{msb}_{|M|}\left(\text{LM-DAE.PRNG}[\widetilde{E}_K](T, \lceil |M|/n \rceil)\right)$
3: **return** (C, T)

Decryption LM-DAE.Dec$[\widetilde{E}_K](A, C, \hat{T})$

1: $M \leftarrow C \oplus \mathsf{msb}_{|C|}\left(\text{LM-DAE.PRNG}[\widetilde{E}_K](\hat{T}, \lceil |C|/n \rceil)\right)$
2: $T \leftarrow$ LM-DAE.MAC$[\widetilde{E}_K](A, M)$
3: **if** $T = \hat{T}$ **then return** M; **else return reject**

MAC LM-DAE.MAC$[\widetilde{E}_K](A, M)$

1: $u \leftarrow 2$; $v \leftarrow 4$ // These induces are used if $|A| \bmod n \neq 0$
2: **if** $|A| \bmod n = 0 \wedge A \neq \varepsilon$ **then** $\{u \leftarrow 3;\ v \leftarrow 5\}$
3: **if** $A = \varepsilon$ **then** $v \leftarrow 6$
4: **if** $|M| \bmod n \neq 0$ **then** $w \leftarrow v + 3$
5: **if** $|M| \bmod n = 0 \wedge M \neq \varepsilon$ **then** $w \leftarrow v + 6$
6: **if** $M = \varepsilon$ **then** $w \leftarrow v + 9$
7: $(\mathrm{Ft}, \mathrm{Fb}) \leftarrow$ LM-DAE.Hash$[\widetilde{E}_K](A, M, u, v)$
8: $\mathrm{Ft} \leftarrow \widetilde{E}_K^{w,\mathrm{Fb}}(\mathrm{Ft})$; $\mathrm{Fb} \leftarrow \pi(\mathrm{Fb}) \oplus \mathrm{Ft}$; $\mathrm{Ft} \leftarrow \widetilde{E}_K^{w,\mathrm{Fb}}(\mathrm{Ft})$; $\mathrm{Fb} \leftarrow \pi(\mathrm{Fb}) \oplus \mathrm{Ft}$
9: **return** $T \leftarrow \mathrm{Ft} \| \mathrm{Fb}$

Hash LM-DAE.Hash$[\widetilde{E}_K](A, M, u, v)$

1: $(A_1, \ldots, A_a) \xleftarrow{n} A$; $(M_1, \ldots, M_m) \xleftarrow{n} M$; $\mathrm{Ht} \leftarrow 0^n$; $\mathrm{Hb} \leftarrow 0^n$
2: **if** $A \neq \varepsilon$ **then**
3: **for** $i = 1, \ldots, a-1$ **do** $\{\mathrm{Ht} \leftarrow \widetilde{E}_K^{1,\mathrm{Hb}}(\mathrm{Ht} \oplus A_i);\ \mathrm{Hb} \leftarrow \pi(\mathrm{Hb}) \oplus \mathrm{Ht}\}$
4: $\mathrm{Ht} \leftarrow \widetilde{E}_K^{u,\mathrm{Hb}}(\mathrm{Ht} \oplus \mathsf{ozp}(A_a))$; $\mathrm{Hb} \leftarrow \pi(\mathrm{Hb}) \oplus \mathrm{Ht}$
5: **end if**
6: **if** $M \neq \varepsilon$ **then**
7: **for** $i = 1, \ldots, m$ **do** $\{\mathrm{Ht} \leftarrow \widetilde{E}_K^{v,\mathrm{Hb}}(\mathrm{Ht} \oplus \mathsf{ozp}(M_i));\ \mathrm{Hb} \leftarrow \pi(\mathrm{Hb}) \oplus \mathrm{Ht}\}$
8: **end if**
9: **return** $(H_1, H_2) \leftarrow (\mathrm{Ht}, \mathrm{Hb})$

PRNG LM-DAE.PRNG$[\widetilde{E}_K](T, m)$

1: $(\mathrm{Rt}_0, \mathrm{Rb}_0) \xleftarrow{n} T$
2: **for** $i = 1, \ldots, m$ **do** $\{\mathrm{Rt}_i \leftarrow \widetilde{E}_K^{0,\mathrm{Rb}_{i-1}}(\mathrm{Rt}_{i-1});\ \mathrm{Rb}_i \leftarrow \pi(\mathrm{Rb}_{i-1}) \oplus \mathrm{Rt}_i\}$
3: **return** $\mathrm{Rt}_1 \| \mathrm{Rt}_2 \| \cdots \| \mathrm{Rt}_m$

5.2 Security Bounds

In Theorem 2, we show that LM-DAE.MAC achieves n-bit prf-security. Thus, LM-DAE.MAC becomes a secure MAC with low memory. In Theorem 3, we show that LM-DAE achieves n-bit dae-security. The proof of Theorem 3 makes use of the prf-security in Theorem 2. In these theorems, we use the following adversarial parameters: $q_{\mathcal{E}}$ is the number of encryption queries; $q_{\mathcal{D}}$ is the number of decryption queries; σ_{M} is the number of TBC calls in LM-DAE.MAC; $\sigma_{\mathcal{E}}$ is the number of TBC calls in LM-DAE.PRNG; σ is the number of TBC calls in LM-DAE.

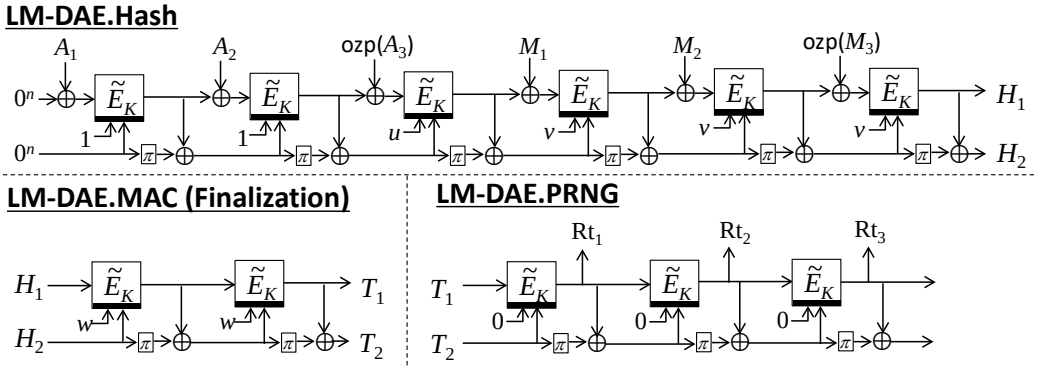

Figure 9: LM-DAE where $a = 3$ and $m = 3$.

Theorem 2. *For any permutation π, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{E}_K]}(\sigma_{\mathsf{M}}, t) \le \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \frac{2.5\sigma_{\mathsf{M}}^2}{(2^n - \sigma_{\mathsf{M}})^2} + \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma_{\mathsf{M}}, t + O(\sigma_{\mathsf{M}})) \ ,$$

where a tprp*-adversary makes σ_{M} queries.*

Theorem 3. *Let $\mathcal{Q} := (q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma_{\mathsf{M}}, \sigma_{\mathcal{E}}, \sigma)$. For any permutation π, we have*

$$\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{E}_K]}(\mathcal{Q}, t) \le \frac{\sigma_{\mathcal{E}}^2}{2^{2n}} + \frac{q_{\mathcal{D}}}{2^{2n}} + \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \frac{2.5\sigma_{\mathsf{M}}^2}{(2^n - \sigma_{\mathsf{M}})^2} + \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma, t + O(\sigma)) \ ,$$

where a tprp*-adversary makes σ queries.*

Remark 2. We discuss the tightness of the upper bound. For privacy, the randomness of a ciphertext cannot be ensured if a collision for inputs to $\widetilde{E}_K$ in LM-DAE.PRNG occurs. By the birthday analysis, the collision probability is roughly $\sigma_{\mathcal{E}}^2/2^{2n}$. For authenticity, there are two strategies for forging a tag. The first strategy is to guess a tag of $2n$ bits. The success probability is roughly $q_{\mathcal{D}}/2^{2n}$. The second strategy is to use a collision of $2n$ internal state values of LM-DAE.Hash by encryption queries, which yields a collision of tags. The collision messages offer a new collision by extending the messages: appending a message block with the collision messages. Using the extended messages, one can forge a tag. By the birthday analysis, the collision probability is roughly $q_{\mathcal{E}}^2/2^{2n}$. Hence, the lower bound of LM-DAE is roughly $\sigma_{\mathcal{E}}^2/2^{2n} + q_{\mathcal{D}}/2^{2n} + q_{\mathcal{E}}^2/2^{2n}$. For our bound in Theorem 3, the terms $\sigma_{\mathsf{M}}/(2^n - \sigma_{\mathsf{M}}) + 2.5\sigma_{\mathsf{M}}^2/(2^n - \sigma_{\mathsf{M}})^2$ are different from the lower bound, which include lengths of data processed in LM-DAE.MAC and might be improved.[6] Filling the gap between lower and upper bounds is an open problem.

6 Proof of Theorem 2

First, the keyed TBC $\widetilde{E}_K$ is replaced with a TRP $\widetilde{P}$. By this replacement, the tprp-advantage function $\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma_{\mathsf{M}}, t + O(\sigma_{\mathsf{M}}))$ is introduced in the security bound, and the remaining work is to upper-bound the dae-advantage function $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\sigma_{\mathsf{M}})$, where adversaries are computationally unbounded algorithms and the complexities are solely measured by query complexity. Without loss of generality, adversaries are deterministic.

[6] Indeed, some CBC-type MACs, which iterate a BC, achieve length-free security [Pie06, JN16]. On the other hand, the CBC-type MACs update all bits of an internal state for each BC call, whereas LM-DAE updates half bits of an internal state for each TBC call. The difference might become impossible to remove data lengths from the terms.

6.1 Strategy and Upper Bound of $\mathrm{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\sigma_{\mathsf{M}})$

Let q_{M} be the number of queries by an adversary, $\mathsf{H} := \mathsf{LM\text{-}DAE.Hash}$, and $\ell := a + m$. For $\alpha \in [q_{\mathsf{M}}]$, the lengths a, m, and ℓ at the α-th query are denoted by a_α, m_α, and ℓ_α, respectively, and values and variables defined at the α-th query are denoted by using the superscript of (α), e.g., $M^{(\alpha)}, T^{(\alpha)}$, etc.

Let zero be an event that some output of $\widetilde{P}$ defined in $\mathsf{H}[\widetilde{P}]$ is equal to 0^n. As an output of $\widetilde{P}$ is chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values, we have

$$\Pr[\mathsf{zero}] \le \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} \ .$$

We next define two probabilities for $\mathsf{H}[\widetilde{P}]$. For two inputs to $\mathsf{H}[\widetilde{P}]$: $(A^{(\alpha)}, M^{(\alpha)}, u^{(\alpha)}, v^{(\alpha)})$ and $(A^{(\beta)}, M^{(\beta)}, u^{(\beta)}, v^{(\beta)})$, the first (collision) probability is defined as follows:

$$\delta_1(\alpha, \beta) := \begin{cases} \Pr[(H_1^{(\alpha)}, H_2^{(\alpha)}) = (H_1^{(\beta)}, H_2^{(\beta)}) | \neg\mathsf{zero}] & \text{if } w^{(\alpha)} = w^{(\beta)} \ , \\ 0 & \text{if } w^{(\alpha)} \neq w^{(\beta)} \ . \end{cases}$$

Note that by the definition of $\mathsf{LM\text{-}DAE}$, $w^{(\alpha)} = w^{(\beta)}$ implies $(u^{(\alpha)}, v^{(\alpha)}) = (u^{(\beta)}, v^{(\beta)})$. The second probability is defined as follows: for an input to $\mathsf{H}[\widetilde{P}]$: $(A^{(\alpha)}, M^{(\alpha)}, u^{(\alpha)}, v^{(\alpha)})$,

$$\delta_2(\alpha) := \max_{Z \in \{0,1\}^n} \Pr[H_2^{(\alpha)} = Z \wedge (A^{(\alpha)} \neq \varepsilon \vee M^{(\alpha)} \neq \varepsilon)] \ .$$

We then give two lemmas. In the first one, the prf-security bound of $\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]$ is given that uses the probabilities $\Pr[\mathsf{zero}]$, $\delta_1(\alpha, \beta)$, and $\delta_2(\alpha)$. In the second one, the upper bounds of $\delta_1(\alpha, \beta)$ and $\delta_2(\alpha)$ are given.

Lemma 1.

