TEDT, a Leakage-Resistant AEAD Mode for High Physical Security Applications

Francesco Berti¹, Chun Guo¹,², Olivier Pereira¹, Thomas Peters¹ and François-Xavier Standaert¹

¹ ICTEAM/ELEN/Crypto Group, UCL, Louvain-la-Neuve, Belgium
² School of Cyber Science and Technology and Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University

Abstract. We propose TEDT, a new Authenticated Encryption with Associated Data (AEAD) mode leveraging Tweakable Block Ciphers (TBCs). TEDT provides the following features: (i) It offers full leakage-resistance, that is, it limits the exploitability of physical leakages via side-channel attacks, even if these leakages happen during every message encryption and decryption operation. Moreover, the leakage integrity bound is asymptotically optimal in the multi-user setting. (ii) It offers nonce misuse-resilience, that is, the repetition of nonces does not impact the security of ciphertexts produced with fresh nonces. (iii) It can be implemented with a remarkably low energy cost when strong resistance to side-channel attacks is needed, supports online encryption and handles static and incremental associated data efficiently. Concretely, TEDT encourages so-called leveled implementations, in which two TBCs are implemented: the first one needs strong and energy demanding protections against side-channel attacks but is used in a limited way, while the other only requires weak and energy-efficient protections and performs the bulk of the computation. As a result, TEDT leads to more energy-efficient implementations compared to traditional AEAD schemes, whose side-channel security requires to uniformly protect every (T)BC execution.

Keywords: Authenticated encryption · leakage-resilience/resitance · beyond-birthday bound · multi-user security · masking countermeasure · leveled implementations

1 Introduction

The development of Authenticated Encryption with Associated Data (AEAD) schemes has been an area of extremely active research since the beginning of this millennium. Numerous prominent designs have emerged and, on top of the traditional confidentiality and integrity requirements [KY01, BN08], a number of desirable functional and (sometimes conflicting) security properties emerged. This paper proposes TEDT, a new AEAD mode for tweakable block ciphers that primarily aims at a high efficiency when a strong resistance to side-channel attacks is needed, which are among the most practical threats against cryptographic implementations, as highlighted in a recent white paper [ABB⁺, chapter 1.1] – see also [EKM⁺08, MBKP11, ZYSQ13, BGRV15, GPT15, GST17, DK18].

Every time an encryption or a decryption operation takes place, some side-effects may be observable, which can leak information on the internal state of a computing device, including keys: these can be timing, power consumption and electromagnetic radiation measurements. Such attacks can be mounted in two main flavors which, in the context of power consumption, are called Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [MOP07]. In an SPA, an attacker takes advantage of the leakages resulting from a single input (message) provided for encryption, with measurements that are possibly repeated multiple times in order to remove the noise in measurements. A DPA exploits...
the leakages resulting from multiple inputs, which all provide new information about the internal state of the device, reducing the computational secrecy of this state at a rate that is exponential in the number of distinct inputs.

Standard AEAD are typically susceptible to the mounting of a DPA attack: for instance, prominent modes like OCB [RBBK01], CCM [WHF03], or GCM [MV04] all evaluate a block cipher used with a single key on a distinct input for each message block, which is the exact setting in which DPA applies.

The protection against side-channel attacks is then typically left to engineers who will implement the block ciphers and other components of the AEAD in a way that limits the leakages as much as possible. One of the most popular countermeasures is masking [CJRR99], in which the internal state of the device is secret-shared into a number of pieces (leading to so-called higher-order security under some noise and independence assumptions [DFS19]), which are then used for the computation. The strong protection of a block cipher against DPA usually decreases the standard performance metrics of both software and hardware implementations by orders of magnitude compared to non-protected implementations [GR17, GMK17]. The overheads for the full AEAD mode are of the same order, since all message blocks are processed by the strongly protected block cipher.

Another approach consists in designing leakage-resilient or leakage-resistant modes of operations [BMOS17, BKP+18, BPPS17, DEM+17]. These modes, which often come with some computational overheads in the black-box setting, e.g., require more block cipher calls than a standard AEAD or require more keying material, aim at considerably reducing the effect of leakages and the possibility to mount a DPA. A first classical ingredient is to use some form of key update or re-keying [DP08, YSPY10] in order to make sure that each execution of a block cipher leaks about a different key, hence effectively leaving the adversary with the possibility to mount an SPA only. A second common ingredient, required in modes aiming at integrity properties, is that the decryption/verification of the validity of a ciphertext/MAC does not require computing the correct value of the authentication tag [BMOS17, BKP+18]. This prevents attacks in which an adversary uses a verification oracle, which it repeatedly queries with a forged message or ciphertext and an invalid tag in order to obtain leakages about the correct tag.

Leakage-resistant modes can also be designed in such a way that they are amenable to so-called leveled implementations, in which different implementations of the mode components (e.g., block ciphers) are used. On the one hand, a leveled implementation relies on the limited use of highly protected components. In practice, these components are at least expected to be secure against DPA, thanks to protections like masking. Theoretically, they are usually modeled as leak-free for simplicity. On the other hand, it will be tolerated that the rest of the components, which would perform the bulk of the computation, continuously leak a certain amount of information to the adversary every time they are used, hence requiring very limited protections, or even no specific protection at all depending on the platform. For example, shuffled implementations could be considered in case of mid-range devices [MSGR10, VMKS12], and plain unprotected (parallel) implementations could even be sufficient in case of hardware devices [BGSD10]. The expected benefits of this leakage-resistant approach are twofold.

First, they can lead to more efficient implementations for a given level of resistance to side-channel attacks. Indeed, even if these modes come with apparently heavier requirements in the black-box world (TEDT requires 4 calls of a TBC per message block), this cost is expected to be largely compensated by the more limited use of side-channel countermeasures that is required. For example, using the cycle counts of the higher-order masked implementations in ARM 32-bit devices from the recent work of Goudarzi and

---

1 We use the term leakage-resistant for security definitions of confidentiality that allow all the computations (including the computation of the “challenge ciphertext”) to leak, and the term leakage-resilient for security definitions that exclude it, following the terminology in [GPPS19a, Sta19].
Rivain at Eurocrypt 2017 [GR17], we show that TEDT leads to reduced cycle counts (hence more energy efficiency) compared to a uniformly protected implementation of OCB already with two shares, and that the factor of gain approximately reaches \( \frac{\ell + 2}{2} \) for messages of \( \ell \) blocks as the number of shares in the masking schemes increases. For “reasonable” number of shares (given the high security goal of TEDT), like four to eight, the gains can reach factors larger than ten for moderate size messages (like \( \ell = 100 \)). We expect similar energy gains to be observed in hardware (since the cost of masking is in general quadratic).

Second, the security reductions that come with the definition of leakage-resistant modes bring clear requirements on the specific blocks to implement. This considerably simplifies the task of designers and evaluation laboratories, and can then also increase the confidence that can be placed in the result of these evaluations. The reductions also clarify the effect of the failure of some components. For instance, in the case of TEDT, we show that weakly protected components that would leak their internal state in full through an SPA would break confidentiality, but would have no impact on ciphertext integrity. In the context of confidentiality, we reduce the security requirements on these weakly protected components, in encryption and decryption, to two simple security games that require evaluating the effect of at most two leakages of a single TBC, a task that is considerably simpler than evaluating what can be derived from millions of leakages of a full mode.

Besides, TEDT also comes with the following important benefits:

- TEDT avoids the so-called multi-user security degradation, i.e., its security bounds do not degrade significantly as the number of users increases. Such a feature is typically hard to avoid massively parallel attacks that could occur under a mass-surveillance adversary [BT16], but are also potentially applicable in an IoT context where an adversary tries to take control of some physical infrastructure. While it is known that increasing the length of secret keys and randomizing nonces can help avoid such degradation, we achieve this using a public key (used in addition to the usual secret key). This approach might be of independent interest.

- Using a TBC with \( n \)-bit blocks, TEDT offers asymptotically optimal multi-user (mu) integrity against \( 2^n / n^2 \) queries up to \( \approx 2^n \) users, even with nonce misuse and full side-channel leakages. Note that such a “Beyond Birthday Bound” (BBB, beyond \( 2^{n/2} \)) security was highlighted in [GL17] and [ABB+, chapter 1.2].

- TEDT offers nonce misuse-resilience in the sense of Ashur et al. [ADL17]. Misuse-resilience guarantees that repeated nonces do not have an impact on the security of messages that are encrypted with fresh nonces, a property that is not satisfied by many standard modes of operation. We do not aim for nonce misuse-resistance [RS06], a stronger form of protection that requires that security is maintained even for ciphertexts produced with repeated nonces, provided that distinct messages are encrypted. Misuse-resistance requires two successive passes on messages for encryption, which creates additional latency and memory requirements, and is expected to be infeasible to achieve in most settings in which the encryption process leaks [GPPS19a] (as also discussed in supplementary material, Appendix A).

- Under very conservative leakage assumptions (see later), TEDT is proved CCA secure with misuse-resilience and decryption leakages, under the definitions in [GPPS19a]. While the conservative assumptions only enable a birthday-type \( n / 2 \)-bit security, the security is again retained up to \( \approx 2^n \) users (i.e., no degradation).

TEDT analysis. TEDT runs a keyless hash function from the underlying TBC. By this, the TBC has to be as strong as an ideal TBC, i.e., a randomly selected TBC. Strong dedicated TBC algorithms, e.g., Skinny-128-256, and Deoxys-BC-256, could be used to instantiate TEDT. On the other hand, many common block cipher-based TBCs such as the
XEX construction are not sufficient (as justified in supplementary material, Appendix B). While as debated as the random oracle model [CGH98], this ideal model also gives a clear and rigorous way to measure the offline computation in terms of the number of adversarial queries to the ideal TBC, as argued in previous works [BT16, BHT18]. In fact, non-degrading mu security results typically rely on the ideal model [Tes15], and for some constructions (e.g., XGCM [BT16] and its underlying FX-key length extension [KR01]) it can only be proved effective in the ideal model. Still, most of the security properties of TEDT can also be proven in the standard model. This analysis then leads to weaker security bounds, which is a well-known artifact of the proof techniques on re-keying designs and rarely relates to actual weaknesses [ST16, Men17] (this constitutes another equally important reason for relying on ideal model analysis).

In terms of leakages, the ideal cipher model prompts us to make very simple assumptions. Our weakly protected components might even leak their full state as far as authentication is concerned, which follows [BKP+18], while they only need to satisfy hard-to-invert leakages for confidentiality, which follows [YSPY10] and appears theoretically minimal & practically measurable (which we believe is essential for modes in use). Our strongly protected components are modeled as leak-free, that is, hiding their key and, in the case of confidentiality, also mildly hiding their output, which follows [PSV15]. We additionally provide analyzes in the standard model in supplementary material, Appendix C: they require to make a stronger (physical) assumption of leakage simulatability [SPY13] regarding the weakly protected components. Interestingly, these analyzes lead to comparable bounds.

**Organization.** Preliminaries and security definitions with leakage in the multi-user setting are in Section 2. We detail and motivate our techniques step by step in Section 3, which leads to the specification of TEDT in Section 4. We give a thorough analysis of the leakage integrity (resp. confidentiality) of TEDT in Section 5 (resp. Section 6). We conclude the paper with a discussion of performances and related works attempting to model and realize secure authenticated encryption in a leakage setting in Section 7.

To better explain and to further illustrate the advantage of TEDT we provide a black-box CCA Ssecurity in supplementary material, Appendix D.