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(q_{\mathsf{M}}) \le \Pr[\mathsf{zero}] + \sum_{\alpha \in [q_{\mathsf{M}}]} \sum_{\beta \in [\alpha - 1]} \left(\delta_1(\alpha, \beta) + \frac{3\delta_2(\alpha)}{2^n} \right) + \frac{0.5 q_{\mathsf{M}}^2}{2^{2n}} \ .$$

Lemma 2. *For two distinct inputs to* $\mathsf{H}[\widetilde{P}]$*:* $(A^{(\alpha)}, M^{(\alpha)})$ *and* $(A^{(\beta)}, M^{(\beta)})$ *such that* $w^{(\alpha)} = w^{(\beta)}$*, we have*

$$\delta_1(\alpha, \beta) \le \frac{\ell_\alpha + \ell_\beta}{(2^n - \sigma_{\mathsf{M}})^{2n}} \ .$$

For an input to $\mathsf{H}[\widetilde{P}]$*,* $(A^{(\alpha)}, M^{(\alpha)})$*, we have*

$$\delta_2(\alpha) \le \frac{1}{2^n - \sigma_{\mathsf{M}}} \ .$$

Putting the upper bounds of $\Pr[\mathsf{zero}]$, $\delta_1(\alpha, \beta)$ and $\delta_2(\alpha)$ into Lemma 1 gives

$$\begin{aligned} \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\sigma_{\mathsf{M}}) &\le \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \sum_{\alpha \in [q_{\mathsf{M}}]} \sum_{\beta \in [\alpha - 1]} \left(\frac{\ell_\alpha + \ell_\beta}{(2^n - \sigma_{\mathsf{M}})^2} + \frac{3}{2^n (2^n - \sigma_{\mathsf{M}})} \right) + \frac{0.5 q_{\mathsf{M}}^2}{2^{2n}} \\ &\le \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \frac{0.5\sigma_{\mathsf{M}}^2}{(2^n - \sigma_{\mathsf{M}})^2} + \frac{1.5 q_{\mathsf{M}}^2}{2^n (2^n - \sigma_{\mathsf{M}})} + \frac{0.5 q_{\mathsf{M}}^2}{2^{2n}} \\ &\le \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \frac{2.5\sigma_{\mathsf{M}}^2}{(2^n - \sigma_{\mathsf{M}})^2} \ . \end{aligned}$$

6.2 Proof of Lemma 1

This proof permits for an adversary $\mathbf{A}$ to obtain all hash values $(H_1^{(1)}, H_2^{(1)}), \ldots, (H_1^{(q_\mathsf{M})}, H_2^{(q_\mathsf{M})})$ after all queries. In the ideal world, after all queries, a TRP $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^n)$ is defined and then the hash values are defined. Hence, $\mathbf{A}$ obtains the following values, called "transcript,"

$$\tau := \left\{(A^{(1)}, M^{(1)}, T^{(1)}, H_1^{(1)}, H_2^{(1)}), \ldots, (A^{(q_\mathsf{M})}, M^{(q_\mathsf{M})}, T^{(q_\mathsf{M})}, H_1^{(q_\mathsf{M})}, H_2^{(q_\mathsf{M})})\right\} .$$

In this proof, we use the following notations: for each $\alpha \in [q_\mathsf{M}]$,

$$T_1^{(\alpha)} := \mathsf{msb}_n(T^{(\alpha)}), \quad T_2^{(\alpha)} := \mathsf{lsb}_n(T^{(\alpha)}), \quad X_1^{(\alpha)} := H_1^{(\alpha)}, \quad Y_1^{(\alpha)} := H_2^{(\alpha)},$$
$$X_2^{(\alpha)} := \pi(H_2^{(\alpha)}) \oplus \pi^{-1}(T_1^{(\alpha)} \oplus T_2^{(\alpha)}), \quad Y_2^{(\alpha)} := \pi^{-1}(T_1^{(\alpha)} \oplus T_2^{(\alpha)}) .$$

In the real world, for $i \in [2]$, $X_i^{(\alpha)}$ and $Y_i^{(\alpha)}$ are the input block and tweak at the i-th TRP call in the finalization function.

6.2.1 Coefficient H Technique

This proof uses the coefficient H technique [Pat08]. Let T_1 be a transcript in the real world obtained by sampling $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^n)$. Let T_2 be a transcript in the ideal world obtained by sampling $\mathcal{R} \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \{0,1\}^{2n})$ where $\mathcal{X}$ is the domain space of LM-DAE.MAC. We call a transcript τ *valid* if $\Pr[\mathsf{T}_2 = \tau] > 0$. Let $\mathcal{T}$ be all valid transcripts. Then,

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\mathbf{A}) = \mathsf{SD}(\mathsf{T}_1, \mathsf{T}_2) = \frac{1}{2}\sum_{\tau \in \mathcal{T}} |\Pr[\mathsf{T}_1 = \tau] - \Pr[\mathsf{T}_2 = \tau]| .$$

Then, the statistical distance $\mathsf{SD}(\mathsf{T}_1, \mathsf{T}_2)$ is upper-bounded as follows. $\mathcal{T}$ is partitioned into two transcripts: good transcripts $\mathcal{T}_\mathsf{good}$ and bad transcripts $\mathcal{T}_\mathsf{bad}$. Then, $\mathsf{SD}(\mathsf{T}_1, \mathsf{T}_2)$ is upper-bounded by the following lemma.

Lemma 3 ([Pat08])**.** *Let $0 \le \varepsilon \le 1$ be such that for all $\tau \in \mathcal{T}_\mathsf{good}$, $\frac{\Pr[\mathsf{T}_1=\tau]}{\Pr[\mathsf{T}_2=\tau]} \ge 1-\varepsilon$. Then,* $\mathsf{SD}(\mathsf{T}_1, \mathsf{T}_2) \le \Pr[\mathsf{T}_2 \in \mathcal{T}_\mathsf{bad}] + \varepsilon$.

Hereafter, first good and bad transcripts are defined. Then $\Pr[\mathsf{T}_2 \in \mathcal{T}_\mathsf{bad}]$ is upper bounded, and $\frac{\Pr[\mathsf{T}_1=\tau]}{\Pr[\mathsf{T}_2=\tau]}$ is lower-bounded. Finally, by the above lemma, an upper bound of $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\mathbf{A})$ is obtained.

6.2.2 Definitions of Good and Bad Transcripts

Bad transcripts $\mathcal{T}_\mathsf{bad}$ are defined so that one of the following conditions is satisfied. Good transcripts $\mathcal{T}_\mathsf{good}$ are defined as $\mathcal{T}_\mathsf{good} = \mathcal{T} \backslash \mathcal{T}_\mathsf{bad}$.

- $\mathsf{zero} \Leftrightarrow$ some output of $\widetilde{P}$ in the hash function is equal to 0^n.
- $\mathsf{bad}_1 \Leftrightarrow \exists \alpha, \beta \in [q_\mathsf{M}], i, j \in [2]$ s.t.
  $(\alpha, i) \neq (\beta, j) \wedge (X_i^{(\alpha)}, Y_i^{(\alpha)}) = (X_j^{(\beta)}, Y_j^{(\beta)}) \wedge w^{(\alpha)} = w^{(\beta)}$.
- $\mathsf{bad}_2 \Leftrightarrow \exists \alpha, \beta \in [q_\mathsf{M}]$ s.t.
  $\alpha \neq \beta \wedge X_1^{(\alpha)} \neq X_1^{(\beta)} \wedge Y_1^{(\alpha)} = Y_1^{(\beta)} \wedge X_2^{(\alpha)} = X_2^{(\beta)} \wedge w^{(\alpha)} = w^{(\beta)}$.
- $\mathsf{bad}_3 \Leftrightarrow \exists \alpha, \beta \in [q_\mathsf{M}]$ s.t.
  $\alpha \neq \beta \wedge X_2^{(\alpha)} \neq X_2^{(\beta)} \wedge Y_2^{(\alpha)} = Y_2^{(\beta)} \wedge T_1^{(\alpha)} = T_1^{(\beta)} \wedge w^{(\alpha)} = w^{(\beta)}$.

bad_1 defines an input collision for $\widetilde{P}^{w^{(\alpha)}}$. For $i \in \{2,3\}$, bad_i defines an impossible output for $\widetilde{P}$ in the finalization function, that is, the input blocks are distinct and the tweaks are the same, but the output is the same.

6.2.3 Upper Bound of $\mathbf{Pr}[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}]$

We upper-bound the four probabilities in the ideal world $\Pr[\mathsf{zero}]$, $\Pr[\mathsf{bad}_1|\neg\mathsf{zero}]$, $\Pr[\mathsf{bad}_2]$, and $\Pr[\mathsf{bad}_3]$, because

$$\Pr[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}] \leq \Pr[\mathsf{zero}] + \Pr[\mathsf{bad}_1|\neg\mathsf{zero}] + \Pr[\mathsf{bad}_2] + \Pr[\mathsf{bad}_3] \ .$$

The upper bounds of $\Pr[\mathsf{bad}_1|\neg\mathsf{zero}]$, $\Pr[\mathsf{bad}_2]$ and $\Pr[\mathsf{bad}_3]$ (given below) give

$$\Pr[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}] \leq \Pr[\mathsf{zero}] + \sum_{\alpha\in[q_{\mathsf{M}}]}\sum_{\beta\in[\alpha-1]}\left(\delta_1(\alpha,\beta) + \frac{3\delta_2(\alpha)}{2^n}\right) + \frac{0.5q_{\mathsf{M}}^2}{2^{2n}} \ .$$

Upper Bounding $\mathbf{Pr}[\mathsf{bad}_1|\neg\mathsf{zero}]$. The following cases are considered.

1. $\exists\alpha,\beta \in [q_{\mathsf{M}}]$ s.t. $\alpha \neq \beta \wedge (X_1^{(\alpha)}, Y_1^{(\alpha)}) = (X_1^{(\beta)}, Y_1^{(\beta)}) \wedge w^{(\alpha)} = w^{(\beta)}$. These collisions are of the forms
$$X_1^{(\alpha)} = X_1^{(\beta)} \Leftrightarrow H_1^{(\alpha)} = H_1^{(\beta)} \qquad Y_1^{(\alpha)} = Y_1^{(\beta)} \Leftrightarrow H_2^{(\alpha)} = H_2^{(\beta)}.$$
The probability that this case occurs is at most
$$\sum_{\alpha\in[q_{\mathsf{M}}]}\sum_{\beta\in[\alpha-1]}\delta_1(\alpha,\beta) \ .$$

2. $\exists\alpha,\beta \in [q_{\mathsf{M}}]$ s.t. $(X_1^{(\alpha)}, Y_1^{(\alpha)}) = (X_2^{(\beta)}, Y_2^{(\beta)}) \wedge w^{(\alpha)} = w^{(\beta)}$. These collisions are of the forms
$$X_1^{(\alpha)} = X_2^{(\beta)} \Leftrightarrow H_1^{(\alpha)} = \pi(H_2^{(\beta)}) \oplus \pi^{-1}(T_1^{(\beta)} \oplus T_2^{(\beta)})$$
$$Y_1^{(\alpha)} = Y_2^{(\beta)} \Leftrightarrow H_2^{(\alpha)} = \pi^{-1}(T_1^{(\beta)} \oplus T_2^{(\beta)}).$$
Fix α, β, $H_1^{(\alpha)}$, $H_2^{(\beta)}$, and $T_2^{(\beta)}$. Regarding $X_1^{(\alpha)} = X_2^{(\beta)}$, as $T_1^{(\beta)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[X_1^{(\alpha)} = X_2^{(\beta)}] \leq 1/2^n$. Regarding $Y_1^{(\alpha)} = Y_2^{(\beta)}$, we have $\Pr[Y_1^{(\alpha)} = Y_2^{(\beta)}] \leq \delta_2(\alpha)$. Summing the upper bound for each combination (α,β), the probability that this case occurs is at most
$$\sum_{\alpha\in[q_{\mathsf{M}}]}\sum_{\beta\in[\alpha-1]}\frac{\delta_2(\alpha)}{2^n} \ .$$

3. $\exists\alpha,\beta \in [q_{\mathsf{M}}]$ s.t. $\alpha \neq \beta \wedge (X_2^{(\alpha)}, Y_2^{(\alpha)}) = (X_2^{(\beta)}, Y_2^{(\beta)}) \wedge w^{(\alpha)} = w^{(\beta)}$. These collisions are of the forms
$$X_2^{(\alpha)} = X_2^{(\beta)} \Leftrightarrow \pi(H_2^{(\alpha)}) \oplus \pi^{-1}(T_1^{(\alpha)} \oplus T_2^{(\alpha)}) = \pi(H_2^{(\beta)}) \oplus \pi^{-1}(T_1^{(\beta)} \oplus T_2^{(\beta)})$$
$$Y_2^{(\alpha)} = Y_2^{(\beta)} \Leftrightarrow \pi^{-1}(T_1^{(\alpha)} \oplus T_2^{(\alpha)}) = \pi^{-1}(T_1^{(\beta)} \oplus T_2^{(\beta)}).$$
Fix α, β, $H_2^{(\beta)}$, $T_2^{(\alpha)}$, $T_1^{(\beta)}$ and $T_2^{(\beta)}$. Regarding $Y_2^{(\alpha)} = Y_2^{(\beta)}$, as $T_1^{(\beta)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[Y_2^{(\alpha)} = Y_2^{(\beta)}] \leq 1/2^n$. Regarding $X_2^{(\alpha)} = X_2^{(\beta)}$, fixing $T_1^{(\beta)}$, by $H_2^{(\alpha)}$, we have $\Pr[X_2^{(\alpha)} = X_2^{(\beta)}] \leq \delta_2(\alpha)$. Summing the upper bound for each (α,β), the probability that this case occurs is at most
$$\sum_{\alpha\in[q_{\mathsf{M}}]}\sum_{\beta\in[\alpha-1]}\frac{\delta_2(\alpha)}{2^n} \ .$$

Hence, we have

$$\Pr[\mathsf{bad}_1|\neg\mathsf{zero}] \leq \sum_{\alpha\in[q_\mathsf{M}]}\sum_{\beta\in[\alpha-1]}\left(\delta_1(\alpha,\beta)+\frac{2\delta_2(\alpha)}{2^n}\right) \ .$$

Upper Bounding $\mathbf{Pr[bad_2]}$. First fix $\alpha,\beta\in[q_\mathsf{M}]$ such that $\alpha\neq\beta$. The collisions $Y_1^{(\alpha)}=Y_1^{(\beta)}$ and $X_2^{(\alpha)}=X_2^{(\beta)}$ are of the forms

$$Y_1^{(\alpha)}=Y_1^{(\beta)} \Leftrightarrow H_2^{(\alpha)}=H_2^{(\beta)}$$
$$X_2^{(\alpha)}=X_2^{(\beta)} \Leftrightarrow \pi(H_2^{(\alpha)})\oplus\pi^{-1}(T_1^{(\alpha)}\oplus T_2^{(\alpha)})=\pi(H_2^{(\beta)})\oplus\pi^{-1}(T_1^{(\beta)}\oplus T_2^{(\beta)}).$$

Fix $H_2^{(\beta)}$, $T_2^{(\alpha)}$, $T_1^{(\beta)}$, and $T_2^{(\beta)}$. Regarding $Y_1^{(\alpha)}=Y_1^{(\beta)}$, we have $\Pr[Y_1^{(\alpha)}=Y_1^{(\beta)}]\leq\delta_2(\alpha)$. Regarding $X_2^{(\alpha)}=X_2^{(\beta)}$, as $T_1^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[X_2^{(\alpha)}=X_2^{(\beta)}]\leq 1/2^n$. Summing the upper bound for each (α,β), we have

$$\Pr[\mathsf{bad}_2] \leq \sum_{\alpha\in[q_\mathsf{M}]}\sum_{\beta\in[\alpha-1]}\frac{\delta_2(\alpha)}{2^n} \ .$$

Upper Bounding $\mathbf{Pr[bad_3]}$. First fix $\alpha,\beta\in[q_\mathsf{M}]$ such that $\alpha\neq\beta$, and fix $T_1^{(\beta)}$ and $T_2^{(\alpha)}$. Regarding the condition $T_1^{(\alpha)}=T_1^{(\beta)}$, as $T_1^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[T_1^{(\alpha)}=T_1^{(\beta)}]\leq 1/2^n$. Regarding the condition $Y_2^{(\alpha)}=Y_2^{(\beta)}$, $Y_2^{(\alpha)}=Y_2^{(\beta)} \Leftrightarrow \pi^{-1}(T_1^{(\alpha)}\oplus T_2^{(\alpha)})=\pi^{-1}(T_1^{(\beta)}\oplus T_2^{(\beta)})$ is satisfied. As $T_2^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[Y_2^{(\alpha)}=Y_2^{(\beta)}]\leq 1/2^n$. Summing the upper bound for each (α,β), we have

$$\Pr[\mathsf{bad}_3] \leq \frac{0.5q_\mathsf{M}^2}{2^{2n}} \ .$$

6.2.4 Lower Bound of $\frac{\mathbf{Pr[\mathsf{T}_1=\tau]}}{\mathbf{Pr[\mathsf{T}_2=\tau]}}$

Let $\tau\in\mathcal{T}_{\mathsf{good}}$. Let p_H be the probability that a hash function is compatible with τ.

Regarding $\Pr[\mathsf{T}_2=\tau]$, in the ideal world, for each $\alpha\in[q_\mathsf{M}]$, $T^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^{2n}$. We thus have

$$\Pr[\mathsf{T}_2=\tau] = p_H\cdot\frac{1}{2^{2nq_\mathsf{M}}} \ .$$

Next, $\Pr[\mathsf{T}_1=\tau]$ is lower bounded. By bad_2 and bad_3, one has $\Pr[\mathsf{T}_1=\tau]>0$. Let $\sigma[w,Y]$ be the number of input blocks to $\widetilde{P}^{w,Y}$. Then, we have

$$\Pr[\mathsf{T}_R=\tau] = p_H\cdot\prod_{w\in[7,15],Y\in\{0,1\}^n}\frac{1}{(2^n)_{\sigma[w,Y]}} \geq \prod_{w\in[7,15],Y\in\{0,1\}^n}\frac{1}{(2^n)^{\sigma[w,Y]}} \ .$$

By $\neg\mathsf{bad}_1$, we have

$$\sum_{w\in[7,15],Y\in\{0,1\}^n}\sigma[w,Y]=2q_\mathsf{M}$$

as the total number of TRP calls in the finalization is $2q_\mathsf{M}$, and thus

$$\Pr[\mathsf{T}_1=\tau] \geq p_H\cdot\frac{1}{2^{2nq_\mathsf{M}}} \ .$$

Finally, these bounds give

$$\frac{\Pr[\mathsf{T}_1=\tau]}{\Pr[\mathsf{T}_2=\tau]}\geq 1 \ .$$

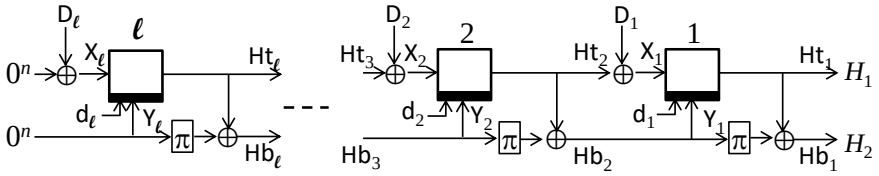

Figure 10: LM-DAE.Hash with indices counted from the last TRP call to the first one.

6.2.5 Upper Bound of $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(q_{\mathsf{M}})$

Putting the above bounds into Lemma 3, we obtain the upper bound in Lemma 1.

6.3 Proof of Lemma 2: Upper Bounding $\delta_1(\alpha, \beta)$

In this proof, values and variables for $(A^{(\alpha)}, M^{(\alpha)})$ (resp. $(A^{(\beta)}, M^{(\beta)})$) are denoted by using the black dot $\bullet$ (resp. the star $\star$) instead of (α) (resp. (β)). Hence, we upper-bound the probability that for two distinct inputs $(A^\bullet, M^\bullet), (A^\star, M^\star)$ such that $w^\bullet = w^\star$ (i.e., $u^\bullet = u^\star$ and $v^\bullet = v^\star$), the collisions $(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star)$ occur, under the assumption that zero does not occur.

In the following analysis, we count indices from the last TRP call in the hash function and then use Sans-serif font for the values with the count rule such as $\mathsf{Ht}_i, \mathsf{Hb}_i$, which are the i-th internal states values (see Fig. 10). The input block and tweaks for i-th TRP inputs (the indices are counted from the last TRP call) are denoted by X_i and $(\mathsf{d}_i, \mathsf{Y}_i)$, respectively, where

$$\mathsf{d}_i := d_{m-i+1} \ (i \in [m]) \text{ and } \mathsf{d}_{i+m} := d_{a-i+1} \ (i \in [a]).$$

Data blocks are denoted by

$$\mathsf{D}_i := M_{m-i+1} \ (i \in [m]) \text{ and } \mathsf{D}_{i+m} := A_{a-i+1} \ (i \in [a]).$$

Let

$$\mathcal{I}^{\neq} := \left\{ i \middle| i \in [\max\{\ell_\bullet, \ell_\star\}] \wedge (\mathsf{D}_i^\bullet, \mathsf{d}_i^\bullet) \neq (\mathsf{D}_i^\star, \mathsf{d}_i^\star) \right\}$$

be a set of indices with distinct pairs of data block and tweak, where $\mathsf{D}_i := \varepsilon$ if $i > \ell$. Note that as $(A^\bullet, M^\bullet) \neq (A^\star, M^\star)$, $|\mathcal{I}^{\neq}| \geq 1$ is satisfied.

In this analysis, we consider three cases.

- **Case 1**: $\forall i \in [\min\{\ell_\bullet, \ell_\star\}] : (\mathsf{D}_i^\bullet, \mathsf{d}_i^\bullet) = (\mathsf{D}_i^\star, \mathsf{d}_i^\star)$.
- **Case 2**: $\exists i \in [\min\{\ell_\bullet, \ell_\star\}]$ s.t. $(\mathsf{D}_i^\bullet, \mathsf{d}_i^\bullet) \neq (\mathsf{D}_i^\star, \mathsf{d}_i^\star) \wedge \mathsf{D}_\gamma^\bullet \neq \mathsf{D}_\gamma^\star \wedge \mathsf{d}_\gamma^\bullet = \mathsf{d}_\gamma^\star$ for $\gamma = \min \mathcal{I}^{\neq}$.
- **Case 3**: $\exists i \in [\min\{\ell_\bullet, \ell_\star\}]$ s.t. $(\mathsf{D}_i^\bullet, \mathsf{d}_i^\bullet) \neq (\mathsf{D}_i^\star, \mathsf{d}_i^\star) \wedge \mathsf{d}_\gamma^\bullet \neq \mathsf{d}_\gamma^\star$ for $\gamma = \min \mathcal{I}^{\neq}$.

For each case, the probability $\delta_1(\alpha, \beta) = \Pr[(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star) | \neg\mathsf{zero}]$ where $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^n)$ is upper bounded below. These upper bounds give

$$\delta_1(\alpha, \beta) \leq \frac{\ell_\bullet + \ell_\star}{(2^n - \sigma_{\mathsf{M}})^2} \ .$$

6.3.1 Case 1

In this case, $\ell_\bullet \neq \ell_\star$ must be satisfied. Without loss of generality, we assume that $\ell_\bullet > \ell_\star$. Then, $(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star) \Leftrightarrow (\mathsf{Ht}_{\ell_\star+1}^\bullet, \mathsf{Hb}_{\ell_\star+1}^\bullet) = (0^n, 0^n)$ is satisfied. By $\neg\mathsf{zero}$, in Case 1, we have

$$\delta_1(\alpha, \beta) = 0 \ .$$

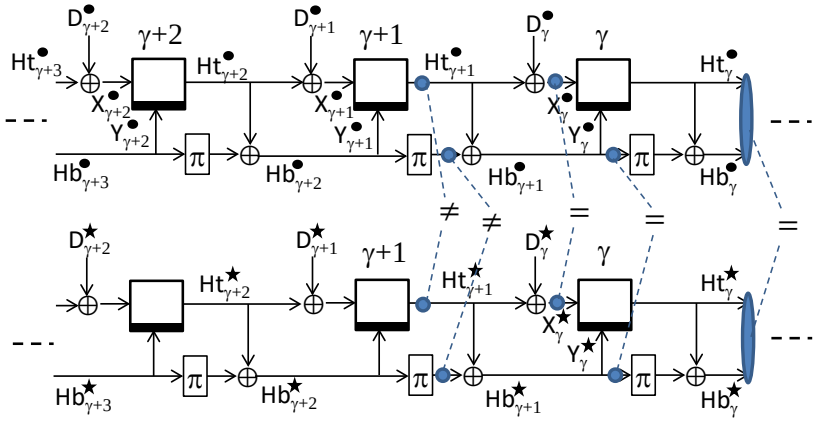

Figure 11: Hash procedures in Case 2, where tweaks for domain separation are omitted.