## 2 Preliminaries

The size or the length of a bit string \(x \in \{0,1\}^*\), denoted \(|x|\), is the integer \(a\) such that \(x \in \{0,1\}^a\); an a-bit string has length a. Let \(n\) be a non-negative integer so that, when \(n\) is clear in the context, given a bit string \(x \in \{0,1\}^*\), the padding \(x\|0^n\) is the smallest bit string containing the prefix \(x\) only followed, if necessary, by 0’s and whose length is a multiple of \(n\). Given \(x \in \{0,1\}^n\) and \(1 \leq a \leq n\), \(x[a]\) is the a-bit prefix of \(x\). We denote by \([\text{num}]\text{size}\) the size-bit binary encoding of the integer \(\text{num}\). We denote by a \((q_1, \ldots, q_\omega, t)\)-bounded adversary a probabilistic algorithm that has access to \(\omega\) oracles, \(O_1, \ldots, O_\omega\), can make at most \(q_i\) queries to its i-th oracle \(O_i\), and can perform computation bounded by running time \(t\). A leaking version of an algorithm \(\text{Algo}\) is denoted \(\text{LAlgo}\). It runs both \(\text{Algo}\) and a leakage function \(\text{Lalgo}\) which captures the information given by an implementation of \(\text{Algo}\) during its execution. \(\text{LAlgo}\) returns the outputs of both \(\text{Algo}\) and \(\text{Lalgo}\) which all take the same input.

### 2.1 Primitives

A Tweakable Block Cipher (TBC) with key space \(\{0,1\}^\kappa\), tweak space \(\{0,1\}^t\), and domain \(\{0,1\}^n\) is a mapping \(\text{TE} : \{0,1\}^\kappa \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n\) such that for any key \(K \in \{0,1\}^\kappa\) and any tweak \(T \in \{0,1\}^t\), \(X \mapsto \text{TE}(K,T,X)\) is a permutation of \(\{0,1\}^n\). We call such TBCs \((\kappa, t, n)\)-TBC. Similarly, we denote \((\kappa, n)\)-block cipher those with
As our model, we extend important existing leakage integrity notions of Berti et al. [BPPS17] and confidentiality notions of Guo et al. [GPPS19a] to the multi-user setting.

In this paper, we focus on nonce-based authenticated encryption schemes with associated data (AEAD), which is defined as a tuple $\text{AEAD} = (\text{Enc}, \text{Dec})$ such that:

- $\text{Enc}: \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \rightarrow \mathcal{C}$ maps a key selected from $\mathcal{K}$, a nonce from $\mathcal{N}$, blocks of associated data from $\mathcal{AD}$, and a message from $\mathcal{M}$ to a ciphertext in $\mathcal{C}$.
- $\text{Dec}: \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\bot\}$ maps a key from $\mathcal{K}$, a nonce from $\mathcal{N}$, blocks associated data from $\mathcal{AD}$, and a ciphertext from $\mathcal{C}$ to a message in $\mathcal{M}$ that is the decryption of the ciphertext, or to a special symbol $\bot$ if integrity checking fails.

We consider AEAD schemes with constant stretch, for which the message size $\ell_m$ uniquely determines the ciphertext size $\ell_c = \ell_m + \text{oh}$, where the constant $\text{oh}$ is the overhead. $\mathcal{C}_{\text{enc}}$ denotes the set of all the ciphertexts encrypting $\ell_m$-size messages. Given a key $k \leftarrow \mathcal{K}$, $\text{Enc}_k(N, A, M) := \text{Enc}(k, N, A, M)$ and $\text{Dec}_k(N, A, C) := \text{Dec}(k, N, A, C)$ are deterministic functions whose implementations may be probabilistic. Nonce-based AEAD must be correct meaning that for any key $k \leftarrow \mathcal{K}$ and any triple $(N, A, M) \in \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$, $\text{Dec}_k(N, A, \text{Enc}_k(N, A, M)) = M$. Since we only focus on correct nonce-based authenticated encryption with associated data in this paper, we will often simply refer to it as authenticated encryption in the following.

### 2.2 Leakage Security of AEAD in the Multi-User Setting

As our model, we extend important existing leakage integrity notions of Berti et al. [BPPS17] and confidentiality notions of Guo et al. [GPPS19a] to the multi-user setting.

In face of a leakage adversary, separate definitions for integrity and confidentiality potentially offer more gradual degradation. This relies on the feature of physically observable cryptography that unpredictability is much easier to ensure than indistinguishability [MR04], which naturally splits the level of confidence we might expect to achieve both notions. Here, we will focus on leaking AEAD with nonce-misuse-resistant integrity and nonce-misuse-resistant confidentiality (as mentioned earlier, nonce-misuse-resistant confidentiality seems unachievable, see Appendix A). To formalize the leakage depending on an implementation, AEAD is associated to both an encryption leakage function $\text{L}_{\text{enc}}$ and a decryption leakage function $\text{L}_{\text{dec}}$. Berti et al. defined a leakage integrity notion Ciphertext Integrity with Misuse-resistance and (encryption & decryption) Leakage in [BKP+18, BPPS17], which is denoted $\text{CIIM2}$. In some sense, the definition is obtained by enhancing the traditional INT-CTX security game with encryption and decryption leakage. Here we further extend it to multi-user setting, denoted $\text{muCIIM2}$. Formally, given a nonce-based authenticated encryption $\text{AEAD} = (\text{Enc}, \text{Dec})$ with leakage function pair $\mathbf{L} = (\text{L}_{\text{enc}}, \text{L}_{\text{dec}})$, the multi-user ciphertext integrity advantage with misuse-resistance and leakage of an adversary $\mathcal{A}$ against AEAD with $u$ users is

$$\text{Adv}_{\text{AEAD},u}^{\text{muCIIM2}} \overset{\text{def}}{=} \Pr \left[\mathcal{A}^{\text{L}_{\text{Enc}}, \text{L}_{\text{Dec}}\mathbf{IC}, \mathbf{IC}^{-1}} \Rightarrow 1\right] - \Pr \left[\mathcal{A}^{\text{L}_{\text{Enc}}, \text{L}_{\text{Dec}}\mathbf{IC}, \mathbf{IC}^{-1}} \Rightarrow 1\right],$$

where the probability is taken over the $u$ user keys $\mathbf{K} = (K_1, \ldots, K_u)$, with $K_i \leftarrow \mathcal{K}$, over $\mathcal{A}$'s random tape and the ideal TBC $\mathbf{IC}$, and where (for $1 \leq i \leq u$):

- $\text{L}_{\text{Enc}}(i, N, A, M)$: outputs the cipher $\text{Enc}_{K_i}(N, A, M)$ and the corresponding leakage trace $\text{L}_{\text{enc}}(K_i, N, A, M)$;

\[\text{L}_{\text{dec}}\]

\[\text{L}_{\text{enc}}\]

We follow Bellare and Tackmann [BT16] and highlight IC in the definition.
Initialization: generates $u$ secret keys $K_1, \ldots, K_u \leftarrow \mathcal{K}$ and sets $\mathcal{E}_ch, \mathcal{E}_1, \ldots, \mathcal{E}_u \leftarrow \emptyset$.

Leaking encryption queries: $\mathcal{A}^L$ gets adaptive access to $\mathsf{LEnc}(\cdot, \cdot, \cdot, \cdot)$, 

$\mathsf{LEnc}(i, N, A, M)$ outputs $\bot$ if $(i, N, *, *) \in \mathcal{E}_ch$, else computes $C \leftarrow \mathsf{Enc}_{K_i}(N, A, M)$ and $\text{leak}_e \leftarrow \mathsf{Lenc}(K_i, N, A, M)$, updates $\mathcal{E}_i \leftarrow \mathcal{E}_i \cup \{N\}$ and finally returns $(C, \text{leak}_e)$.

Leaking decryption queries: $\mathcal{A}^L$ gets adaptive access to $\mathsf{LDec}(\cdot, \cdot, \cdot, \cdot)$, 

$\mathsf{LDec}(i, N, A, C)$ outputs $\bot$ if $(i, N, A, C) \in \mathcal{E}_ch$, else computes $M \leftarrow \mathsf{Dec}_{K_i}(N, A, C)$ and $\text{leak}_d \leftarrow \mathsf{Ldec}(K_i, N, A, C)$ and returns $(M, \text{leak}_d)$.

Challenge queries: on possibly many occasions $\mathcal{A}^L$ submits $(i, N_{ch}, A_{ch}, M^0, M^1)$,

If $M^0$ and $M^1$ have different (block) length or $N_{ch} \in \mathcal{E}_i$ or $(i, N_{ch}, *, *) \in \mathcal{E}_ch$, returns $\bot$; Else computes $C^b \leftarrow \mathsf{Enc}_{K_i}(N_{ch}, A_{ch}, M^b)$ and $\text{leak}_e^b \leftarrow \mathsf{Lenc}(K_i, N_{ch}, A_{ch}, M^b)$, updates $\mathcal{E}_ch \leftarrow \mathcal{E}_ch \cup \{(i, N_{ch}, A_{ch}, C^b)\}$ and finally returns $(C^b, \text{leak}_e^b)$.

Decryption challenge leakage queries: $\mathcal{A}^L$ gets adaptive access to $\mathsf{Ldech}(\cdot, \cdot, \cdot, \cdot)$, 

$\mathsf{Ldech}(i, N_{ch}, A_{ch}, C^b)$ computes and outputs $\text{leak}_d^b \leftarrow \mathsf{Ldec}(K_i, N_{ch}, A_{ch}, C^b)$ if $(i, N_{ch}, A_{ch}, C^b) \in \mathcal{E}_{ch}$; Else it outputs $\bot$.

Finalization: $\mathcal{A}^L$ outputs a guess bit $b'$ which is defined as the output of the game.

Figure 1: The $\mathsf{PrivK}^{\mu\text{CCAmL2},b}_{\mathcal{A},\text{AEAD},L,u}$ game.

- $\mathsf{LDec}(\cdot, \cdot, \cdot)$: outputs $(\mathsf{Dec}_{K_i}(N, A, C), \mathsf{Ldec}(K_i, N, A, C))$;
- $\mathsf{LDec}(\cdot, \cdot, \cdot, \cdot)$: computes $\text{leak}_d \leftarrow \mathsf{Ldec}(K_i, N, A, C)$ and if $C$ is an output of some leaking encryption query $(i, N, A, M)$ for some $M$ outputs $(M, \text{leak}_d)$, else outputs $(\bot, \text{leak}_d)$.

To capture an achievable level of nonce robustness in confidentiality, Guo et al. formalized CCAmL2, after chosen-ciphertext security with misuse-resilience and leakage. In a high level, this notion is a leakage CCA security with "fresh nonces" [GPPS19a]. Such a level of nonce robustness was borrowed from Ashur et al. who formalized nonce-misuse-resilience for integrity, CPA, and CCA security [ADL17].

Below we define the mu extension $\mu\text{CCAmL2}$: given an authenticated encryption $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ with leakage function pair $L = (\mathsf{Lenc}, \mathsf{Ldec})$, the multi-user chosen-ciphertext advantage with misuse-resilience and leakage of an adversary $\mathcal{A}$ against $\mathsf{AEAD}$ with $u$ users is

$$\text{Adv}_{\mathcal{A},\text{AEAD},L,u}^{\mu\text{CCAmL2}} \overset{\text{def}}{=} \left| \Pr \left[ \mathsf{PrivK}^{\mu\text{CCAmL2},0}_{\mathcal{A},\text{AEAD},L} \Rightarrow 1 \right] - \Pr \left[ \mathsf{PrivK}^{\mu\text{CCAmL2},1}_{\mathcal{A},\text{AEAD},L} \Rightarrow 1 \right] \right|,$$

where the security game $\mathsf{PrivK}^{\mu\text{CCAmL2},b}_{\mathcal{A},\text{AEAD},L}$ is defined in Figure 1.

When the context is clear, we will refer to the $\mu\text{CCAmL2}$ advantage (also in the ideal TBC model) with the next less formal notation

$$\text{Adv}_{\mathcal{A},\text{AEAD},L,u}^{\mu\text{CCAmL2}} \overset{\text{def}}{=} \left| \Pr \left[ \mathsf{A}^{\mathsf{Lenc},\mathsf{Ldec},L_{\text{dech}},L_{\text{dec}},\mathsf{IC},\mathsf{IC}^{-1},\mathsf{IC}^{-1}} \Rightarrow 1 \right] - \Pr \left[ \mathsf{A}^{\mathsf{Lenc},\mathsf{Ldec},L_{\text{dech}},L_{\text{dec}},\mathsf{IC},\mathsf{IC}^{-1}} \Rightarrow 1 \right] \right|,$$

which allow us to talk about the 1st oracle, the 2nd oracle, and so on.

The CCAmL2 notion is more or less the traditional CCA notion enhanced with encryption and decryption leakage. The presence of $L_{\text{dech}}$, which provides decryption leakage to challenge queries, follows the single-user version of Guo et al. [GPPS19a]: their intention was to capture the informativeness of valid decryption leakage, and it was motivated by applications such as secure bootloading [OC15], in which decrypting valid ciphertexts may leak crucial information. It was proved in [GPPS19a] that the combination
Figure 2: The AE mode EDT. \( E \) denotes a classical block cipher. EDT is a block cipher-based AEAD mode. The two dark blocks indicate two “leak-free” executions of \( E \), while the others are leaky. \( p_A \) and \( p_B \) could be any two distinct constants, while in section 3.1 we consider the simplest case \( p_A = 0 \) and \( p_B = 1 \).

CIML2 + CCAmL2 is not implied by other combinations of any natural relaxation of these two notions. To provide a high leakage-resistant guarantee, an AEAD should satisfy both CIML2 and CCAmL2 (while in this paper we consider the \( \mu \) extensions of both).

Note that leakage CCA is not defined in the “real-or-random” form due to the conceptual difficulty to define the leakage of the idealized (random) objects that are used in such definitions [GPPS19a, Sta19].

3 Background and Design Considerations

Starting from EDT [BPPS17], a nonce-based authenticated encryption mode without associated data (see Figure 2), we detail the different techniques to concretize our TEDT mode and improve the level of integrity and confidentiality.

3.1 Overview of the Starting Point EDT

While composing leakage-resistant building blocks does not necessarily lead to a leakage resistant scheme, EDT is an Encrypt-then-MAC (EtM) composition of a re-keying encryption due to [PSV15] and a Hash-then-MAC, where the hash function is keyless and the fixed-length MAC is a Strong PseudoRandom Permutation (SPRP).

While going one step further in the direction of a fully analyzed practical mode with tight security bounds maintained in a multi-user setting, TEDT inherits some of the core ideas of EDT for achieving security under leakage.\(^4\)

Re-keying against Side-Channel Key-Recovery. Re-keying prevents DPA by deriving session keys and by consistently refreshing the key, so that the attacker cannot collect side-channel leakage on the key in use during cryptographic operations with different inputs. Concretely, for EDT, upon each encryption, a new session key \( k_0 \) is computed via an IV-based Key-Derivation Function (KDF), i.e., \( k_0 \leftarrow KDF_K(N) \). This KDF is heavily protected, so that the master key \( K \) is kept safe. Therefore, as long as the IV is fresh, the resulting encryption process generates a new internal state that has not been affected by the previously observed leakage. Then, the key \( k_0 \) is served to a block cipher \( E \) to produce a key stream block \( y_1 \leftarrow E_{k_0}(1) \). In parallel, \( k_0 \) is refreshed into a new ephemeral key \( k_1 \leftarrow E_{k_0}(0) \). This process is repeated until the number of key stream blocks matches the number of message blocks. In all, upon a message of \( \ell \) blocks, \( \ell \) keys are produced during the encryption, and \( E \) is re-keyed \( \ell \) times. But for each of these keys, only 2 different leakage traces of \( E \) are produced (i.e., with inputs 0 and 1), and this typically makes side-channel attacks (mainly DPAs) hard to mount (if not impossible). Re-keying is usually

\(^4\) EDT was proved CIML2 in [PSV15], but its CCAmL2 security was not formally analyzed.
deemed as costly. However, as mentioned in the introduction, this cost is expected to be largely compensated by the more limited use of side-channel countermeasures (i.e., when targeting high side-channel security – see Section 7).5

Minimal Message Manipulation for Side-Channel Confidentiality. Encryption schemes have to perform operations on sensitive messages. Concretely, the more operations manipulate messages, the more information leaks about them, and this leakage is very hard to completely avoid [PSV15]. To remedy this situation, in EDT each message block is involved in only a single and easy to protect XOR operation. This is arguably the minimum that cannot be avoided in the current state-of-the-art.

EtM & Hash-then-MAC for Side-Channel Integrity. As efficiency remains the primary concern for crypto designs it may seem surprising, at a first sight, not to simply adopt a good black-box design and add leakage protections to it. Concretely, for AE, integrated designs that perform encryption and tag generation in a “single run” seem the most efficient and attractive: examples include IAPM [Jut08], OCB [RBBK01], TAE [LRW11], OTR [Min14], COFB [CIMN17], and sponge-based proposals [BDPV12, BDH+15, CDNY18, BBLT18]. However, integrated designs typically employ the Decrypt-then-Verify style decryption, which (as observed by Barwell et al. [BMOS17]) would leak unverified plaintext and alter the side-channel security. In contrast, as independently observed in [DEM+17, BPPS17, BMOS17], the plain Encrypt-then-MAC (EtM) paradigm grants resistance to decryption leakage: since its decryption is Verify-then-Decrypt, invalid decryption queries are prevented to step into the decryption process and thus cannot produce much leakage (i.e., it only needs to secure the tag verification). As a result, EtM seems the most suitable classical solution to mitigate concrete side-channel leakages.

For MAC designs, a similar situation appears: during the input-absorbing phase, keyless crypto hash functions are preferred to the typically more efficient universal hash functions. The motivation is simply to minimize the number of calls to keyed primitives. Used in the plain Hash-then-MAC paradigm $T = \text{TGF}_K(H(U))$ ($U = IV || C$ in EDT as well as in all our examples: this maximizes the resistance to invalid decryption queries), side-channel protections are only needed for the Tag Generation Function $\text{TGF}_K$, which is much easier. In fact, leakage security of Hash-then-MAC has been extensively analyzed [MOSW15, BKP+18, DEM+17]. Interestingly, if we replace the $IV$ by a nonce, then such designs naturally achieve full nonce-robustness on the decryption side, since resisting invalid (i.e., inherently nonce-misuse) decryption means resisting misuse.

The elegant and provably secure idea used in EDT is to use an invertible block cipher for TGF, and to define the integrity checking as “For input $(U, Z)$, if $H(U) = \text{TGF}_K^U(Z)$ then accept Else reject”. In this vein, decryption only leaks a useless pseudorandom value $\text{TGF}_K^U(Z)$ rather than the right tag of $U$, excluding the obvious forgery.

Shortages of EDT. EDT did not consider handling Associated Data (AD), rendering it insufficient for practical use. Note that existing approaches for enhancing an authenticated encryption scheme with efficient AD handling [Rog02] was only proved secure in the black-box setting, and thus may not suffice in the leakage setting.

Another drawback of EDT is the concrete security, which contains the birthday terms $\sigma^n/2^n + t\sigma/2^n + t^2/2^n$ even in the black-box setting, where $t$ denotes the time complexity of the adversary, $\sigma$ denotes the number of processed message blocks, and $n$ is the block-size of the block cipher in use. While the term $\sigma^n/2^n$, concerning with the security of the encryption part, could be overcome by using counters (like the standard approach in the counter mode, or in GCM) instead of two fixed constants, the terms $t\sigma/2^n$ and $t^2/2^n$, resulting from the hash-then-SPRP authentication, are more problematic. Indeed the

5 This re-keying-per-block stream cipher was first analyzed in [AB00, BY03] for black-box security, but its leakage security was only later proved in [YSPY10, SPY13]. These were actually the basis of EDT.
term $t^2/2^n$ is matched by a simple attack.\footnote{The term $t^2/2^n$ is more severe as it indicates low-data attacks with $2^{n/2}$ computations. But this is not matched by any attack, and might be an artifact of the standard model proof approach.} In detail, the attacker first pre-computes $t$ hash digests $H(IV, c_1) = d_1, \ldots, H(IV, c_t) = d_t$ for arbitrary $IV$’s and $c_1, \ldots, c_t$, and then encrypts $\sigma$ messages $m^*_1, \ldots, m^*_\sigma$ to obtain $(IV_1, c_1, Z_1), \ldots, (IV_\sigma, c_\sigma, Z_\sigma)$. Now, as long as a hash collision $H(IV_i, c_i) = d_j$ is found—the probability is $t^2/2^n$—the tuple $(IV, c_j, Z_i)$ constitutes a valid forgery. As the standard NIST recommendation is to stop using a key when the probability of some leakage exceeds $2^{-32}$, this means (assuming that the adversary could afford $t$ computations) that the key of EDT has to be updated after processing $\approx 2^{36}/t$ message blocks. Nowadays it is likely to have $t \geq 2^{64}$ [SBK+17], meaning that the number of message blocks that can be securely processed by EDT is less than $2^{32}$, which is less than 64GB when $n = 128$. In all, using a (tweakable) block cipher with $n$-bit blocks, we would like to achieve beyond $n/2$-bit security which is usually called “beyond-birthday-bound” (see, e.g., [PS15]).

In addition, unlike modes such as GCM-SIV [BHT18], the multi-user security of EDT cannot be boosted by simply increasing the key length of the block cipher. This constitutes another limitation for its practical use. In fact, it is not even clear how to use a $(2n, n)$-block cipher in a re-keying encryption mode while keeping the rate $1/2$.

3.2 Our New Ideas

First, it is quite obvious that the random $IV$ in EDT can be replaced by a nonce, and this increases its robustness against randomness misuse (see [Rog04c]). Then, besides replacing the two constants in EDT by GCM-style counters to achieve BBB encryption (as mentioned), EDT uses four main new ideas that we describe next.

**Efficient Handling of AD.** Our first observation is that, to securely handle the AD, we can simply insert the AD as a part of the input to the Hash-then-MAC authenticator.

**Hash-then-TBC for Efficient BBB Authentication.** After many failed trials, it appears that the simplest and most efficient approach to BBB secure leakage-resistant authentication is to increase the output of the hash to $2n$ bits, and then use an $(n, n, n)$-TBC for the tag generation function $\mathcal{TGF}$ to absorb this digest.

Concretely, upon tagging $U$, we apply a $2n$-bit hash function $H$, i.e., $V||W = H(U)$, where $V$ and $W$ are the two $n$-bit halves. Then, the tag is $Z = \mathcal{TGF}_K(W, V) = (E^W_K)(V)$. To resist verification/decryption leakage, we utilize the inverse, i.e., upon $(U, Z, W)$, “If $(E^W_K)^{-1}(Z) = V$ for $V||W = H(U)$ then accept Else reject”.

If one insists on using classical $(n, n)$-block ciphers, then it seems two calls are necessary for the BBB (leakage-resistant) authentication. Both of them as well as the additional internal wires have to be well protected, which could be problematic. We believe a single protected TBC-call would be more efficient/secured than two protected classical block cipher calls. Moreover, the $(n, n, n)$-TBC also cinches the mu security strengthening trick.

**Making the Hash Function Concrete.** To make the mode more concrete (and practical), we instantiate the hash function $H$ from the same TBC $E$, as in Figure 3. For this purpose, we (have to) view $E$ as an ideal TBC IC. An ideal $(n, n, n)$-TBC is essentially the same as an ideal $(2n, n)$-block cipher—both are $2^{2n}$ independent random permutations. With the consideration that Hirose’s double-block-length (DBL) construction is XOR-only and does not require re-keying within each invocation, we select it to instantiate a $3n$-to-$2n$-bit compression function $\text{Hir}[E]$ [Hir06], and then use $\text{Hir}[E]$ in the strengthened Merkle-Damgård to build $H[E]$. The formal description is given later in Fig 5. In this vein, the obtained mode is purely TBC-based which may reduce implementation costs. Yet, the DBL hash does not trivially offer security: see the end of this section.

**Public Randomness to Remedy mu Security Degradation.** Roughly, mu security
degradation stems from (i) collision between user keys, and (ii) multiple user keys that increase the effectiveness of offline computations.