6.3.2 Case 2

Let $\gamma := \min \mathcal{I}^{\neq}$. In this case, $(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star) \Leftrightarrow (\mathsf{X}_\gamma^\bullet, \mathsf{Y}_\gamma^\bullet) = (\mathsf{X}_\gamma^\star, \mathsf{Y}_\gamma^\star)$ is satisfied, where

$$\mathsf{X}_\gamma^\bullet = \mathsf{X}_\gamma^\star \Leftrightarrow \mathsf{D}_\gamma^\bullet \oplus \mathsf{Ht}_{\gamma+1}^\bullet = \mathsf{D}_\gamma^\star \oplus \mathsf{Ht}_{\gamma+1}^\star,$$
$$\mathsf{Y}_\gamma^\bullet = \mathsf{Y}_\gamma^\star \Leftrightarrow \mathsf{Hb}_{\gamma+1}^\bullet = \mathsf{Hb}_{\gamma+1}^\star \Leftrightarrow \mathsf{Ht}_{\gamma+1}^\bullet \oplus \pi(\mathsf{Hb}_{\gamma+2}^\bullet) = \mathsf{Ht}_{\gamma+1}^\star \oplus \pi(\mathsf{Hb}_{\gamma+2}^\star).$$

See also Fig. 11. Thus, one has $(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star) \Leftrightarrow$

$$\mathsf{Ht}_{\gamma+1}^\bullet \oplus \mathsf{Ht}_{\gamma+1}^\star = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star \text{ and } \pi(\mathsf{Hb}_{\gamma+2}^\bullet) \oplus \pi(\mathsf{Hb}_{\gamma+2}^\star) = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star.$$

By $\mathsf{D}_\gamma^\bullet \neq \mathsf{D}_\gamma^\star$, to have the collision $(H_1^\bullet, H_2^\bullet) = (H_1^\star, H_2^\star)$, the following conditions must be satisfied:

$$\mathsf{Ht}_{\gamma+1}^\bullet \neq \mathsf{Ht}_{\gamma+1}^\star \text{ and } \pi(\mathsf{Hb}_{\gamma+2}^\bullet) \neq \pi(\mathsf{Hb}_{\gamma+2}^\star) \text{ (i.e, } \mathsf{Hb}_{\gamma+2}^\bullet \neq \mathsf{Hb}_{\gamma+2}^\star).$$

We first upper-bound $\Pr[\mathsf{Ht}_{\gamma+1}^\bullet \oplus \mathsf{Ht}_{\gamma+1}^\star = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star]$. By $\mathsf{Ht}_{\gamma+1}^\bullet \neq \mathsf{Ht}_{\gamma+1}^\star$, the TRP outputs $\mathsf{Ht}_{\gamma+1}^\bullet$ and $\mathsf{Ht}_{\gamma+1}^\star$ are distinct variables and chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values in $\{0,1\}^n$. Using the randomness of $\mathsf{Ht}_{\gamma+1}^\bullet$ or $\mathsf{Ht}_{\gamma+1}^\star$, we have

$$\Pr[\mathsf{Ht}_{\gamma+1}^\bullet \oplus \mathsf{Ht}_{\gamma+1}^\star = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star] \leq \frac{1}{2^n - \sigma_{\mathsf{M}}} \ .$$

We next upper-bound $\Pr[\pi(\mathsf{Hb}_{\gamma+2}^\bullet) \oplus \pi(\mathsf{Hb}_{\gamma+2}^\star) = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star]$, where $\mathsf{Hb}_{\gamma+2} = \mathsf{Ht}_{\gamma+2} \oplus \pi(\mathsf{Ht}_{\gamma+3} \oplus \pi(\mathsf{Ht}_{\gamma+4} \oplus \pi(\cdots \pi(\mathsf{Ht}_\ell \oplus \pi(0^n)) \cdots)))$. Without loss of generality, we assume that $\ell_\bullet \geq \ell_\star$. Then, the following cases are considered.

- In the first case, $\mathsf{X}_{\gamma+2}^\bullet$ collides with at least one of $\{\mathsf{X}_{\gamma+3}^\bullet, \ldots, \mathsf{X}_{\ell_\bullet}^\bullet, \mathsf{X}_{\gamma+2}^\star, \ldots, \mathsf{X}_{\ell_\star}^\star\}$, which are input blocks with indices greater than $\gamma+1$ except for $\mathsf{X}_{\gamma+2}^\bullet$. $\mathsf{X}_{\gamma+2}^\bullet = \mathsf{Ht}_{\gamma+3}^\bullet \oplus \mathsf{D}_{\gamma+2}^\bullet$ is satisfied, and $\mathsf{Ht}_{\gamma+3}^\bullet$ is chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values in $\{0,1\}^n$. By using the randomness of $\mathsf{Ht}_{\gamma+3}^\bullet$, the collision probability is at most
$$\frac{\ell_\bullet - 1 + \ell_\star}{2^n - \sigma_{\mathsf{M}}} \ .$$
- In the second case, no such collision occurs, that is, none of $\{\mathsf{X}_{\gamma+3}^\bullet, \ldots, \mathsf{X}_{\ell_\bullet}^\bullet, \mathsf{X}_{\gamma+2}^\star, \ldots, \mathsf{X}_{\ell_\star}^\star\}$ is equal to $\mathsf{X}_{\gamma+2}^\bullet$. Then, $\Pr[\pi(\mathsf{Hb}_{\gamma+2}^\bullet) \oplus \pi(\mathsf{Hb}_{\gamma+2}^\star) = \mathsf{D}_\gamma^\bullet \oplus \mathsf{D}_\gamma^\star]$ is upper-bounded, using

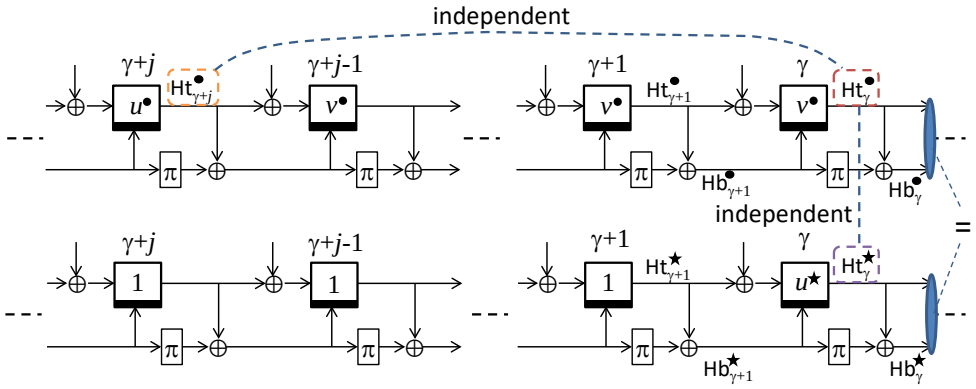

Figure 12: Hash procedures in Case 3, where tweaks for domain separation are in the TRP boxes.

the randomness of $\mathsf{Ht}^{\bullet}_{\gamma+2}$, as $\mathsf{Hb}^{\bullet}_{\gamma+2} = \mathsf{Ht}^{\bullet}_{\gamma+2} \oplus \pi(\mathsf{Hb}^{\bullet}_{\gamma+3})$. This case ensures that $\mathsf{Ht}^{\bullet}_{\gamma+2}$ is a variable distinct from other variables used to define $\mathsf{Hb}^{\bullet}_{\gamma+3}$ or $\mathsf{Hb}^{\star}_{\gamma+2}$, which are $\{\mathsf{Ht}^{\bullet}_{\gamma+3}, \ldots, \mathsf{Ht}^{\bullet}_{\ell_\bullet}, \mathsf{Ht}^{\star}_{\gamma+2}, \mathsf{Ht}^{\star}_{\gamma+3}, \ldots, \mathsf{Ht}^{\star}_{\ell_\star}\}$. Thus, as $\mathsf{Ht}^{\bullet}_{\gamma+2}$ is chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values in $\{0,1\}^n$, we have

$$\Pr[\pi(\mathsf{Hb}^{\bullet}_{\gamma+2}) \oplus \pi(\mathsf{Hb}^{\star}_{\gamma+2}) = \mathsf{D}^{\bullet}_{\gamma} \oplus \mathsf{D}^{\star}_{\gamma}] \leq \frac{1}{2^n - \sigma_{\mathsf{M}}} \ .$$

Summing the upper bounds gives

$$\Pr[\pi(\mathsf{Hb}^{\bullet}_{\gamma+2}) \oplus \pi(\mathsf{Hb}^{\star}_{\gamma+2}) = \mathsf{D}^{\bullet}_{\gamma} \oplus \mathsf{D}^{\star}_{\gamma}] \leq \frac{\ell_\bullet + \ell_\star}{2^n - \sigma_{\mathsf{M}}} \ .$$

Finally, in Case 2, we have

$$\delta_1(\alpha, \beta) \leq \frac{\ell_\bullet + \ell_\star}{(2^n - \sigma_{\mathsf{M}})^2} \ .$$

6.3.3 Case 3

Let $\gamma := \min \mathcal{I}^{\neq}$. In this case, $(H^{\bullet}_1, H^{\bullet}_2) = (H^{\star}_1, H^{\star}_2) \Leftrightarrow \mathsf{Ht}^{\bullet}_{\gamma} = \mathsf{Ht}^{\star}_{\gamma} \wedge \mathsf{Hb}^{\bullet}_{\gamma} = \mathsf{Hb}^{\star}_{\gamma}$ is satisfied, where

$$\mathsf{Hb}^{\bullet}_{\gamma} = \mathsf{Hb}^{\star}_{\gamma} \Leftrightarrow \mathsf{Ht}^{\bullet}_{\gamma} \oplus \pi(\mathsf{Ht}^{\bullet}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\bullet}_{\gamma+2})) = \mathsf{Ht}^{\star}_{\gamma} \oplus \pi(\mathsf{Ht}^{\star}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\star}_{\gamma+2})).$$

Hence, we have $(H^{\bullet}_1, H^{\bullet}_2) = (H^{\star}_1, H^{\star}_2) \Leftrightarrow$

$$\mathsf{Ht}^{\bullet}_{\gamma} = \mathsf{Ht}^{\star}_{\gamma} \text{ and } \mathsf{Ht}^{\bullet}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\bullet}_{\gamma+2}) = \mathsf{Ht}^{\star}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\star}_{\gamma+2}),$$

where $\mathsf{Ht}_{\gamma} = \widetilde{P}^{\mathsf{d}_\gamma, \mathsf{Y}_\gamma}(\mathsf{X}_\gamma)$ and $\mathsf{Ht}_{\gamma+1} = \widetilde{P}^{\mathsf{d}_{\gamma+1}, \mathsf{Y}_{\gamma+1}}(\mathsf{X}_{\gamma+1})$. By $\mathsf{d}^{\bullet}_{\gamma} \neq \mathsf{d}^{\star}_{\gamma}$, one has $a_\bullet, a_\star, m_\bullet, m_\star > 0$ and $m^{\bullet} \neq m^{\star}$. Without loss of generality, assume that $m_\bullet > m_\star$. Then, as shown in Fig. 12, by $v^{\bullet} \neq u^{\star}$, $\mathsf{Ht}^{\bullet}_{\gamma}$ and $\mathsf{Ht}^{\star}_{\gamma}$ are independent variables, and by $u^{\bullet} \neq v^{\bullet}$, $\mathsf{Ht}^{\bullet}_{\gamma}$ and $\mathsf{Ht}^{\bullet}_{\gamma+j}$ that is defined by $\widetilde{P}$ with $u^{\bullet}$ are independent variables. $\mathsf{Ht}^{\bullet}_{\gamma}$ and $\mathsf{Ht}^{\bullet}_{\gamma+j}$ are chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values in $\{0,1\}^n$. By the randomness of $\mathsf{Ht}^{\bullet}_{\gamma+j}$, we have

$$\Pr[\mathsf{Ht}^{\bullet}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\bullet}_{\gamma+2}) = \mathsf{Ht}^{\star}_{\gamma+1} \oplus \pi(\mathsf{Hb}^{\star}_{\gamma+2})] \leq \frac{1}{2^n - \sigma_{\mathsf{M}}} \ ,$$

and by the randomness of $\mathsf{Ht}^{\bullet}_{\gamma}$, we have

$$\Pr[\mathsf{Ht}^{\bullet}_{\gamma} = \mathsf{Ht}^{\star}_{\gamma}] \leq \frac{1}{2^n - \sigma_{\mathsf{M}}} \ .$$

Thus, in Case 3, we have

$$\delta_1(\alpha, \beta) \leq \frac{1}{(2^n - \sigma_{\mathsf{M}})^2} \ .$$

6.4 Proof of Lemma 2: Upper Bounding $\delta_2(\alpha)$

By $A^{(\alpha)} \neq \varepsilon \vee M^{(\alpha)} \neq \varepsilon$, the hash value $H_2^{(\alpha)}$ is defined by using the output of the last TRP call, which is chosen uniformly at random from at least $2^n - \sigma_{\mathsf{M}}$ values in $\{0,1\}^n$. Thus, for any Z, we have

$$\delta_2(\alpha) \leq \frac{1}{2^n - \sigma_{\mathsf{M}}} \ .$$

7 Proof of Theorem 3

First, the keyed TBC $\widetilde{E}_K$ is replaced with a TRP $\widetilde{P}$, and thus we have

$$\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{E}_K]}(\mathcal{Q}, t) \leq \mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(\mathcal{Q}) + \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma, t + O(\sigma)) \ .$$

Hereafter, the dae-advantage $\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma)$ is upper-bounded, where adversaries are computationally unbounded algorithms and the complexities are solely measured by the numbers of queries and their lengths. Without loss of generality, adversaries are deterministic.

Remark 3. The beginning of the proof is the same as the proof of Theorem 1 in [RS06], which shows that the dae-advantage $\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(\mathcal{Q})$ is upper-bounded by the advantage of distinguishing ciphertexts from random strings plus the prf-advantage of $\mathsf{LM\text{-}DAE.MAC}$ plus $(q_{\mathcal{E}} + q_{\mathcal{D}})/2^{2n}$. As the proof is simple, and for the sake of completeness, we show the proof in the following proof.

7.1 Strategy and Upper Bound of $\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(\mathcal{Q})$

First, the MAC function $\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]$ is replaced with a random function $\mathcal{M}$. The modified DAE ($\mathsf{LM\text{-}DAE}[\widetilde{P}]$ with $\mathcal{M}$) is denoted by $\mathsf{LM\text{-}DAE1}[\widetilde{P}, \mathcal{M}]$. The encryption (resp. decryption) is denoted by $\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P}, \mathcal{M}]$ (resp. $\mathsf{LM\text{-}DAE.Dec1}[\widetilde{P}, \mathcal{M}]$). By the replacement, we have

$$\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(\mathcal{Q}) \leq \mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE1}[\widetilde{P}, \mathcal{M}]}(\mathcal{Q}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\sigma_{\mathsf{M}}) \ .$$

The upper bound of $\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{LM\text{-}DAE.MAC}[\widetilde{P}]}(\sigma_{\mathsf{M}})$ is given in Theorem 2. The upper bound of $\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE1}[\widetilde{P}, \mathcal{M}]}(\mathcal{Q})$ is given in Eq. (2) in Subsection 7.2. We thus have

$$\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE}[\widetilde{P}]}(\mathcal{Q}) \leq \frac{\sigma_{\mathcal{E}}^2}{2^{2n}} + \frac{q_{\mathcal{D}}}{2^{2n}} + \frac{\sigma_{\mathsf{M}}}{2^n - \sigma_{\mathsf{M}}} + \frac{2.5\sigma_{\mathsf{M}}^2}{(2^n - \sigma_{\mathsf{M}})^{2n}} \ .$$

7.2 Upper Bounding $\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{R}]}(\mathcal{Q})$

A middle system between $\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]$ and an ideal system $(\$,\perp)$ is introduced: $\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}] = (\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P},\mathcal{M}],\perp)$. Using the middle system, we have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}(\mathcal{Q}) &\le \max_{\mathbf{A}}\Big(\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}=1]-\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]\Big)\\ &\quad+\max_{\mathbf{A}}\Big(\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]-\Pr[\mathbf{A}^{\$,\perp}=1]\Big)\\ &\le \max_{\mathbf{A}}\Big(\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}=1]-\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]\Big)\\ &\quad+\max_{\mathbf{A}}\Big(\Pr\Big[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P},\mathcal{M}]}=1\Big]-\Pr\Big[\mathbf{A}^{\$}=1\Big]\Big)\ .\end{aligned}$$

These upper bounds are given in Eqs. (3) and (4) in Subsecs. 7.3 and 7.4, respectively. We thus have

$$\mathbf{Adv}^{\mathsf{dae}}_{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}(\mathcal{Q}) \le \frac{\sigma_{\mathcal{E}}^2}{2^{2n}}+\frac{q_{\mathcal{D}}}{2^{2n}}\ . \tag{2}$$

7.3 Upper Bounding $\mathbf{Pr}[\mathbf{A}^{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}=1]-\mathbf{Pr}[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]$

If all responses of decryption queries to the system $\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]$ are $\perp$, then $\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]$ and $\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]$ are indistinguishable. Hence, the difference $\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}=1]-\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]$ is upper-bounded by the probability that some response of decryption query to $\mathsf{LM\text{-}DAE.MAC}[\widetilde{P},\mathcal{M}]$ is not **reject**. As an output of a random function $\mathcal{M}$ is chosen uniformly at random from $\{0,1\}^{2n}$, the probability of forging a tag is $1/2^{2n}$, and thus we have

$$\max_{\mathbf{A}}\Big(\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}[\widetilde{P},\mathcal{M}]}=1]-\Pr[\mathbf{A}^{\mathsf{LM\text{-}DAE1}^{\perp}[\widetilde{P},\mathcal{M}]}=1]\Big)\le\frac{q_{\mathcal{D}}}{2^{2n}}\ . \tag{3}$$

7.4 Upper Bounding $\mathbf{Pr}\left[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P},\mathcal{M}]}=1\right]-\mathbf{Pr}\left[\mathbf{A}^{\$}=1\right]$

An output of $\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P},\mathcal{M}]$ is a set of an output of $\mathcal{M}$ and a ciphertext that is defined by xoring a plaintext with an output of $\mathsf{LM\text{-}DAE.PRNG}[\widetilde{P}]$. Thus, the difference is upper-bounded by the probability of distinguishing between the following system and \$:

$$\mathsf{LM\text{-}DAE.Enc1}^{0}[\widetilde{P},\mathcal{M}](A,M) := (\mathcal{M}(A,M),\mathsf{LM\text{-}DAE.PRNG}[\widetilde{P}](\mathcal{M}(A,M),m))$$

where $m=\lceil|M|/n\rceil$. That is,

$$\begin{aligned}&\max_{\mathbf{A}}\Big(\Pr\Big[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}[\widetilde{P},\mathcal{M}]}=1\Big]-\Pr\Big[\mathbf{A}^{\$}=1\Big]\Big)\\ &\le\max_{\mathbf{A}}\Big(\Pr\Big[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}^{0}[\widetilde{P},\mathcal{M}]}=1\Big]-\Pr\Big[\mathbf{A}^{\$}=1\Big]\Big)\ ,\end{aligned}$$

where for each query, $\$(A,M)$ returns a random string of length $|\mathsf{LM\text{-}DAE.Enc1}^{0}[\widetilde{P},\mathcal{M}](A,M)|$. We say the world with $\mathsf{LM\text{-}DAE.Enc1}^{0}[\widetilde{P},\mathcal{M}]$ (resp. \$) is a left (resp. right) world.

The proof for upper bounding the distinguishing probability can be done by the same analysis as the proof of Lemma 1. Hereafter, first, an overview of the proof is given, and then the details of the proof are given.

7.4.1 Overview

As the proof of Lemma 1, this proof uses the coefficient H technique, and we can ensure that $\mathsf{LM\text{-}DAE.Enc1}^0[\widetilde{P}, \mathcal{M}](A, M)$ and \$ are indistinguishable if (1) no collision in $2n$-bit inputs to $\widetilde{P}$ in $\mathsf{LM\text{-}DAE.PRNG}[\widetilde{P}]$ occurs (corresponding with the bad event bad_1 in the proof of Lemma 1) and (2) there is no impossible output for $\widetilde{P}$, that is, the n-bit tweaks are the same and the n-bit input blocks are distinct but the n-bit outputs are the same (corresponding with the bad event bad_2 and bad_3 in the proof of Lemma 1). By using the coefficient H technique, these probabilities are evaluated in the right world, and using the randomness of \$, the probability that the event (1) occurs can be upper-bounded by $\binom{\sigma_\mathcal{E}}{2}/2^{2n} \leq 0.5\sigma_\mathcal{E}^2/2^{2n}$ (as the $2n$-bit inputs are random), and the probability that the event (2) occurs can be upper-bounded by $\binom{\sigma_\mathcal{E}}{2}/2^{2n} \leq 0.5\sigma_\mathcal{E}^2/2^{2n}$ (as the n-bit tweaks and the n-bit outputs are random). Then, assuming that (1) and (2) do not occur, by the same analysis as the proof of Lemma 1, we can ensure that $\mathsf{LM\text{-}DAE.Enc1}^0[\widetilde{P}, \mathcal{M}](A, M)$ and \$ are indistinguishable. Hence, we have

$$\max_{\mathbf{A}} \left(\Pr\left[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}^0[\widetilde{P},\mathcal{M}]} = 1\right] - \Pr\left[\mathbf{A}^{\$} = 1\right] \right) \leq \frac{\sigma_\mathcal{E}^2}{2^{2n}} \ .$$

7.4.2 Strategy, Notation, and Transcript

We upper-bound the probability of distinguishing the left world from the right world, using the coefficient H technique (see the definition and Lemma 3 in Subsection 6.2). Let T_1 be a transcript in the left world obtained by sampling $\widetilde{P}$ and $\mathcal{M}$. Let T_2 be a transcript in the right world obtained by sampling \$. Let $\mathcal{T}$ be a set of all valid transcripts.

For $\alpha \in [q_\mathcal{E}]$, the plaintext length in blocks m at the α-th query is denoted by m_α, and values and variables defined are denoted by using the superscript of (α), e.g., $M^{(\alpha)}, T^{(\alpha)}$, etc.

In the both worlds, $\mathbf{A}$ obtains the following transcript after all queries:

$$\tau := \left\{ (A^{(1)}, M^{(1)}, R^{(1)}, T^{(1)}), \ldots, (A^{(q_\mathcal{E})}, M^{(q_\mathcal{E})}, R^{(q_\mathcal{E})}, T^{(q_\mathcal{E})}) \right\},$$

where $T^{(\alpha)} = \mathcal{M}(A^{(\alpha)}, M^{(\alpha)})$ and $R^{(\alpha)} = \mathsf{LM\text{-}DAE.PRNG}[\widetilde{P}](T^{(\alpha)}, m_\alpha)$ in the left world and $(R^{(\alpha)}, T^{(\alpha)}) = \$(A^{(\alpha)}, M^{(\alpha)})$ in the right world. This analysis uses the following notations.

- $R^{(\alpha)} = \mathrm{Rt}_1^{(\alpha)} \| \cdots \| \mathrm{Rt}_{m_\alpha}^{(\alpha)}$, where $\alpha \in [q_\mathcal{E}]$ and $|\mathrm{Rt}_i^{(\alpha)}| = n$ for $i \in [m_\alpha]$.
- $T^{(\alpha)} = \mathrm{Rt}_0^{(\alpha)} \| \mathrm{Rb}_0^{(\alpha)}$, where $|\mathrm{Rt}_0^{(\alpha)}| = n$ and $|\mathrm{Rb}_0^{(\alpha)}| = n$.
- $\mathrm{Rb}_i^{(\alpha)} = \mathrm{Rt}_i^{(\alpha)} \oplus \pi(\mathrm{Rb}_{i-1}^{(\alpha)})$ for $i \in [m_\alpha]$.

Note that in the left world, $\mathrm{Rt}_i^{(\alpha)}$ is the output of the i-th TRP call and the input of the $(i+1)$-th TRP call, and $\mathrm{Rb}_i^{(\alpha)}$ is the tweak of the $(i+1)$-th TRP call.

7.4.3 Definitions of Good and Bad Transcripts

Bad transcripts $\mathcal{T}_{\mathsf{bad}}$ are defined so that one of the following conditions is satisfied. Good transcripts $\mathcal{T}_{\mathsf{good}}$ are defined as $\mathcal{T}_{\mathsf{good}} = \mathcal{T} \backslash \mathcal{T}_{\mathsf{bad}}$.

- $\mathsf{bad}_1 \Leftrightarrow \exists \alpha, \beta \in [q_\mathcal{E}], i \in [m_\alpha], j \in [m_\beta]$ s.t.
  $(\alpha, i) \neq (\beta, j) \wedge (\mathrm{Rt}_{i-1}^{(\alpha)}, \mathrm{Rb}_{i-1}^{(\alpha)}) = (\mathrm{Rt}_{j-1}^{(\beta)}, \mathrm{Rb}_{j-1}^{(\beta)})$.
- $\mathsf{bad}_2 \Leftrightarrow \exists \alpha, \beta \in [q_\mathcal{E}], i \in [m_\alpha], j \in [m_\beta]$ s.t.
  $(\alpha, i) \neq (\beta, j) \wedge \mathrm{Rt}_{i-1}^{(\alpha)} \neq \mathrm{Rt}_{j-1}^{(\beta)} \wedge \mathrm{Rb}_{i-1}^{(\alpha)} = \mathrm{Rb}_{j-1}^{(\beta)} \wedge \mathrm{Rt}_i^{(\alpha)} = \mathrm{Rt}_j^{(\beta)}$.

bad_1 defines a collision in inputs to TRP. bad_2 defines an impossible output for $\widetilde{P}^0$, that is, the input blocks are distinct and the tweaks are the same, but the output is the same.

7.4.4 Upper Bounding $\Pr[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}]$

Since $\Pr[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}] \leq \Pr[\mathsf{bad}_1] + \Pr[\mathsf{bad}_2]$, we upper bound the two probabilities in the right world $\Pr[\mathsf{bad}_1]$ and $\Pr[\mathsf{bad}_2]$.