We illustrate both with examples. Consider $u$ users. For issue (i), the probability to have two users $i$ and $j$ such that $K_i = K_j$ is $\frac{u^2}{n}$. With this, for any $(N, A, M)$, \textsc{Enc}_{K_i}(N, A, M) \rightarrow C$ is a valid forgery for user $j$. For issue (ii), the adversary could compute \textsc{Enc}_{K_i}(N, A, M) \rightarrow C$ for a guess $K^*$, and as long as $K^*$ equals $K_i$ for some user $i$ the adversary could have a chance to detect and notice $K_i = K^*$: the probability of such a collision is $\frac{1}{n}$ in the $su$ setting, yet balloons to $\frac{n}{m}$ in the $mu$ setting.

Clearly, increasing the secret key length solves both (just boosting the denominator). We show that they can be overcome by properly using public key bits. Concretely, after we replace all \textsc{E}-calls in the EDT encryption by TBC-calls, we could simply use the “public key” $PK$ for the tweak input. This simple trick does not work for the authentication since the tweak input of TGF has been “occupied” by a half $V$ of the hash digest. Yet, once we append $PK$ to the hash input $U$, we achieve some separation between users.

Roughly speaking, now two encryption instances collide only if a collision occurs between both their secret and public keys. Thus, the probability of user key collision is decreased and issue (i) is solved. For issue (ii), while $PK$ cannot immediately enlarge the denominator, it makes each guess $K^* = SK^*\|PK^*$ less effective: the guess $K^*$ hits a key $K_i = SK_i\|PK_i$ only if $PK^* = PK_i$. Therefore, if the maximal multiplicity of the $PK$ value is small $\mu \ll u$ (which can be achieved by ensuring distinct $PK$ values, or picking $PK$ at random), for issue (ii) the additional public randomness reduces the probability to $\frac{\mu}{n} \ll \frac{n}{m}$.\(^7\) While more random bits are required, the secret key remains of $n$ bits. In this respect, we note that it is easier to generate “public keys” than secret ones: for the latter a key agreement protocol is needed, while for the former one could uniformly pick and send it to the other user in (authenticated) plaintext form.

Summary. Our technical contribution is focused on solving the two following problems:

1. Proving that public randomness does avoid the aforementioned multi-user degradation (both for encryption and authentication).
2. Proving that the hash-then-TBC authentication does achieve BBB $mu$ security against verification leakage when the hash function is a strengthened Merkle-Damgård iteration of Hirose DBL compression function. We remark this is highly non-trivial, since we aim at $2^n/n^2$ security (so far beyond birthday).\(^8\)

A simpler mode AEDT. If we only apply the efficient AD handling technique to EDT, then we’ll obtain an AEAD mode for block ciphers that achieve the same birthday-bound

---

\(^7\) This clarification is simplified. We have to rely on some non-standard collision properties of $H$.

\(^8\) If $H$ was (indifferentiable from) a random oracle with good $2^n/n^2$ bounds, then the result would be much easier to obtain. Yet, neither “plain” Merkle-Damgård [CDMP05] nor Hir[I] [Men13] is indifferentiable.
security as EDT. We may name it AEDT, standing for EDT with AD handling. In this paper, we concentrate on the more secure mode TEDT.

4 Specification for TEDT

Parameters. Built upon an \((n, n, n)\)-TBC, the key of TEDT is written as \(K \parallel PK\), with \(|K| = n\) and \(|PK| = n - 1\). While \(K\) is secret, \(PK\) can be public.

The secret key \(K\) is picked uniformly (from now on we eschew the notation \(SK\) used in section 3.2). On the other hand, \(PK\) could be either uniformly picked or ensured distinct for each user. See section 3.2: this requirement avoids security degradation. TEDT accepts \(3n\) bytes nonce and results in \(n\)-bit stretch. From a nonce \(N\), it generates two sequences of distinct constants for encryption / decryption, i.e., \(P_i(N) = N || [i]_{2^n-1} || 0\), and \(Q_i(N) = N || [i]_{2^n-1} || 1\), where the integer \(i \in [0, \ldots, 2^p-1]\). The index \(i\) corresponds to the nonce block index, and the first sequence \(P_i(N)\) is used for re-keying, while \(Q_i(N)\) is used to generate key stream blocks: see Figure 4. By this, a single message cannot exceed \(n \cdot 2^p-1\) bits. In Table 1, we list the TEDT parameters for the general \(n\)-bit case, and for the primary use case \(n = 128\).

<table>
<thead>
<tr>
<th>\textbf{}</th>
<th>\textbf{General (n)}</th>
<th>\textbf{n = 128}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key size</td>
<td>(2n - 1) bits</td>
<td>255 bits</td>
</tr>
<tr>
<td>Nonce size</td>
<td>(n) secret, (n - 1) public</td>
<td>128 private, 127 public</td>
</tr>
<tr>
<td>Maximal message</td>
<td>(3n/4 - 1) blocks</td>
<td>96 bits</td>
</tr>
<tr>
<td>Maximal AD</td>
<td>(2n/2 - 1) bits</td>
<td>2^{15} bytes</td>
</tr>
<tr>
<td>Stretch</td>
<td>(n) bits</td>
<td>16 bytes</td>
</tr>
</tbody>
</table>

Hash and Padding. Our TBC-based hash function \(H[E]\) is a Merkle-Damgård iteration of the Hirose DBL compression function \(Hir[E]\). The domain separation constant in \(Hir[E]\) is \(\theta = 1\), and the initial vector for the Merkle-Damgård is all-zero \([0]_{2n}\). The fact that \(P_1(N) = Q_1(N) \oplus \theta\) will help reducing some constant factors in the security proof. Formally, \(H[E]\) is described in Figure 5. For our purpose, the hash input is a 4-tuple \((A, N, c, T)\), with \(A\) first to handle static and some type of incremental associated data. A suffix-free

---

9 Any function ensures \(P_i(N) \neq P_i(N')\), \(Q_i(N) \neq Q_i(N')\) for any \((i, N) \neq (i', N')\), and \(P_i(N) = Q_i(N)\) for any \((i, N), (i', N')\), would suffice. But our choice, motivated by GCM, is an obvious simple approach.
padding $\text{pad}(A, N, c, T)$ is needed for the Merkle-Damgård iteration. This padding must prevent the attacks trying to confuse the $A$ and $c$ fields of variable length. For this, we define $\text{pad}(A, N, c, T) := A||N||c||T||0^n||\[A\]_{n/2}||\[c\]_{n/2}$. This padding upper-bounds the length of $A$ by $2^{n/2} - 1$, as shown in Table 1. We remark that this padding is part of the AEAD scheme and not of the hash function $H$.

The Encryption is an EtM composition of the counter-based variant of Pereira et al.'s re-keying encryption [PSV15] and the Hash-then-TBC tag generation, as discussed in section 3.2. Both KDF (in the encryption) and TGF (in the authentication) are instantiated with a single TBC-call. To achieve a separation (so that we can use the same secret key $K$ for both), we reserve 1 bit in the tweak input: concretely, we derive $T = PK||0$ by padding 0 and use it for the tweak of KDF$_K$, and chop the output of $H$ by 1 bit to obtain an $n - 1$ bit half $W$ and use $W||1$ for the tweak of TGF$_K$. The whole process is described by the algorithm TEDT|E,Enc$_K,PK(N, A, M)$ in Figure 5. We separate KDF and TGF from the other TBC-calls (though algorithmically equal) for conceptual convenience: both highlighting the heavily protected calls and simplifying language. Since $T = PK||0$, in the remainder of this paper, we also call $T$ a “public-key”.

The Decryption is of Verify-then-Decrypt type: it first invokes the Hash-then-TBC verification to check the integrity, and decrypts only if the input is decided as authentic. To ensure this verification does not leak the right tag, the inverse of TGF is invoked. The whole process is described by the algorithm TEDT|E,Dec$_K,PK(N, A, C)$ in Figure 5.\(^{10}\)

Remark. The public-key $T$ in the KDF call is also crucial for avoiding the mu security degradation term $2^n/u$, so it cannot be replaced by a constant. Furthermore, unlike most TBC modes [PS15, IMPS17], we do not use domain separation to make TBC-calls during encryption and authentication independent. This avoids many issues (such as unusual message block size) discussed in [IMPS17, section 6.1].

We are not aware of any comprehensive analysis of how symmetric cryptosystems shall manage “public keys”. In particular, it is not clear if $PK$ shall be updated with the secret part $K$ during key updating, which we recommend in order to be conservative.

## 5 Leakage Integrity of TEDT

We now investigate the muCIML2 security of TEDT in the “unbounded leakage” setting [BKP+18, BPPS17], which means all the intermediate values completely leak, except the master key $K$ of KDF$_K$ and TGF$_K$ (which remains secret). Formally, we define the leakage function $L^* = (L^*_{\text{enc}}, L^*_{\text{dec}})$, where:

- **$L^*_{\text{enc}}$** consists of the following information appearing during the encryption:
  - $\{K, T, X, Y\}$ for each leaking TBC-call to $E^T_K(X) \to Y$ or $(E^T_K)^{-1}(Y) \to X$,
  - $\{T, I, S\}$ for each leak-free KDF-call to $\text{KDF}(K, T, I) \to S$,
  - $\{W, V, Z\}$ for each leak-free TGF-call to $\text{TGF}(K, W, V) \to Z$, or inverse leak-free TGF-call $\text{TGF}^{-1}(K, W, Z) \to V$,
  - $\{a, b\}$ for each XOR action $a \oplus b$.

- **$L^*_{\text{dec}}$** consists of the above that are generated during the decryption.

Note that the above means KDF- and TGF-calls do not leak any information about $K$; the same assumption will be used in Section 6. Of course it is too strong to be fulfilled in reality. Yet, it nicely abstracts the details in the TBC and allows to focus on the mode-level leakages. So we use it as a first step and leave relaxations for further research.

\(^{10}\)There are two decryption times for the ciphertexts of any given length, corresponding to invalid & valid ciphertexts (the difference of which is not critical since available through black box queries).
Then our main claim is as follows:

\[
\text{Adv}^\text{muCIML2}_{\text{TEDT},L^*}(u, \sigma) \overset{\text{def}}{=} \max \left\{ \text{Adv}^\text{muCIML2}_{A,\text{TEDT},L^*}(u) \right\},
\]

where the maximum is taken over all q-bounded adversaries against u users that have at most ∑ blocks in all their queried plaintexts and ciphertexts including associated data. Then our main claim is as follows:
Theorem 1. Assume that the $u$ public-keys $T_1, \ldots, T_u$ are uniformly distributed, $n \geq 6$, $2\sigma + 3(q_e + q_d + q_{IC}) \leq 2^n/8$, and leakage $L^*$ is “unbounded” as above. Then

$$\text{Adv}_{\text{muCIML2}}^T(u, q, \sigma) \leq \frac{(2n^2 + 17)(4\sigma + 6(q_e + q_d) + 2q_{IC})}{2^n} + \frac{u^2}{2^{n/2}} + \frac{1}{2n!} \cdot \left(\frac{4u}{2^n}\right)^n$$

(1)

in the ideal TBC model.

The bound means that, as long as we carefully protect the key of $KDF_K$ and $TGF_K$, the muCIML2 bound is asymptotically optimal $O\left(\frac{n^2 u^2 + n^2 \sigma + n^2 q_{IC}}{2^n}\right)$. Concretely, when $n = 128$, the integrity security is up to $u \approx 2^{126}$ users, $\sigma \approx 2^{114}$ blocks, and $q_{IC} \approx 2^{114}$ offline computation. If the $u$ public-keys are ensured to be distinct, then we could drop the terms $\frac{u^2}{2^n}$ and $\frac{1}{2n!} \cdot (4u/2^n)^n$, but the bound does not substantially improve. In the black-box model, similar asymptotically optimal bounds can be proved: see Appendix D.\footnote{In an ongoing lightweight cryptography standardization effort, the NIST requires new proposals to securely process more than $2^{50}$ bytes of data (before key updating) and resist $2^{112}$ computations in the single-user setting [NIS19]. What TEDT achieves is clearly far better.}

The remaining of this section is filled by the proof of Theorem 1. We first serve an outline of the proof and some hints on how we obtain the involved terms in subsection 5.1. The two core steps of the proof will be in subsection 5.3 and 5.4, but before them we outline of the proof and hints on the terms

5.1 Outline of the Proof and Hints on the Terms

We proceed in two main steps. First, below in section 5.3, we idealize the scheme TEDT[I|K,T] via replacing calls to $KDF_{K_i}, TGF_{K_i}, \ldots, KDF_{K_u}, TGF_{K_u}$ by several tweakable random permutations $\pi_1, \ldots, \pi_u$, such that $\pi_i$ and $\pi_j$ are independent if and only if $K_i \neq K_j$. These permutations are secret, i.e., they cannot be queried by $A$.