Regarding $\Pr[\mathsf{bad}_1]$, we first fix $\alpha, \beta \in [q_{\mathcal{E}}], i \in [m_\alpha], j \in [m_\beta]$ such that $(\alpha, i) \neq (\beta, j)$, and then upper bound $\Pr[(\mathrm{Rt}_{i-1}^{(\alpha)}, \mathrm{Rb}_{i-1}^{(\alpha)}) = (\mathrm{Rt}_{j-1}^{(\beta)}, \mathrm{Rb}_{j-1}^{(\beta)})]$. Regarding $\mathrm{Rt}_{i-1}^{(\alpha)} = \mathrm{Rt}_{j-1}^{(\beta)}$, as $\mathrm{Rt}_{i-1}^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[\mathrm{Rt}_{i-1}^{(\alpha)} = \mathrm{Rt}_{j-1}^{(\beta)}] \leq 1/2^n$. Regarding the condition $\mathrm{Rb}_{i-1}^{(\alpha)} = \mathrm{Rb}_{j-1}^{(\beta)}$, the collision is of the form $\mathrm{Rb}_0^{(\alpha)} = \mathrm{Rb}_{j-1}^{(\beta)}$ if $i = 1$; $\mathrm{Rt}_1^{(\alpha)} \oplus \pi(\mathrm{Rb}_0^{(\alpha)}) = \mathrm{Rb}_{j-1}^{(\beta)}$ if $i = 2$; $\mathrm{Rt}_{i-1}^{(\alpha)} \oplus \pi(\mathrm{Rt}_{i-2}^{(\alpha)} \oplus \pi(\mathrm{Rb}_{i-3}^{(\alpha)})) = \mathrm{Rb}_{j-1}^{(\beta)}$ if $i > 2$. As $\mathrm{Rb}_0^{(\alpha)}$ $(i = 1, 2)$ and $\mathrm{Rt}_{i-2}^{(\alpha)}$ $(i > 2)$ are chosen uniformly at random from $\{0,1\}^n$, we have $\Pr[\mathrm{Rb}_{i-1}^{(\alpha)} = \mathrm{Rb}_{j-1}^{(\beta)}] \leq 1/2^n$. Summing the upper bound for each α, β, i, j gives

$$\Pr[\mathsf{bad}_1] \leq \binom{\sigma_{\mathcal{E}}}{2} \cdot \frac{1}{2^{2n}} \leq \frac{0.5\sigma_{\mathcal{E}}^2}{2^{2n}} \ .$$

Regarding $\Pr[\mathsf{bad}_2]$, we first fix $\alpha, \beta \in [q], i \in [m_\alpha], j \in [m_\beta]$ such that $(\alpha, i) \neq (\beta, j)$, and then upper-bound $\Pr[\mathrm{Rb}_{i-1}^{(\alpha)} = \mathrm{Rb}_{j-1}^{(\beta)} \wedge \mathrm{Rt}_i^{(\alpha)} = \mathrm{Rt}_j^{(\beta)}]$. The analyses of the Rt and Rb collision events are the same as those of $\Pr[\mathsf{bad}_1]$, and thus we have

$$\Pr[\mathsf{bad}_2] \leq \binom{\sigma_{\mathcal{E}}}{2} \cdot \frac{1}{2^{2n}} \leq \frac{0.5\sigma_{\mathcal{E}}^2}{2^{2n}} \ .$$

These upper bounds give

$$\Pr[\mathsf{T}_2 \in \mathcal{T}_{\mathsf{bad}}] \leq \Pr[\mathsf{bad}_1] + \Pr[\mathsf{bad}_2] \leq \frac{\sigma_{\mathcal{E}}^2}{2^{2n}} \ .$$

7.4.5 Lower Bounding $\frac{\Pr[\mathsf{T}_1 = \tau]}{\Pr[\mathsf{T}_2 = \tau]}$

Let $\tau \in \mathcal{T}_{\mathsf{good}}$.

First, $\Pr[\mathsf{T}_2 = \tau]$ is evaluated. In the right world, for each $\alpha \in [q]$, $T^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^{2n}$, and $R^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^{nm_\alpha}$. As $\sum_{\alpha \in [q_{\mathcal{E}}]} m_\alpha = \sigma_{\mathcal{E}}$, we have

$$\Pr[\mathsf{T}_2 = \tau] = \prod_{\alpha=1}^{q_{\mathcal{E}}} \left(\frac{1}{2^{2n}} \cdot \frac{1}{2^{nm_\alpha}} \right) = \frac{1}{2^{2nq_{\mathcal{E}} + \sum_{\alpha \in [q_{\mathcal{E}}]} nm_\alpha}} = \frac{1}{2^{2nq_{\mathcal{E}} + n\sigma_{\mathcal{E}}}} \ .$$

Next, $\Pr[\mathsf{T}_1 = \tau]$ is lower-bounded. By bad_2, one has $\Pr[\mathsf{T}_1 = \tau] > 0$. Let σ_Y be the number of input blocks to $\widetilde{P}^{0,Y}$. For each $\alpha \in [q]$, $T^{(\alpha)}$ is chosen uniformly at random from $\{0,1\}^{2n}$, and thus we have

$$\Pr[\mathsf{T}_1 = \tau] = \left(\frac{1}{2^{2n}} \right)^{q_{\mathcal{E}}} \cdot \prod_{Y \in \{0,1\}^n} \frac{1}{(2^n)_{\sigma_Y}} \geq \left(\frac{1}{2^{2n}} \right)^{q_{\mathcal{E}}} \cdot \prod_{Y \in \{0,1\}^n} \frac{1}{(2^n)^{\sigma_Y}} \ .$$

By $\neg\mathsf{bad}_1$, $\sum_{Y \in \{0,1\}^n} \sigma_Y = \sigma_{\mathcal{E}}$ is satisfied, and thus we have

$$\Pr[\mathsf{T}_1 = \tau] \geq \frac{1}{2^{2nq_{\mathcal{E}} + n\sigma_{\mathcal{E}}}} \ .$$

Finally, these bounds give

$$\frac{\Pr[\mathsf{T}_1 = \tau]}{\Pr[\mathsf{T}_2 = \tau]} \geq 1 \ .$$

7.4.6 Upper Bound of $\max_{\mathbf{A}} \left(\mathbf{Pr}\left[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}^0[\widetilde{P},\mathcal{M}]} = \mathbf{1}\right] - \mathbf{Pr}\left[\mathbf{A}^{\$} = \mathbf{1}\right]\right)$

Putting these bounds into Lemma 3 gives

$$\max_{\mathbf{A}} \left(\Pr\left[\mathbf{A}^{\mathsf{LM\text{-}DAE.Enc1}^0[\widetilde{P},\mathcal{M}]} = 1\right] - \Pr\left[\mathbf{A}^{\$} = 1\right]\right) \leq \frac{\sigma_{\mathcal{E}}^2}{2^{2n}} \ . \tag{4}$$

8 TweSKINNY: SKINNY with Small Additional Tweak

We extend SKINNY-128-256 to process 4 extra bits of tweak by applying the elastic-tweak framework proposed by Chakraborti et al. [CDJ+19a]. The construction is called TweSKINNY-128-256. We first recall the elastic-tweak framework and present some improvements. We then describe the specification of TweSKINNY-128-256 and finally explain the design rationale and security evaluation.

8.1 Elastic-Tweak Framework and Our Improvements

The elastic-tweak is a dedicated method for converting a BC into a TBC with a small tweak, e.g. 4, 8, and 16 bits [CDJ+19a]. The elastic-tweak is defined with four parameters $(\theta_1, \theta_2, \theta_3, \theta_4)$, where an input of size θ_1 bits is expanded to θ_2 bits by using a linear function and then the expanded tweak is added to θ_3 bytes of the internal state in every θ_4 rounds of the target BC. For example, TweAES$[4, 8, 8, 2]$ proposed by Chakraborti et al. [CDJ+19a] converts AES into a TBC to accept a 4-bit tweak, where a 4-bit tweak is expanded to 8 bits by a simple linear code and each bit of the expanded tweak is added to 8 bytes of the state in every 2 rounds.

Regarding the security, as long as the small tweak value is 0, the elastic-tweak does not modify the original scheme. Hence the security concern is only the case where the attacker exploits the tweak. The core argument is that the linear expansion of the small tweak makes the attack difficult, e.g., it ensures differences in many bytes when the tweak has a non-zero difference.

Here we point out several points of the elastic-tweak that can be improved.

A. The expanded tweak must be stored in a register, which requires the memory of the expanded size, or θ_2 bits.

B. The value of the expanded tweak is the same in all places, which may cause a security concern.

C. The tweak addition in every θ_4 rounds requires a selector in the hardware implementation. Note that the designers of the elastic-tweak [CDJ+19a] intentionally make θ_4 sparse because it helps them to quickly evaluate the security.

In our design, we improve the above points as follows.

- Points A and B are solved together. Instead of storing θ_2-bit expanded tweak, we generate it on the fly from the θ_1-bit state by applying an LFSR of θ_1 bits denoted by `LFSR` iteratively. Let tw be a θ_1-bit tweak and $\theta_2 = \ell \cdot \theta_1$. Then, we compute the expanded tweak as $tw \| \mathtt{LFSR}^1(tw) \| \cdots \| \mathtt{LFSR}^{\ell-1}(tw)$.

- For point C, we decided to add the tweak in every round, namely $\theta_4 = 1$, which makes the round function identical for all rounds.

8.2 Specification of TweSKINNY-128-256

The improved elastic-tweak is applied to SKINNY-128-256 [BJK+16], which is a TBC designed by Beierle et al. supporting a 128-bit block and up to 256 bits of a combination of a key and a tweak (tweakey). In our scheme, both the key and the (big) tweak are 128 bits. When $tw = 0$, the computation is exactly the same as SKINNY-128-256, hence the new components introduced by TweSKINNY-128-256 are a linear update of tw in every round and an addition of the expanded tweak to the state. The round function is illustrated in Fig. 13. SKINNY-128-256 consists of 48 rounds. TweSKINNY-128-256 also consists of 48 rounds. Refer to Supplementary material A for the full description.

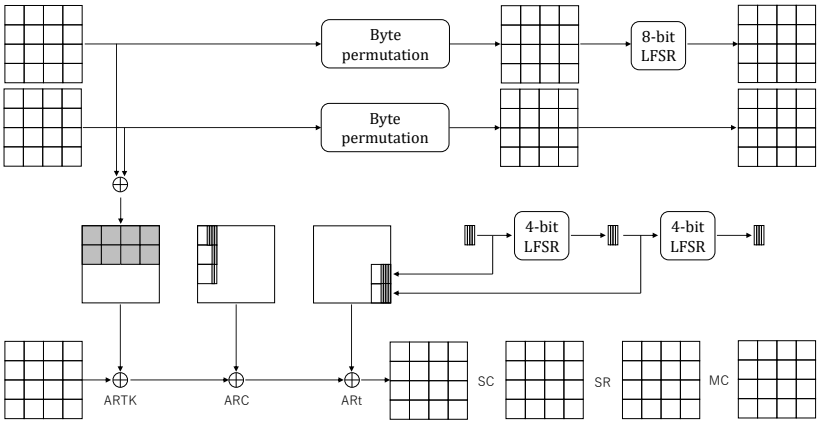

Figure 13: Round Function of TweSKINNY-128-256. Grey bits are the added bits.

SubCells (SC). Apply an 8-bit S-box to all 16 bytes in parallel.

AddRoundTweakey (ARTK). Two 128-bit states are initialized to a 128-bit key and a 128-bit tweak. In each round, 8 bytes in the top 2 rows of the both states are added to the internal state. Then byte positions are permuted in the same way for two 128-bit tweakey states. Finally, each byte value is updated by an 8-bit LFSR only for the second tweaky state.

AddRoundConstnat (ARC). A 4-bit and 2-bit round constant and a fixed 1-bit constant `0x2` are added to 3 bytes in Fig. 13.

AddRoundSmallTweak (ARt). A 4-bit state $\mathcal{T}$ is initialized to a small tweak. In each round, $\mathcal{T}$ is added to the 4 LSBs of the third row in the rightmost column and is updated by $\mathcal{T} \leftarrow \mathtt{LFSR}(\mathcal{T})$, where `LFSR` is the 4-bit LFSR defined as $x_3\|x_2\|x_1\|x_0 \rightarrow x_0 \oplus x_3\|x_3\|x_2\|x_1$. Then, $\mathcal{T}$ is added to the 4 LSBs of the fourth row in the rightmost column and is updated by $\mathcal{T} \leftarrow \mathtt{LFSR}(\mathcal{T})$.

ShiftRows (SR). Bytes in row i are rotated by i positions to right, $i \in \{0, 1, 2, 3\}$.

MixColumns (MC). Four bytes in each column are updated as $(b_0, b_1, b_2, b_3) \rightarrow (b_0 \oplus b_2 \oplus b_3, b_0, b_1 \oplus b_2, b_0 \oplus b_2)$.

8.3 Design Rationale and Security Evaluation

To achieve 128-bit security, LM-DAE requires a TBC that supports a 128-bit block, a 128-bit key and a 132-bit tweak. We first decided to mainly rely on the existing TBC. This motivated us to consider SKINNY-128-256.

The next decision is how to process the extra 4-bit tweak. A natural choice is to use SKINNY-128-384. However, increasing 128 bits of the tweak space only for the 4-bit domain separation overly increases the hardware implementation cost. Moreover, for SKINNY-128-384, security will fall to the so-called TK3 setting, which is weaker than SKINNY-128-256. This motivated us to extend the design of SKINNY-128-256 by applying the elastic-tweak. We also consider that the role of the 4-bit of tweak (for domain separation) is different from the 128-bit tweak, thus developing a dedicated method to handle those 4 bits is reasonable. Indeed, in the mode, the value of the domain separation is usually fixed. In particular, it is fixed to 0 during the encryption. Then the security of TweSKINNY-128-256 is the same as that of SKINNY-128-256 as long as the encryption function is analyzed.