We then show the real and idealized schemes are indistinguishable. The actual goal of this step is to argue that during the interaction, the secret keys $K$ cannot be recovered from information gained by the adversary. The indistinguishability is proved using the H-coefficient technique [CS14]. For this step, note that all the internal values of the TBC-calls, except $K$, are given to the adversary $A$. This additional information increases the offline computation power of $A$. Therefore, conceptually, the situation is that the adversary $A$ makes $O(\sigma + q_{IC})$ queries to $I$ and tries to recover the secret user keys.

Typically, the naive proof idea would be: since each offline $I$-query will “hit” one of the $u$ user secret keys with probability $u/2^n$, a union bound over all the offline queries yields $O\left(\frac{u(\sigma + q_{IC})}{2^n}\right)$. But our goal is to avoid the factor $u$, i.e., the multi-user security degradation. To have a better bound, we rely on the fact that these $O(\sigma + q_{IC})$ offline queries are “separated” by the tweak inputs. In detail,

- The KDF-calls make $I$-calls of the form $IC_K^{P_{K_i}[0]}(I)$, and this is related to the distribution of the public keys. When the public keys are uniformly picked, the number of users sharing a same public key value cannot exceed $u$, except with a sufficiently small probability of $O(u/2^n)$. The TBC ensures independence between calls with different tweaks, i.e., for a user with public key $PK_i \neq PK_j$ and secret key $K_j$, an offline $IC$ query of the form $IC_k^{P_{K_j}[0]}(x)$ is independent from its KDF-calls $IC_k^{P_{K_i}[0]}(I)$, even if $k = K_j$. By this, an offline $IC$ query of the form $IC_k^{P_{K_i}[0]}(x)$ can
have its key $k$ “hit” at most $n$ users (i.e., the users using $pk$ as the public key). This
decreases the term to $O\left(\frac{n(n+q)}{2^n}\right)$ and avoids the factor $u$.\footnote{When the public keys are ensured distinct, then following the same idea, the factor $n$ is decreased to 1.}

- The TGF-calls make $IC$-calls of the form $IC_K^{W\parallel 1}(V)$ or $(IC_K^{W\parallel 1})^{-1}(Z)$, in which the
tweak $W$ is always (half of) the output of the hash function. Given distinct inputs,
the number of semi-collisions among the outputs of $H$ cannot exceed $n$, except with
a sufficiently small probability of $O\left(\frac{n(n+q)}{2^n}\right)$. This means for each $W$, TEDT makes at
most $n$ calls of the form $IC_K^{W\parallel 1}(V)$ or $(IC_K^{W\parallel 1})^{-1}(Z)$ for each user. But note that, due
to the collision among the public keys, TEDT may make the same hash evaluation
for different users: as argued, this situation appears at most $n$ times. The two effects
indicate that each offline $IC$ query of the form $IC_K^{W\parallel 1}(x)$ simultaneously “targets” at
most $n^2$ TGF calls, and this creates the term $O\left(\frac{n^2(n+q)}{2^n}\right)$. Although the factor $n^2$
is worse, the factor $n$ is still avoided.

Next, in subsection 5.4, we prove unforgeability for the idealized scheme to complete the
muCiML2 proof. To this end, we show that various query records gained by the adversary
during the interaction will not form certain “chains”, except with a sufficiently small
probability. As will be seen in the detailed analysis, the adversary has two ways to create
such “chains”: either trying to create a collision between two different hash evaluations,
or trying to create a collision between a hash digest and an inverse TGF-call. Since we
are using a DBL hash, the probability of hash collision is $O\left(\frac{n^2(n+q)}{2^n}\right)$; since it is hard
to create $n$-collisions on the $W$ values, it can be shown that the probability of collisions
between a hash digest and an inverse TGF-call is $O\left(\frac{nq}{2^n}\right)$.

As an additional remark and hint, note that if we compute the tag from the digest
$V\parallel W = H[E](\text{pad}(A,N,c))$ without using the public key $T$, then a user secret key collision
$K_i = K_j$ enables forgery since $TEDT(K_i,W,V) = TEDT(K_j,W,V)$. But since the public keys
are injected, for $V\parallel W = H[E](\text{pad}(A,N,c,T_i))$ and $V'\parallel W' = H[E](\text{pad}(A,N,c),T_j)$ we
have $TEDT(K_i,W,V) = TEDT(K_j,W',V')$ only if both $K_i = K_j$ and $T_i = T_j$, the probability
of which is reduced to the small enough term $u^2/2^{2n}$.

5.2 Properties of Hirose Compression Function and pad

Denote by $Hir[IC]$ the Hirose compression function built upon the ideal TBC $IC$. Note that
any adversary $A$ against $Hir[IC]$ can be normalized to an adversary $A'$ that only makes
pairs of Hirose matching queries: $A'$ runs $A$, and

- each time $A$ makes a forward query $IC_K(T,X)$, $A'$ makes a query $IC_K(T,X \oplus \theta) \rightarrow Y'$
  right after relaying $IC_K(T,X) \rightarrow Y$, and

- each time $A$ makes a backward query $IC_K^{-1}(T,Y)$, $A'$ makes a query $IC_K(T,X \oplus \theta) \rightarrow Y'$
  right after relaying $IC_K^{-1}(T,Y) \rightarrow X$.

Therefore, we could concentrate on adversaries that only make such pairs of matching
queries. In this vein, the function $Hir[IC]$ has the following collision-related properties.

Lemma 1. For any $A$ making $q$ pairs of matching queries to $IC$ with $1 \leq q \leq 2^n/4$, it holds

$$\Pr\left[ (X,X') \leftarrow A^{IC} : \text{chop}(Hir[IC](X)) = \text{chop}(Hir[IC](X')) \right] \leq \frac{6q}{2^n}.$$
Proof. The collision resistance of $\text{Hir}[\mathcal{I}]$ does not necessarily imply the collision resistance of $\text{chop}\circ\text{Hir}[\mathcal{I}]$. To show the latter, consider any two pairs of matching queries $((v, h, g, y), (v \oplus \theta, h, g \oplus \theta, z))$ and $((v', h', g', y'), (v' \oplus \theta, h', g' \oplus \theta, z'))$. We now bound the probability of collision due to these queries. We distinguish two cases:

Case 1: $(v, h, g, y) = (v' \oplus \theta, h', g' \oplus \theta, z')$. This means the two pairs are actually the same two IC queries in different orders. Then the collision indicates $\text{chop}(g \oplus y) = \text{chop}(g \oplus \theta \oplus z)$. Without loss of generality, assume $(v, h, g, y)$ is made after $(v \oplus \theta, h, g \oplus \theta, z)$. Then regardless of whether the query is forward or backward, either $g$ or $y$ is uniform in a set of size at least $2^n - 2q$. Therefore,

$$\Pr[\text{chop}(g \oplus y) = \text{chop}(g \oplus \theta \oplus z) | g \oplus y = b \neq 0 \text{ or } 1] \leq \frac{2}{2^n - 2q},$$

and with $q$ matching pairs, we have

$$\Pr[\text{Case 1}] \leq \frac{2q}{2^n - 2q}.$$

Case 2: $(v, h, g, y) \neq (v' \oplus \theta, h', g' \oplus \theta, z')$. Then it can be seen the involved queries are four different ones, and a collision indicates

$$g \oplus y = g' \oplus y' \land \text{chop}(g \oplus \theta \oplus z) = \text{chop}(g' \oplus \theta \oplus z')$$

or

$$g \oplus y = g' \oplus \theta \oplus z' \land \text{chop}(g \oplus \theta \oplus z) = \text{chop}(g' \oplus y').$$

Similarly to Case 1, in any subcase, we have

$$\Pr[g \oplus y = g' \oplus y'] \leq \frac{1}{2^n - 2q} \text{ and } \Pr[\text{chop}(g \oplus \theta \oplus z) = \text{chop}(g' \oplus \theta \oplus z')] \leq \frac{2}{2^n - 2q},$$

so that $\Pr[\text{Eq. (2)}] \leq \frac{2}{(2^n - 2q)^2}$. A similar reasoning shows $\Pr[\text{Eq. (3)}] \leq \frac{2q}{(2^n - 2q)^2}$. Therefore, when $q \leq 2^n/4$ we have

$$\Pr[\text{Case 2}] \leq \frac{q}{2^n} \cdot \frac{4}{(2^n - 2q)^2} \leq \left(\frac{4q}{2^n}\right)^2 \leq \frac{4q}{2^n}.$$

In summary,

$$\Pr[\text{collision after chopping}] = \Pr[\text{Case 1}] + \Pr[\text{Case 2}] \leq \frac{6q}{2^n},$$

as claimed.

We also need the multi-collision resistance of the chopped Davies-Meyer.

Lemma 2. Consider the Davies-Meyer function $\text{DM}[\mathcal{I}](K || T, h) = \text{IC}^T(K) \oplus h$ built upon the ideal $TBC$ $\mathcal{I}$. Then, for any adversary $A$ making at most $q \leq 2^n/2$ queries to $\mathcal{I}$ and any integer $\lambda$, it holds

$$\Pr[(X_1, \ldots, X_\lambda) \leftarrow A^{\mathcal{I}} : \text{chop}(\text{DM}[\mathcal{I}](X_1)) = \ldots = \text{chop}(\text{DM}[\mathcal{I}](X_\lambda))] \leq \frac{(4q)^\lambda}{\lambda ! 2^{(\lambda - 1)n}}.$$

Proof. Consider any $\lambda$ IC queries $(K_1 || T_1, h_1, y_1), \ldots, (K_\lambda || T_\lambda, h_\lambda, y_\lambda)$ listed according the order they were made, and let $z_i = h_i \oplus y_i$ for each $i$. Then, since $q \leq 2^n/2$, for any $i \geq 2$ we have $\Pr[\text{chop}(z_i) = \text{chop}(z_1)] \leq \frac{2^2}{2^n - q} \leq \frac{1}{2^n}$. Therefore,

$$\Pr[\lambda \text{ collisions}] \leq \prod_{i=2}^\lambda \Pr[\text{chop}(z_i) = \text{chop}(z_1)] \leq \left(\frac{q}{2^n}\right)^{\lambda - 1} \left(\frac{4q}{2^n}\right)^\lambda \leq \frac{(4q)^\lambda}{\lambda ! 2^{(\lambda - 1)n}},$$

as claimed.
To formally define the idealized scheme, we define an ideal primitive TEDT.

Note that collision-free, resulting in the undesired birthday term and recall the technique.

First define the transcripts of the distinguisher when interacting with the real/ideal scheme, notation D oracles of the idealized scheme.

Denote by TRPFamily and Lindell [GL17]. Based on this, the ideal scheme is obtained by replacing every call to the permutations keys are actually not used by TRPFamily: they only define a “pattern” for the permutations π1, ..., πu. The definition and use of TRPFamily are similar to Gueron and Lindell [GL17]. Based on this, the ideal scheme is obtained by replacing every call to KDF(Ki, t, X)/TGF(Ki, t, X) and KDF−1(Ki, t, Y)/TGF−1(Ki, t, Y) by TRPFamily(i, t, X) and TRPFamily−1(i, t, Y) for i = 1, ..., u. Denote the obtained idealized scheme by TEDT[Ic, TRPFamily][T].