As mentioned above, the elastic tweak has 4 parameters, and θ_1 is fixed to 4 bits from the mode requirement and θ_4 is fixed to 1 to improve the implementation. There are still design spaces for θ_2, θ_3, and the position of θ_3 bytes that are xored during the ARt operation. Following the strategy of the expanded tweak, we set $\theta_2 > \theta_1$. We tested several choices by evaluating the minimum number of active S-boxes with MILP.

A valuable case analysis is depicted in Fig. 14, which adds the expanded tweak to 4 bytes in the last row. We found that the minimum number of active bytes is $4r$ for r rounds.

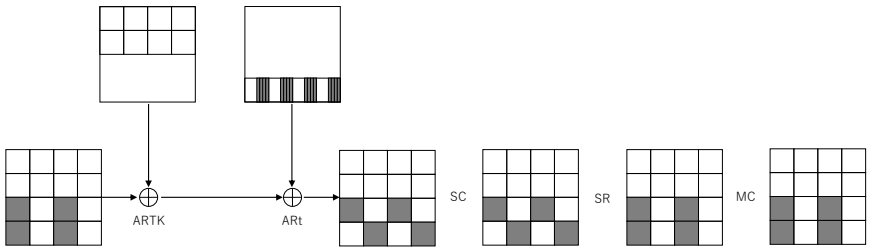

Figure 14: One-Round Iterative Differential in Row-wise Addition. ARC is omitted.

Having 4 active bytes in every round ensures the maximum differential characteristic probability of $(2^{-2})^4 = 2^{-8}$ that ensures 2^{-128} after 16 rounds. This is not a bad choice, but the 1-round iterative characteristic may cause a vulnerability against an unknown cryptanalytic method. This motivated us to add the expanded tweak at least in two rows.[7]

In Table 3, we compare the minimum number of active S-boxes in several constructions. For SKINNY-128-256, we borrowed the number of active S-boxes from the design document [BJK+16]. Attempt 1 is our attempt analyzed in Fig. 14 that enables the 1-round iterative differential characteristic. Attempt 2 is our attempt in which the size of the expanded tweak is 16 bits and the tweak is added to 4 bytes in the ith row and jth column $i, j \in \{2, 3\}$. TweSKINNY-128-256 is our final attempt specified and illustrated in Fig. 13. SK means the single-key. TK1 means the single-key but the (large) tweak has some difference. TK2 implies the related-key-related-tweak setting. In our attempts, by setting $\Delta tw = 0$, the constructions have the same number of active S-boxes as SKINNY-128-256. Hence only numbers for $\Delta tw \neq 0$ are listed. SKINNY-128-256 has only two tweakey states, thus TK3 cannot be used. If we consider using SKINNY-128-384 for the extra 4-bit tweak, TK3 needs to be considered.

Table 3 shows that our final choice is stronger than Attempt 1. Attempt 2 is more secure because of two additional bytes affected by the ARt operation. However, if we compute 1 or 2 more rounds by our final choice, it would reach the similar number of rounds, and we thus adopted our final choice. Finally, compared to the original SKINNY-

[7] We observed that similar 2-round iterative differentials can be built even by adding the tweak to 4 bytes in the third row. We want to avoid adding the tweak in the first two rows in order to avoid the cancellation against the differences in tweakey.

Table 3: Comparison of the Minimum Number of Active S-boxes.

Construction	Setting	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
SKINNY-128-256	SK	1	2	5	8	12	16	26	36	41	46	51	55	58	61	66	75
Attempt 1	SK, $\Delta tw \neq 0$	0	4	11	16	20	24	28	32	36	40	44	48	52	56	60	64
Attempt 2	SK, $\Delta tw \neq 0$	0	4	9	15	23	29	34	39	36	45	50	54	58	64	70	74
TweSKINNY-128-256	SK, $\Delta tw \neq 0$	0	2	5	10	16	22	27	32	35	38	43	50	55	60	65	68
SKINNY-128-256	TK1	0	0	1	2	3	6	10	13	16	23	32	38	41	45	49	54
Attempt 1	TK1, $\Delta tw \neq 0$	0	4	8	12	16	24	29	33	39	43	47	52	56	61	65	-
Attempt 2	TK1, $\Delta tw \neq 0$	0	4	9	15	21	25	30	36	43	48	52	60	65	70	75	81
TweSKINNY-128-256	TK1, $\Delta tw \neq 0$	0	2	5	10	18	22	28	33	39	45	49	56	60	65	73	78
SKINNY-128-256	TK2	0	0	0	0	1	2	3	6	9	12	16	21	25	31	35	40
Attempt 1	TK2, $\Delta tw \neq 0$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Attempt 2	TK2, $\Delta tw \neq 0$	0	4	9	15	20	23	29	34	41	45	51	57	61	68	73	79
TweSKINNY-128-256	TK2, $\Delta tw \neq 0$	0	2	5	10	16	22	27	32	35	38	47	53	58	64	69	76
SKINNY-128-384	TK3	0	0	0	0	0	0	1	2	3	6	10	13	16	19	24	27

128-256, the small tweak slightly reduces the security in the SK setting. However, if the attacker puts a difference both in the tweakey and a small tweak, the cancellation does not occur frequently, and the number of active bytes actually increases from the original SKINNY-128-256. Hence we believe that in the TK1 or TK2 settings, our design via the elastic-tweak does not impose new security vulnerability.

The fact that TweSKINNY is the same as SKINNY for zero tweak allows us to skip analyzing many attacks that do not exploit the additional tweak. The additional tweak space is only 4 bits, hence many attacks cannot work in nature. Integral analysis is an example; collecting 256 tweak values for one byte is impossible.

9 Hardware Implementation

We implement LM-DAE instantiated with TweSKINNY-128-256 in hardware, and compare its performance with the conventional AE schemes PFB with SKINNYe-64-256 and PFB_Plus with SKINNY-128-256; we chose them for their similarity to our scheme and the availability of the detailed performance evaluation [NSS20]. We refer to these instantiations as LM-DAE[TweSKINNY-128-256], PFB_Plus[SKINNYe-64-256], and PFB[SKINNY-128-256]. Although our evaluation is limited to hardware, LM-DAE's low-memory property (see Table 1) is applicable to software implementation and should be useful to implement a compact software that uses a small RAM footprint [MM13].

9.1 Comparing Register Sizes

We first compare the register sizes between the schemes with and without TI because they dominate the circuit area in a compact implementation. We can obtain the non-TI register sizes for 128-bit security by assigning $s = 128$ in Table 2.

Without TI: LM-DAE uses 384 ($= 3 \times 128$) bits without TI, aligned with SUNDAE [BBLT18]. Note, however, that SUNDAE and ESTATE [CDJ+19b] need a 256-bit primitive for $s = 128$, which is uncommon in lightweight cryptography, as discussed in Section 1. Meanwhile, ZAE [IMPS17] with a 128-bit primitive has the larger register size of 896 bits. In the non-deterministic AE category, PFB [NS20] and PFB_Plus [NSS20] also use 384 bits, and they are the smallest.

With TI: LM-DAE uses 896 ($= 7 \times 128$) bits with TI, the smallest among the other DAEs with 128-bit security. The advantage of LM-DAE over SUNDAE comes from its

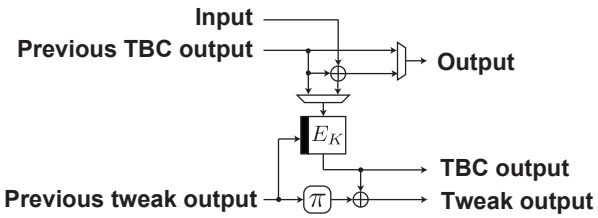

Figure 15: The operational unit that covers all the block-wise operations in LM-DAE

smaller block size, which becomes $\times 3$ in TI [NSS20], while the tweakey part becomes $\times 2$ only. The non-deterministic schemes can have smaller register sizes (e.g., PFB's 768 bits and PFB_Plus' 704 bits) because the tweak storing the public nonce needs no SCA countermeasure.

9.2 Design Policy

We set the circuit area as the primary performance target and prioritize it over latency and throughput; the other AEs can be better than LM-DAE for speed-oriented optimization as summarized in Tables 1 and 2.

For a fair comparison, we follow the design policy of the previous PFB_Plus and PFB implementations [NSS20]. We design a coprocessor that defines a set of commands for processing aligned blocks. The handling of exceptional cases such as padding in a final block, and dispatching operations in an appropriate order is the main processor's responsibility. The design preserves a secret key during its lifetime, and thus there is no need for feeding the key for each block processing.

We implement the design using Verilog at the register-transfer level. We do not make gate-level optimization except explicit instantiation of scan flip-flop, a register with a built-in selector, commonly used in the previous works [MPL+11, BJK+16, NS20, NSS20].

9.3 Implementation

Mode of Operation: LM-DAE's operations on hashing, encryption, and decryption are highly homogeneous, and the functional unit in Figure 15 sufficiently covers all the block-wise processing. This simplicity enables the overhead of conditionally-used circuits and selectors for switching between them to be reduced. Figure 16 shows the datapath diagram of our implementation[8].

Tweakey Assignment: We assign a 128-bit secret key and tweak into TK1 and TK2, respectively, to eliminate the need for an inverse tweakey schedule. TweSKINNY-128-256 with the appropriate number of rounds brings the secret key in the TK1 back to its original value after finishing the on-the-fly scheduling. The TK2 array does not come back to the original state, but we can carry the final TK2 state to the next block processing because LM-DAE integrates the TK2 schedule as π. As a result, we can simplify the tweakey-schedule implementation by eliminating inverse operations.

Tweakey Array: We follow the previous array-based architecture for the tweakey schedule [NSS20], but the absence of the inverse tweakey schedule greatly simplifies the circuit. The TK1 and TK2 arrays are composed of the 16 scan flip-flops with two inputs: one way for a shift register and another for the tweakey schedule (a byte permutation). The previous implementation had yet another byte permutation for an inverse tweakey schedule and an adder for incrementing a counter, which resulted in additional 128-bit selectors for switching between them [NSS20].

[8]The implementation in Figure 16 uses some more AND/selector gates to support initialization that Figure 15 omits for simplicity.

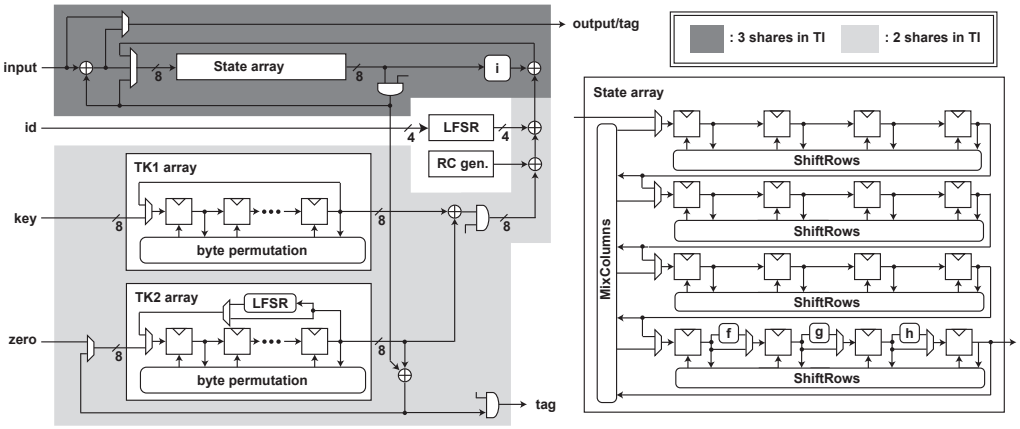

Figure 16: Datapath diagram of LM-DAE instantiated with TweSKINNY.

Table 4: Circuit area breakdown of LM-DAE[TweSKINNY-128-256], PFB_Plus[SKINNYe-64-256], and PFB[SKINNY-128-256]. The PFB_Plus and PFB performances are based on Table 4 of [NSS20].

Component	w/o TI			w/ TI		
	LM-DAE	PFB_Plus	PFB	LM-DAE	PFB_Plus	PFB
Total	**3,717**	4,351	4,400	**8,358**	7,439	8,448
Total/TBC/State	1,102	532	1,098	3,440	1,646	3,517
Total/TBC/Key	1,004	1,268	1,224	2,008	2,620	2,546
Total/TBC/Tweak	1,012	1,551	1,421	2,019	1,551	1,470

State Array: We use the byte-serial architecture following the original SKINNY paper [NSS20, BJK+16]. The state array has 4×4 scan flip-flops connected in two ways: one way for a shift register and another for ShiftRows. The 8-bit SKINNY S-box is decomposed into 4 stages, i.e., $i \circ h \circ g \circ f$, and the state array integrates the first 3 stages for reducing latency.