5.3 Idealizing TEDT

To formally define the idealized scheme, we define an ideal primitive TRPFamily in Fig. 6. Briefly, TRPFamily captures the behavior of an ideal TBC instantiated with u uniformly picked keys (K1, ..., K_u), i.e., u tweakable random permutations π1, ..., π_u. Yet, the u keys (K1, ..., K_u) are actually not used by TRPFamily: they only define a “pattern” for the permutations π1, ..., π_u. The definition and use of TRPFamily are similar to Gueron and Lindell [GL17]. Based on this, the ideal scheme is obtained by replacing every call to KDF(Ki, t, X)/TGF(Ki, t, X) and KDF−1(Ki, t, Y)/TGF−1(Ki, t, Y) by TRPFamily(i, t, X) and TRPFamily−1(i, t, Y) for i = 1, ..., u. Denote the obtained idealized scheme by TEDT[Ic, TRPFamily][T].

5.3.1 Preparation for the Indistinguishability Proof: Notations

Denote by LEncK,T and LDecK,T the leaking encryption and decryption oracles of the real scheme TEDT[Ic,K,T], and by ILEncT and ILDecT the leaking encryption and decryption oracles of the idealized scheme TEDT[Ic, TRPFamily][T] (the keys have been made explicit). Formally, we are to derive an upper bound for

\[ \Pr[\text{A} \xrightarrow[Ic]{\text{ILDecT}} \text{Ic}] - \Pr[\text{A} \xrightarrow[Ic]{\text{ILEncT}}} \text{Ic}] \]

for any q-bounded D—since this is essentially a distinguishing advantage, we use the notation D instead of A. For this, we employ Patarin’s H-coefficient technique [CS14]. We first define the transcripts of the distinguisher when interacting with the real/ideal scheme, and recall the technique.

 Initialization(u):
  1. Choose \( u \) random keys \( K^* = (K^*_1, ..., K^*_u) \) \( \xleftarrow{} \) \( \mathcal{K} \), and tweakable random permutations \( \bar{\pi}_1, ..., \bar{\pi}_u \) from \( \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}^n \), under the constraint that \( \bar{\pi}_i = \bar{\pi}_j \) if and only if \( K^*_i = K^*_j \).

 Interface TRPFamily(i, T, X)
  1. Return \( \bar{\pi}_i (T, X) \)

 Interface TRPFamily−1(i, T, Y)
  1. Return \( \bar{\pi}_i^{-1} (T, Y) \)

 Finalization(u):
  1. Choose \( u \) random keys \( K = (K_1, ..., K_u) \) \( \xleftarrow{} \) \( \mathcal{K} \) under the constraint that \( K_i = K_j \) if and only if \( K^*_i = K^*_j \).

 Figure 6: Definition of the primitive TRPFamily.

Last, the pad scheme is effective in distinguishing between inputs with different semantics.

Lemma 3. It holds pad(A, N, c, T) \( \neq \) pad(A′, N′, c′, T′) for any two distinct tuples (A, N, c, T) and (A′, N′, c′, T′).

Proof. Note that pad(A, N, c, T) \( \neq \) pad(A′, N′, c′, T′) unless \( |A| = |A'| \) & \( |c| = |c'| \), as otherwise the values appended in the length field are different. But if \( |A| = |A'| \) and \( |c| = |c'| \) (note \( |N| = |N'| \) and \( |T| = |T'| \) always hold), then \( (A, N, c, T) \neq (A', N', c', T') \) indicates at least one of them has different values, and we thus have \( A||N||c||T \neq A'||N'||c'||T' \) and the resulting values are different after padded. \( \square \)
In detail, we summarize the adversarial queries to the ideal TBC $\IC$ in a set $\tau_{\IC} = \{(k_1, t_1, x_1, y_1), \ldots, (k_{q_{\text{IC}}}, t_{q_{\text{IC}}}, x_{q_{\text{IC}}}, y_{q_{\text{IC}}})\}$, which indicates the $i$-th query is either forward IC$^{\text{IC}}_{t_i}(x_i) \rightarrow y_i$ or backward IC$^{\text{IC}}_{t_i}(y_i) \rightarrow x_i$.

Note that by our assumption, all the IC queries (except for KDF and TGF) made by TEDT are leaked to $\mathcal{D}$. Such information also gives rise to records of the form $\langle k_i, t_i, x_i, y_i \rangle$. To make a distinction, we denote by $\tau^*_{\IC}$ the union of these records and the adversarial query transcript $\tau_{\IC}$. For the number of internal IC calls made by the scheme, we have:

- **upon encrypting $(N, A, M)$,** TEDT makes at most $2 \cdot 2 \cdot \lceil \frac{|M|}{n} \rceil + 2 \cdot \lceil \frac{|A|}{n} \rceil \leq 4(\lceil \frac{|M|}{n} \rceil + \lceil \frac{|A|}{n} \rceil) + 6$ IC queries during encrypting and hashing $A$ and $M$, and $2 \times 3$ queries during hashing the additional 3 blocks due to $N, T$, and the padded length field. Therefore, the number of internal IC calls is at most $4(\lceil \frac{|M|}{n} \rceil + \lceil \frac{|A|}{n} \rceil) + 6$;

- **upon decrypting $(N, A, C)$ with $C = c||Z$,** the number of internal IC calls is at most $4(\lceil \frac{|C|}{n} \rceil + \lceil \frac{|A|}{n} \rceil) + 6$.

Therefore, when interacting with the idealized scheme TEDT[$\IC$, TRPFamily]$_{\mathcal{T}}$, the number of internal IC calls is at most $4\sigma + 6(q_e + q_d)$, and thus

$$|\tau^*_{\IC}| \leq 4\sigma + 6(q_e + q_d) + q_{\text{IC}}. \quad (4)$$

We also summarize the calls to KDF and TGF in a list $\tau_{\mathcal{R}} = \{(K_1, t_1, x_1, y_1), (K_2, t_2, x_2, y_2), \ldots\}.$

In this set, the $i$-th tuple $(K_i, t_i, x_i, y_i)$ indicates:

- When interacting with the real AEAD scheme TEDT[$\IC$]$_{\mathcal{T}}$, the $i$-th query is either KDF/TGF$(K_i, t_i, x_i) \rightarrow y_i$ or KDF$^{-1}$/TGF$^{-1}$(K_i, t_i, y_i) \rightarrow x_i$; and,

- When interacting with the idealized scheme TEDT[$\IC$, TRPFamily]$_{\mathcal{T}}$, the $i$-th query is either TRPFamily$(j, t_i, x_i) \rightarrow y_i$ or TRPFamily$^{-1}$(j, t_i, y_i) \rightarrow x_i$ for some user index $j$ such that $K_j = K_i$, where $K_j$ is the key sampled by the procedure Finalization$(u)$ (see Table 6).

Note that since we assume leak-freeness of KDF and TGF-calls, the secret user keys cannot be seen by the distinguisher, and don’t appear in the true adversarial transcripts. One could imagine that we append these keys to the true adversarial KDF/TGF transcripts at the end of the interaction for conceptual simplicity (see below), and this enables reconstruct the above $\tau_{\mathcal{R}}$.

For convenience, we also keep a list $\tau_{\mathcal{H}} = \{(U_1, V_1||W_1), (U_2, V_2||W_2), \ldots\}$ for the appeared inputs and outputs of the hash function $H[|\IC|]$. Note that by the specification of TEDT, none of the $U_i$ can be empty. As we assumed all the underlying IC queries have been leaked and included in $\tau^*_{\IC}$, this list contains redundant information: it can be recovered from $\tau^*_{\IC}$. But it simplifies the language: for example, see the simple form of Eq. (5).

In addition to the above, the “public-keys” $\mathcal{T} = (T_1, \ldots, T_u)$, where $T_i = PK_i||0$, are also included in the transcript. Moreover, to simplify the arguments (in particular, the definition of bad transcripts), we reveal to the distinguisher the user keys $\mathbf{K} = (K_1, \ldots, K_u)$ at the end of the interaction. In detail,

- in the real world, we reveal the keys $\mathbf{K}$ in use, and
in the ideal world, we reveal the keys $K$ sampled by the procedure $\text{Finalization}(u)$ (see Table 6).

This is without loss of generality since $D$ is free to ignore this additional information to compute its output bit. Formally, we append both $T$ and $K$ to the tuple $(\tau^*_H, \tau^*_IC, \tau^*_TRP)$ and obtain what we call the transcript

$$\tau = (\tau^*_H, \tau^*_IC, \tau^*_TRP, T, K).$$

It may seem exotic that transcripts for an AEAD scheme do not include encryption and decryption query transcripts $\tau_e$ and $\tau_d$. This is due to the unbound leakage assumption: since during encryption and decryption, almost all the underlying queries to IC are leaked, the transcripts $\tau^*_IC$ and $\tau^*_TRP$ allow completely recover the encryption/decryption queries and answers. Therefore, there is no need for their presence in this section.

With respect to some fixed distinguisher $D$, a transcript $\tau$ is called attainable if there exist oracles (IC, TRPFamily) such that the interaction of $D$ with the ideal scheme $(\text{TEDT}|IC, \text{TRPFamily}|T, IC)$ yields $\tau$. We denote $T$ the set of attainable transcripts. In all the following, we denote $T_{re}$, resp. $T_{id}$, the probability distribution of the transcript $\tau$ induced by the real world, resp. the ideal world (note that these two probability distributions depend on the distinguisher). By extension, we use the same notation to denote a random variable distributed according to each distribution.

Given a set $\tau^*_IC$ and an ideal TBC IC, we say that IC extends $\tau^*_IC$, denoted $IC \vdash \tau^*_IC$, if $IC_i(x) = y$ for all $(K, t, x, y) \in \tau^*_IC$. Given a set $\tau^*_TRP$,

- in the real world, we say IC extends $\tau^*_TRP$, denoted $IC \vdash \tau^*_TRP$, if $IC_i(x) = y$ for all $(K, t, x, y) \in \tau^*_TRP$;
- in the ideal world, we say TRPFamily extends $\tau^*_TRP$, denoted TRPFamily $\vdash \tau^*_TRP$, if TRPFamily $(i, t, x) = y$ for all $(K, t, x, y) \in \tau^*_TRP$ and all user index $i$ such that $K_i = K$.

It’s easy to see that for any attainable transcript $\tau = (\tau^*_H, \tau^*_IC, \tau^*_TRP, T, K)$, the interaction of $D$ with oracles $(\text{TEDT}|IC, \text{TRPFamily}|T, IC)$ produces $\tau$ if and only if $IC \vdash \tau^*_IC$ and TRPFamily $\vdash \tau^*_TRP$.

With the above, the main lemma of H-coefficient technique is as follows.

**Lemma 4.** Fix a distinguisher $D$. Let $T = T_{\text{good}} \cup T_{\text{bad}}$ be a partition of the set of attainable transcripts $T$. Assume that there exists $\varepsilon_1$ such that for any $\tau \in T_{\text{good}}$, one has

$$\frac{Pr[T_{re} = \tau]}{Pr[T_{re} = \tau]} \geq 1 - \varepsilon_1,$$

and that there exists $\varepsilon_2$ such that $Pr[T_{id} \in T_{\text{bad}}] \leq \varepsilon_2$. Then $\text{Adv}(D) \leq \varepsilon_1 + \varepsilon_2$.

A proof could be found in [CS14].

### 5.3.2 Completing the Transition

Following the above framework, we start by defining bad transcripts. For a transcript $\tau$, we define $\mu_T$ and $\mu_W$, the maximum multiplicity of $T$ and $W$, as

$$\mu_T := \max_{t \in \{0,1\}^n} \left| \{i \in \{1,\ldots,u\} : T_i = t\} \right|,$$

$$\mu_W := \max_{w \in \{0,1\}^{n-1}} \left| \{(U, V||W) \in \tau^*_H : W = w\} \right|.$$

(5)
Then, an attainable transcript \( \tau \) is bad, if either of the following conditions is fulfilled:

- (B-1) \( \mu_T \geq n, \mu_W \geq n \).
- (B-2) there exists a KDF/TGF query \((K, t, x, y) \in \tau_\mathbb{F}\) such that \((K, t, x, *) \in \tau^*_C\) or \((K, t, *, y) \in \tau^*_C\).

The bound on \( \mu_T \) depends on the distribution \( \text{PKDistribution} \). If \( T_1, \ldots, T_n \) are uniformly distributed, then it’s easy to see

\[
\Pr[\mu_T \geq n] \leq \binom{n}{u} \cdot \frac{1}{2(n-1)(n-3)} \leq \frac{n^{n-1}}{n!2^{(n-1)n}} \leq \frac{1}{2^n} \left( \frac{4n}{2^2} \right)^n.
\] (6)

Whereas if \( T_1, \ldots, T_n \) are distinct, then it trivially holds \( \mu_T = 1 \) and \( \Pr[\mu_T \geq n] = 0 \).

Reasoning about \( \mu_W \) requires the multi-semi-collision resistance of \( H \). Formally, we rely on the following lemma, which claims (multi-semi) collision resistance on \( H \).

**Lemma 5.** Consider the interaction between an muCIML2 \( q \)-adversary \( \mathcal{A} \) and the scheme TEDT|\( IC \), TRPFamily\( |T \), and \( \mathcal{A} \) has at most \( \sigma \) blocks in all its queried plaintext, ciphertext, and associated data. Define

\[
q^*_C := 4\sigma + 6(q_c + q_d) + 2q_C.
\] (7)

Then with probability at least \( 1 - \frac{8nq^*_C}{2^n} \), it holds:

1. \( H \) is collision-free: any two distinct \((U, V \| W)\) and \((U^*, V^* \| W^*)\) in \( \tau^*_H \) necessarily have \( V\|W \neq V^*\|W^* \);
2. \( H \) is multi-semi-collision-free: \( \mu_W < n \).

**Proof.** We denote by \( G_1 \) the game that captures the interaction between the muCIML2 adversary \( \mathcal{A} \) and TEDT|\( IC \), TRPFamily\( |T \). Following Hirose [Hir06], we normalize the game: for each IC query either made by \( \mathcal{A} \) or made by the TEDT mode, we assume the system makes its Hirose matching query immediately (see section 5.2). Denote by \( G_2 \) the obtained normalized game. We now count the number of IC queries in \( G_2 \):

- Since \( Q_i(N) = P_i(N) \oplus 1 \) for any \( i \), the \( 2\sigma - 1 \) IC queries made during the encryption pass only result in an increase of at most 1 new IC queries for the first constant \( P_i(N) \) (since \( P_i(N) \) is the input to KDF, it wouldn’t appear in \( \tau^*_C \));
- the \( 2\sigma + 6(q_c + q_d) \) IC queries made by \( H \) would not result in new queries, since their matching queries have been included in these \( 2\sigma + 6(q_c + q_d) \) queries, and
- the \( q_C \) IC queries made by \( \mathcal{A} \) may give rise to \( \leq q_C \) new queries.

Therefore, in \( G_2 \), the number of IC queries does not exceed \( q^*_C = 4\sigma + 6(q_c + q_d) + 2q_C \), as in Eq. (7), and the upper bound on the number of matching IC query pairs is \( q^*_C/2 \).

Then consider \( G_2 \). We define three bad conditions during its execution:

- (C-1) Collision occurs between compression function outputs. Formally, there exists two \( 3n \)-bit values \( u\|g\|h \) and \( u'\|g'\|h' \) such that:
  - \( (u, h, g \oplus g'), (u, h, g \oplus \theta, g \oplus \theta \oplus h'), (u, h, g, g \oplus g''), (u, h, g \oplus \theta, g \oplus \theta \oplus h'') \in \tau^*_C \)
  - for some \( g', h', g'', h'' \), and
  - \( \text{chop}(g\|h') = \text{chop}(g''\|h'') \).
- (C-2) Initial-vector is bit: \( \exists u, g, h : \text{Hir}[\tau^*_C](u\|g\|h) = [0]^{2n} \). Formally, there exists a \( 3n \)-bit value \( u\|g\|h \) such that \( (u, h, g, g), (u, h, g \oplus \theta, g \oplus \theta) \in \tau^*_C \).
• (C-3) \( n \)-collision occurs: there exists \( n \) records \((u_1, h_1, g_1, z_1), \ldots, (u_n, h_n, g_n, z_n) \in \tau_k^e\) such that \(\text{chop}(g_1 \oplus z_1) = \ldots = \text{chop}(g_n \oplus z_n)\).

For (C-1), from Lemma 1 we have (when \(g_k^e/2 \leq 2^n/4\), i.e., \(g_k^e \leq 2^n/2\))

\[
\text{Pr}[(C-1)] \leq \frac{6}{2^n} \cdot \frac{g_k^e}{2} = \frac{3g_k^e}{2^n}.
\]

For (C-2), consider any pair of queries \((k, t, x, y)\) and \((k, t, x \oplus \theta, y')\). Clearly, regardless of the directions of these two queries, it holds \(\text{Pr}[y = x \land y' = x \oplus \theta] \leq \frac{1}{(2^n-g_k^e)^2} \leq \frac{4}{2^n}\) when \(g_k^e \leq 2^n/2\), and thus

\[
\text{Pr}[(C-2)] \leq \frac{4}{2^n} \cdot \frac{g_k^e}{2} = \frac{2g_k^e}{2^2} \leq \frac{g_k^e}{2^n}. \quad (n \geq 2)
\]

For (C-3), an application of Lemma 2 (also requires \(g_k^e \leq 2^n/2\)) yields \(\text{Pr}[(C-3)] \leq \frac{(4g_k^e)^n}{2^n} \cdot \frac{2^n}{2^n} \leq \frac{4g_k^e}{2^n} \cdot \frac{2^n}{2^n} \leq \frac{4g_k^e}{2^n} \cdot \left(4g_k^e \leq 2^n\right)
\]

A union bound yields

\[
\text{Pr}[(C-1) \vee (C-2) \vee (C-3)] \leq \frac{8g_k^e}{2^n}.
\]

Then, conditioned on \(\neg(C-1) \wedge \neg(C-2) \wedge \neg(C-3)\), we show that the two claims hold. We first consider claim (1). For any two records \((U, V)\) and \((U^*, V^* \parallel W^*)\), assume that \(\text{tail}\) is the maximum common suffix of \(U\) and \(U^*\), i.e., \(U = \text{header} || u^* \parallel \text{tail}, U^* = \text{header} || u^* \parallel \text{tail}, |u| = |u^*| = n, u \neq u^*,\) and \(|\text{header}|, |\text{header}^*|,\) and \(|\text{tail}|\) are multiples of \(n\). Then we distinguish two cases:

**Case 1:** either \(\text{header} || u\) or \(\text{header}^* || u^*\) is empty. Without loss of generality, we assume \(\text{header} || u\) is empty. Since \(U\) isn’t empty, this means \(\text{tail}\) isn’t empty. Then we have \(H(\text{header}^* || u^*) \neq [0]_{2^n}\) by \(\neg(C-2)\), i.e., in \(H(U^*)\), the hash-chain value after absorbing \(u^*\) is different from the initial vector \([0]_{2^n}\). So the two “first-block” calls in absorbing \(\text{tail}\) in \(H(U)\) and \(H(U^*)\) are different. By \(\neg(C-1)\), this means the resulted hash-chain values are different. Then by iteratively applying \(\neg(C-1)\), it can be seen the “last-block calls” in \(H(U)\) and \(H(U^*)\) are different, and further \(V || W \neq V^* || W^*\).

**Case 2:** neither \(\text{header} || u\) or \(\text{header}^* || u^*\) is empty. Then by \(\neg(C-1)\), \(H(\text{header} || u) \neq H(\text{header}^* || u^*)\), i.e., the two hash-chain values after absorbing \(u \neq u^*\) are different. If \(\text{tail}\) is empty, then as \(u \neq u^*\) the “last-block calls” in \(H(U)\) and \(H(U^*)\) are clearly different and further \(V || W \neq V^* || W^*\); otherwise, the two claims follow by iteratively applying \(\neg(C-1)\).

We then prove claim (2). The above show that distinct hash inputs \(U\) and \(U^*\) necessarily result in distinct “last-block-calls”. By this, any \(n\) hash inputs \((U_1, \ldots, U_n)\) necessarily result in \(n\) distinct “last-block-calls” denoted \(\text{Hir}(u_1, g_1, h_1), \ldots, \text{Hir}(u_n, g_n, h_n)\). By the definition of \(\text{Hir}\), it can be seen such an \(n\)-semicollision correspond to an \(n\)-collision within \(n\) chopped Davies-Meyer function calls. Concretely, assume that for

\[
g'_1 || h'_1 = \text{Hir}(u_1 || g_1 || h_1), \ldots, g'_n || h'_n = \text{Hir}(u_n || g_n || h_n),
\]

it holds \(\text{chop}(h'_1) = \ldots = \text{chop}(h'_n)\), then it essentially holds

\[
\text{chop}(IC_{u_1}^{h_1}(g_1 \oplus \theta) \oplus (g_1 \oplus \theta)) = \ldots = \text{chop}(IC_{u_n}^{h_n}(g_n \oplus \theta) \oplus (g_n \oplus \theta)),
\]

contradicting \(\neg(C-3)\). These complete the proof. \(\square\)
Lemma 5 indicates
\[
\Pr[\mu_W \geq n] \leq \frac{8q^{1/2}}{2^n}. \tag{8}
\]

We continue analyzing bad transcripts, and consider (B-2). Note that by our definition, for every \((K, t, x, y) \in \tau_c\), in the ideal world, the key \(K\) is actually given by the “dummy” key vector \(\mathbf{K}\). We thus need to argue that:

- The “dummy” key vector \(\mathbf{K}\) is uniformly distributed, i.e., for any “target” \(K^\dagger = (K^\dagger_1, \ldots, K^\dagger_n)\), the keys \(\mathbf{K}\) picked during the Finalization\((u)\) process of TRPFamily satisfies \(\Pr[\mathbf{K} = K^\dagger] = \frac{1}{2^n}\).

For this, define a set of keys \(\mathcal{K}^c\) as
\[
\mathcal{K}^c := \{K^c = (K^c_1, \ldots, K^c_n) : \forall (i, j), K^c_i = K^c_j \text{ if and only if } K^c_i = K^c_j\}.
\]

Then, the sufficient and necessary condition for TRPFamily to finally reach \(K^\dagger\) is: (i) a key vector \(K^c \in \mathcal{K}^c\) is sampled during Initialization\((u)\), and (ii) the key vector \(\mathbf{K}\), which is essentially sampled from \(\mathcal{K}^c\), hits \(K^\dagger\) during Finalization\((u)\). Therefore,
\[
\Pr[\mathbf{K} = K^\dagger] = \frac{|\mathcal{K}^c|}{2^n}\cdot \frac{1}{|\mathcal{K}^c|} = \frac{1}{2^n}
\]
as expected. For the remaining analysis, we introduce an auxiliary set
\[
\tau_{IC}^c[t] := \{k \in \{0, 1\}^n : (k, t, x, y) \in \tau_{IC}^c \text{ for some } x, y\}.
\]

The sets \(\tau_{IC}^c[0], \tau_{IC}^c[1], \ldots\) constitutes a partition of \(\tau_{IC}^c\), and thus \(\sum_{t \in \{0, 1\}^n} |\tau_{IC}^c[t]| = q_{IC}^c\).

Then, the condition (B-2) can be divided into two subconditions, i.e.,

- (B-21) there exists a KDF query \((K, pk[0, x, y]) \in \tau_B\) such that \((K, pk[0, x, *]) \in \tau_{IC}^c\) or \((K, pk[0, *, y]) \in \tau_{IC}^c\);
- (B-22) there exists a TGF query \((K, W[0, x, y]) \in \tau_B\) such that \((K, W[0, x, *]) \in \tau_{IC}^c\) or \((K, W[0, *, y]) \in \tau_{IC}^c\).