Elastic Tweak: The elastic tweak is directly achieved as a 4-bit LFSR, similar to the round-constant generator in Figure 16. The LFSR is initialized with a user-supplied domain-separation number and clicks twice a round. There is no need for round-dependent handling, which simplifies a state machine that does not appear in the datapath diagram.

Threshold Implementation: We implement a TI secure up to the first-order attacks, and the decomposed S-box enables uniform sharing with three shares. By following the previous implementation, we also protect the tweak and key schedule [NSS20]. Figure 16 shows the number of shares for each component. The state array needs three shares for the S-box, while the tweakey arrays need only two shares because the tweakey schedule is linear. We need no protection for the public domain-separation bits. The design requires the inputs in the shared representation, and never reconstructs unshared data.

9.4 Performance Evaluation and Comparison

Evaluation: We synthesized the design using Synopsys Design Compiler with the NanGate 45-nm standard cell library [Nan], the same conditions with the previous PFB and PFB_Plus implementations [NSS20]. For a component-wise comparison, we preserve the module hierarchy up to the major components, as summarized in Table 4.

Performance without TI: LM-DAE[TweSKINNY-128-256] achieves 3,717 gates, which

is smaller than both PFB_Plus[SKINNYe-64-256] and PFB[SKINNY-128-256], although they have the same register size in Table 2. The comparison between LM-DAE[TweSKINNY-128-256] with PFB[SKINNY-128-256] clearly shows that the difference comes from the tweakey implementations: LM-DAE[TweSKINNY-128-256]'s TK1 and TK2 are smaller by roughly 200 and 400 gates, respectively. This improvement comes from the elimination of the inverse tweakey schedule and a built-in adder, as discussed in Section 9.3.

Performance with TI: The advantage of the lightweight tweakey arrays preserves in TI, and LM-DAE achieves 8,358 gates, which is comparable to PFB with a smaller register size by 128 bits (see Table 1). LM-DAE should protect a secret tweak as opposed to PFB and PFB_Plus using a public tweak; PFB and PFB_Plus are smaller than LM-DAE by 128 and 192 register bits, respectively. However, the lightweight tweakey arrays made LM-DAE's circuit area comparable to that of PFB by canceling the PFB's register advantage. Meanwhile, we can explain the PFB_Plus[SKINNYe-64-256]'s advantage of 919 gates over LM-DAE[TweSKINNY-128-256] by (i) the even smaller register size and (ii) the advantage of the smaller 4-bit S-box of the underlying SKINNYe-64-256 in TI.

Speed: As a downside, LM-DAE[TweSKINNY-128-256] has longer latency. In the TweSKINNY-128-256 implementation, the state array finishes each round in 21 cycles (16 cycles for the S-box, 4 cycles for MixColumns, and 1 cycle for ShiftRows), and a single TweSKINNY-128-256 call uses 1024 ($=21 \times 48 + 16$) cycles. If we assume that the other TBCs use the same serial architecture that determines the latency by the number of rounds, SKINNY-128-256 and SKINNYe-64-256 use 1024 ($=21 \times 48 + 16$) and 940 ($=21 \times 44 + 16$) cycles, respectively. Since LM-DAE needs two TBC calls for each 16-byte message block (for hash and mac), the latency approaches 128.0 ($=\frac{1024 \times 2}{16}$) cycles/byte. In contrast, the PFB[SKINNY-128-256] and PFB_Plus[SKINNYe-64-256] implementations have 64.0 ($=\frac{1024}{16}$) and 117.5 ($=\frac{940}{8}$) cycles/byte, respectively. LM-DAE[TweSKINNY-128-256]'s latency is exactly the twice of that of PFB[SKINNY-128-256] because of the two-pass message scanning; meanwhile, it is comparable to PFB_Plus[SKINNYe-64-256].

References

[ADMA15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[AHM14] Frederik Armknecht, Matthias Hamann, and Vasily Mikhalev. Lightweight Authentication Protocols on Ultra-Constrained RFIDs - Myths and Facts. In *RFIDSec 2014*, volume 8651 of *LNCS*, pages 1–18. Springer, 2014.

[ALP+19] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A New Primitive for Authenticated Encryption of Very Short Messages. In *ASIACRYPT 2019*, volume 11922 of *LNCS*, pages 153–182. Springer, 2019.

[BBLT18] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. SUNDAE: Small Universal Deterministic Authenticated Encryption for the Internet of Things. *IACR ToSC*, 2018(3):1–35, 2018.

[BDP+14] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Using Keccak technology for AE: Ketje, Keyak and more. SHA-3 2014 Workshop, 2014.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BJK+19] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY-AEAD and SKINNY-Hash v1.1. In *Submission to NIST LWC*, 2019.

[BZD+16] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. *ePrint*, 2016:475, 2016.

[CAE19] CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. Available at https://competitions.cr.yp.to/caesar.html, 2019.

[CDD+19] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR ToSC*, 2019(4):119–146, 2019.

[CDJ+19a] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. Elastic-Tweak: A Framework for Short Tweak Tweakable Block Cipher. ePrint 2019/440, 2019.

[CDJ+19b] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. ESTATE Authenticated Encryption Mode: Hardware Benchmarking and Security Analysis. In *NIST Lightweight Cryptography Workshop 2019*, 2019.

[CDL+19] Anne Canteaut, Sébastien Duval, Ga etan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security version 1.1. In *Submission to NIST LWC*, 2019.

[CJN20] Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the Security of Sponge-type Authenticated Encryption Modes. *IACR Trans. Symmetric Cryptol.*, 2020(2):93–119, 2020.

[DMA17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In *ASIACRYPT*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[DR00] Joan Daemen and Vincent Rijmen. The Block Cipher Rijndael. In *CARDIS'98*, volume 1820 of *LNCS*, pages 277–284. Springer, 2000.

[Har08] D. Harkins. RFC5297: Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). Available at https://tools.ietf.org/html/rfc5297, 2008.

[IKM+19] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Ling Sun. Thank goodness it's friday (TGIF) v.1.0. In *Submission to NIST LWC*, 2019.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *IACR ToSC*, 2020:43–120, 2020.

[IMPS17] Tetsu Iwata, Kazuhiko Minematsu, Thomas Peyrin, and Yannick Seurin. ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication. In *CRYPTO 2017*, volume 10403 of *LNCS*, pages 34–65. Springer, 2017.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[JN16] Ashwin Jha and Mridul Nandi. Revisiting structure graphs: Applications to CBC-MAC and EMAC. *J. Math. Cryptol.*, 10(3-4):157–180, 2016.

[MM13] Mitsuru Matsui and Yumiko Murakami. Minimalism of Software Implementation — Extensive Performance Analysis of Symmetric Primitives on the RL78 Microcontroller. In *FSE 2013*, volume 8424 of *LNCS*, pages 393–409. Springer, 2013.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015.

[Nan] NanGate. NanGate FreePDK45 open cell library. http://www.nangate.com.

[NIS18] NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography, 2018.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In *ICICS 2006*, volume 4307 of *LNCS*, pages 529–545. Springer, 2006.

[NS20] Yusuke Naito and Takeshi Sugawara. Lightweight Authenticated Encryption Mode of Operation for Tweakable Block Ciphers. *IACR TCHES*, 2020(1):66–94, 2020.

[NSS20] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight Authenticated Encryption Mode Suitable for Threshold Implementation. In *EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 705–735. Springer, 2020. The latest version is available at https://eprint.iacr.org/2020/542.

[Pat08] Jacques Patarin. The "Coefficients H" Technique. In *SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, 2008.

[Pie06] Krzysztof Pietrzak. A Tight Bound for EMAC. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006*, volume 4052 of *LNCS*, pages 168–179. Springer, 2006.

[PS16] Thomas Peyrin and Yannick Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In *CRYPTO 2016*, volume 9814 of *LNCS*, pages 33–63. Springer, 2016.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, 2006.

[Sön19] Meltem Sönmez. On the NIST Lightweight Cryptography Standardization. In *23rd Workshop on Elliptic Curve Cryptography (ECC 2019)*, 2019.

Supplementary Material

A Supplementary Material: Details of SKINNY-128-256

The major part of the specification of TweSKINNY-128-256 relies on SKINNY-128-256 [BJK+16]. Here, we give the details of SKINNY-128-256.

S-box

Let $x_0, \ldots, x_7$ represent the eight inputs bits of the S-box (x_0 being the least significant bit), it iteratively applies the following two operations four times:

$$(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \rightarrow (x_7, x_6, x_5, x_4 \oplus (\overline{x_7 \vee x_6}), x_3, x_2, x_1, x_0 \oplus (\overline{x_3 \vee x_2})),$$
$$(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \rightarrow (x_2, x_1, x_7, x_6, x_4, x_0, x_3, x_5).$$

The table representation of 8-bit S-box [BJK+16] is as follows.

```
0x65,0x4c,0x6a,0x42,0x4b,0x63,0x43,0x6b,0x55,0x75,0x5a,0x7a,0x53,0x73,0x5b,0x7b,
0x35,0x8c,0x3a,0x81,0x89,0x33,0x80,0x3b,0x95,0x25,0x98,0x2a,0x90,0x23,0x99,0x2b,
0xe5,0xcc,0xe8,0xc1,0xc9,0xe0,0xc0,0xe9,0xd5,0xf5,0xd8,0xf8,0xd0,0xf0,0xd9,0xf9,
0xa5,0x1c,0xa8,0x12,0x1b,0xa0,0x13,0xa9,0x05,0xb5,0x0a,0xb8,0x03,0xb0,0x0b,0xb9,
0x32,0x88,0x3c,0x85,0x8d,0x34,0x84,0x3d,0x91,0x22,0x9c,0x2c,0x94,0x24,0x9d,0x2d,
0x62,0x4a,0x6c,0x45,0x4d,0x64,0x44,0x6d,0x52,0x72,0x5c,0x7c,0x54,0x74,0x5d,0x7d,
0xa1,0x1a,0xac,0x15,0x1d,0xa4,0x14,0xad,0x02,0xb1,0x0c,0xbc,0x04,0xb4,0x0d,0xbd,
0xe1,0xc8,0xec,0xc5,0xcd,0xe4,0xc4,0xed,0xd1,0xf1,0xdc,0xfc,0xd4,0xf4,0xdd,0xfd,
0x36,0x8e,0x38,0x82,0x8b,0x30,0x83,0x39,0x96,0x26,0x9a,0x28,0x93,0x20,0x9b,0x29,
0x66,0x4e,0x68,0x41,0x49,0x60,0x40,0x69,0x56,0x76,0x58,0x78,0x50,0x70,0x59,0x79,
0xa6,0x1e,0xaa,0x11,0x19,0xa3,0x10,0xab,0x06,0xb6,0x08,0xba,0x00,0xb3,0x09,0xbb,
0xe6,0xce,0xea,0xc2,0xcb,0xe3,0xc3,0xeb,0xd6,0xf6,0xda,0xfa,0xd3,0xf3,0xdb,0xfb,
0x31,0x8a,0x3e,0x86,0x8f,0x37,0x87,0x3f,0x92,0x21,0x9e,0x2e,0x97,0x27,0x9f,0x2f,
0x61,0x48,0x6e,0x46,0x4f,0x67,0x47,0x6f,0x51,0x71,0x5e,0x7e,0x57,0x77,0x5f,0x7f,
0xa2,0x18,0xae,0x16,0x1f,0xa7,0x17,0xaf,0x01,0xb2,0x0e,0xbe,0x07,0xb7,0x0f,0xbf,
0xe2,0xca,0xee,0xc6,0xcf,0xe7,0xc7,0xef,0xd2,0xf2,0xde,0xfe,0xd7,0xf7,0xdf,0xff
```

Tweakey Update

After eight bytes in top two rows are added to the state, 16-byte positions are permuted by applying the following permutation to both 128-bit tweakey states;

$$(0, \ldots, 15) \rightarrow (9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7),$$

where a byte position $4i + j, 0 \leq i, j \leq 3$ corresponds to the ith row and the jth column of the 4×4 representation of the state. Then, every cell of the first and second rows for 128-bit tweak are updated with the following LFSR.

$$(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \rightarrow (x_6, x_5, x_4, x_3, x_2, x_1, x_0, x_7 \oplus x_5),$$

where x_0 stands for the LSB of the cell.

Round Constant

A 6-bit affine LFSR, whose state is denoted $(rc_5, rc_4, \ldots, rc_0)$ (with rc_0 being the least significant bit), is used to generate round constants. Its update function is defined as:

$$(rc_5, rc_4, rc_3, rc_2, rc_1, rc_0) \rightarrow (rc_4, rc_3, rc_2, rc_1, rc_0, rc_5 \oplus rc_4 \oplus 1).$$

The six bits are initialized to zero and updated before use in a given round. Let c_0, c_1, c_2 be the byte value added to the first, the second, and the third rows of the leftmost column, respectively. Those are computed as

$$(c_0, c_1, c_2) = (0\|0\|0\|0\|rc_3\|rc_2\|rc_1\|rc_0, 0\|0\|0\|0\|0\|0\|rc_5\|rc_4, \texttt{0x2}).$$