A necessary condition (not yet sufficient) for (B-21) is that there exists a user index \(i\) such that \(K_i \in \tau_{IC}^c[PK_i[0]]\). As argued, \(K_i\) is uniformly distributed. Therefore, conditioned on \(\neg(B-1)\), we have
\[
\Pr[[B-21]] = \sum_{i=1}^n \Pr[K_i \in \tau_{IC}^c[PK_i[0]]] \leq \mu_T \sum_{pk \in \{0, 1\}^{n-1}} \frac{|\tau_{IC}^c[pk][0]|}{2^n} \leq \sum_{pk \in \{0, 1\}^{n-1}} \frac{n|\tau_{IC}^c[pk][0]|}{2^n}.
\]

A necessary condition (not yet sufficient) for (B-22) is that there exists a user index \(i\) such that, there exists a corresponding encryption query \(\text{ILEnc}(i, N, A, M) \rightarrow c\|Z\) or decryption query \(\text{ILDec}(i, N, A, c\|Z)\) that results in a hash evaluation of the form \(H[IC](\text{pad}(A, N, c, T_i)) \rightarrow U\|W\), and that \(K_i \in \tau_{IC}^c[W[0]]\). Therefore, conditioned on \(\neg(B-1)\), we have
With this notation, the ideal world probability is calculated as

\[
\Pr[\text{B-22}] \leq \sum_{i=1}^{u} \sum_{W \leftarrow \{\text{pad}(A,N,e,T),*\} \mid W} \frac{\tau^*_{ic} \cdot |W|}{2^n} \leq \mu_T \cdot \sum_{T \in \{0,1\}^n} \sum_{W \leftarrow \{\text{pad}(A,N,e,T),*\} \mid W} \frac{\tau^*_{ic} \cdot |W|}{2^n} \leq \mu_T \cdot \mu_W \cdot \sum_{W \in \{0,1\}^{n-1}} \frac{\tau^*_{ic} \cdot |W|}{2^n} \leq n^2 \cdot \sum_{W \in \{0,1\}^{n-1}} \frac{\tau^*_{ic} \cdot |W|}{2^n} \]  

(9)

Summing over the two subconditions, we obtain

\[
\Pr[(\text{B-2}) \mid \neg(\text{B-1})] \leq n^2 \cdot \sum_{\ell \in \{0,1\}^n} \frac{|\tau^*_{ic}|}{2^n} \leq \frac{n^2 \cdot |\tau^*_{ic}|}{2^n} \leq \frac{n^2 \cdot \eta_{ic}}{2^n}.
\]  

(10)

Now consider a good transcript \( \tau = (\tau^*_{ih}, \tau^*_{ic}, \tau^*_p, T, K) \). Define

\[
\tau^*_p[K,t] := \{(x,y) \in ((0,1)^n)^2 : (K,t,x,y) \in \tau^*_p\}.
\]

With this notation, the ideal world probability is calculated as

\[
\Pr[T_{id} = \tau] = \Pr(K,T) \cdot \Pr[|\mathcal{C} \vdash \tau^*_{ic}] \cdot \prod_{(K,t)} \frac{1}{(2^n)|\tau^*_p[K,t]|},
\]

where \( \Pr(K,T) \) stands for the probability that the \( u \) user keys take the values \((K,T)\). For the real world distribution, we have

\[
\Pr[T_{re} = \tau] = \Pr(K,T) \cdot \Pr[|\mathcal{C} \vdash \tau^*_{ic}] \cdot \Pr[|\mathcal{C} \vdash \tau^*_p] = \Pr(K,T) \cdot \Pr[|\mathcal{C}^*_K(x) = y \text{ for all } (K,t,x,y) \in \tau^*_p] \cdot \Pr[|\mathcal{C} \vdash \tau^*_{ic}],
\]

Since \( \tau \) is good, \((K,t,*,*) \notin \tau^*_{ic}\) for all \((K,t,x,y) \in \tau^*_p\) (otherwise (B-2) is fulfilled). Therefore,

\[
\Pr[|\mathcal{C}^*_K(x) = y \text{ for all } (K,t,x,y) \in \tau^*_p] = \Pr[|\mathcal{C}^*_K(x) = y \text{ for all } (K,t,x,y) \in \tau^*_p] = \prod_{(K,t)} \frac{1}{(2^n)|\tau^*_p[K,t]|}.
\]

Therefore, for any good transcript \( \tau \) we have

\[
\Pr[T_{re} = \tau] = \Pr[T_{id} = \tau],
\]

and thus

\[
\Pr[|\mathcal{A}^*_{\text{Enc} \rightarrow \text{ILDec} : \mathcal{C} ; \mathcal{C}^{-1} \rightarrow 1}] - \Pr[|\mathcal{A}^*_{\text{Enc} \rightarrow \text{ILDec} : \mathcal{C} ; \mathcal{C}^{-1} \rightarrow 1}] \leq \Pr[T_{id} \in \mathcal{T}_{bad}] \leq \Pr[(\text{B-2}) \mid \neg(\text{B-1})] + \Pr[(\text{B-1})] \leq \frac{(n^2 + 8) \eta_{ic}}{2^n} + \frac{1}{2n^1!} \cdot \left(\frac{4u}{2^n}\right)^n.
\]  

(11)
5.4 Unforgeability of the Idealized TEDT

Denote by $G_1$ (as before) the game that captures the interaction between the muCIML2 adversary $\mathcal{A}$ and TEDT$\mathcal{[IC, TRPFamily]}_\mathcal{T}$. We divide the unforgeability argument into two substeps in two paragraphs below: first, we define and bound several simple bad conditions that may be fulfilled during an execution of the game $G_1$; then, we show that certain chains of queries never appear during the interaction between the muCIML2 adversary $\mathcal{A}$ and TEDT$\mathcal{[IC, TRPFamily]}_\mathcal{T}$. As a decryption query returning non-empty message is necessarily validated by such chains of queries, the above indicates that $\mathcal{A}$ is unable to forge in $G_1$ as long as none of these conditions is fulfilled.

5.4.1 Bad Conditions for Unforgeability

We identify the following conditions during an execution of $G_1$:

- (C-1) there exists two user indices $i, j$ such that $K_i||T_i = K_j||T_j$;
- (C-2) one of the three claims in Lemma 5 is fulfilled, i.e.,
  - collision: there exists $(U, V||W) \neq (U^*, V^*||W^*) \in \tau_H^*$ with $V||W = V^*||W^*$;
  - multi semi-collision: $\mu_W \leq \lambda$.

For (C-1), since $\mathcal{T}$ is uniform, we have $\Pr[(C-1)] \leq \binom{n}{2} \cdot \frac{1}{2^n} = \frac{n^2}{2^n}$. The other bound $\Pr[(C-2)] \leq \frac{8q_{IC}^*}{2^n}$ follows from Lemma 5, where $q_{IC}^*$ is defined in Eq. (7). For simplicity let $\text{Bad} = (C-1) \lor (C-2)$, then

$$\Pr[\text{Bad}] \leq \frac{u^2}{2^n} + \frac{8q_{IC}^*}{2^n}.$$

5.4.2 Unforgeability unless Bad

We call a decryption query $\text{ILDec}_\mathcal{T}(i, N, A, C)$ non-trivial, if the corresponding encryption query $\text{ILEnc}_\mathcal{T}(i, N, A, M) \rightarrow C$ never happened before. Conditioned on $\neg \text{Bad}$, we argue all such non-trivial decryption queries result in $\perp$ except with a bounded probability. Let $C = \pi[Z]$. If this query does not give rise to $\perp$, then right after this query is processed, for some user index $i$, there exists a hash record $(\text{pad}(A, N, c, T_i), V||W)$ and a TRPFamily query $(K_i, W^*||1, V^*, Z)$ in the history, such that $W^* = W$ and $V^* = V$. This means at some time during the execution, the following queries exist in the history:

$$(a, h, g, g \oplus V) \in \tau_H^*, \ (a, h, g \oplus \theta, g \oplus \theta \oplus W||b) \in \tau_H^*, \ (K_i, W^*||1, V^*, Z) \in \tau_R,$$

where $a$ is the last block of $\text{pad}(A, N, c, T_i)$, $b$ is either 0 or 1, and $W^* = W$ and $V^* = V$. We distinguish two cases:

**Case 1:** $(K_i, W^*||1, V^*, Z)$ is created After the pair of IC queries. As $W^* = W$, we simplify the notation as $(K_i, W||1, V^*, Z)$. We argue this query $(K_i, W||1, V^*, Z)$ cannot be forward. For this, assume otherwise, then it’s due to an earlier encryption query $\text{ILEnc}_\mathcal{T}(j, N', A', M') \rightarrow c'||Z$, and that $H(\text{pad}(A', N', c', T_j)) = V||W$ (i.e., it collides with the pair of IC queries in question). Now,

- if $j = i$ and $(N, A, c) = (N', A', c')$, then since we forbid trivial decryption queries, the tag produced by $\text{ILEnc}_\mathcal{T}(j, N', A', M')$ cannot be $Z$, and hence cannot create the query $(K_i, W||1, V^*, Z)$;
- if $j = i$ while $(N, A, c) \neq (N', A', c')$, then by Lemma 3 we have $\text{pad}(A, N, c, T_i) \neq \text{pad}(A', N', c', T_i)$, which then implies a hash collision and contradicts $\neg (C-2)$;
• if \( j \neq i \) and \( K_j \neq K_i \), then the query \( \text{ILEnc}_T(j, N', A', M') \) cannot lead to creating \((K_i, W||1, V^*, Z)\);

• finally, if \( j \neq i \) yet \( K_j = K_i \), then we have \( T_j \neq T_i \) by \( \neg(C-1) \), which by Lemma 3 further implies \( \text{pad}(A, N, c, T_i) \neq \text{pad}(A', N', c', T_j) \) and contradicts \( \neg(C-2) \).

In all, \((K_i, W||1, V^*, Z)\) has to be backward. During the execution of \( G_1 \), the number of such backward queries is at most \( q_d \). Conditioned on \( \neg(C-2) \), the number of records \((U^\dagger, V^\dagger||W^\dagger) \in \mathcal{R}_N \) with \( W^\dagger = W \) is at most \( n \). This implies that the number of “target” \( V^\dagger \) values is also at most \( n \). For each such “target” \( V^\dagger \) and each backward query \((K_i, W||1, V^*, Z)\), we have

\[
\Pr[V^* = V^\dagger] \leq \frac{1}{2^n - q_e - q_d} \leq \frac{2}{2^n}.
\]

Therefore,

\[
\Pr[\text{Case 1} | \neg\text{Bad}] \leq \frac{2nq_d}{2^n}.
\]

**Case 2:** \((K_i, W^*||1, V^*, Z)\) is created **Before the pair of IC queries.** We consider the query \((v, h, g, g \oplus V)\) first. Regardless of its direction, \( V = g \oplus (g \oplus V) \) is uniform in \( \geq 2^n - q_{IC}^* \) possibilities. So

\[
\Pr[V = V^*] \leq \frac{1}{2^n - q_{IC}^*} \leq \frac{2}{2^n}.
\]

As argued before,

\[
\Pr[W = W^*] \leq \frac{2}{2^n - q_{IC}^*} \leq \frac{4}{2^n}.
\]

Therefore, for each such triple of queries, the probability of collision is at most \( \frac{8}{2^n} \). We have at most \( q_d + q_e \) choices for \((K_i, W^*||1, V^*, Z)\), and \( q_{IC}^* \) choices for the pair of IC queries.\(^\text{14}\)

Therefore,

\[
\Pr[\text{Case 2}] \leq \frac{8q_{IC}^*(q_d + q_e)}{2^{2n}}.
\]

Note that these arguments are significantly simplified by the normalization of the game: without the normalization, \((K_i, W^*||1, V^*, Z)\) may be created between the two matching IC queries, which gives rise to many additional cases. Gathering (12) and (13) yields

\[
\Pr[A \text{ forge in } G_2 | \neg\text{Bad}] \leq \frac{2nq_d}{2^n} + \frac{8q_{IC}^*(q_d + q_e)}{2^{2n}}.
\]

This plus \( \Pr[\text{Bad}] \) yield

\[
\text{Adv}_{\text{muCIML2}}^{\text{TEDT}, \text{IC}, \text{TRPFamily} || r, L^*, u} \leq \frac{u^2}{2^{2n}} + \frac{8q_{IC}^*}{2^n} + \frac{2nq_d}{2^n} + \frac{8q_{IC}^*(q_d + q_e)}{2^{2n}}
\]

\[
\leq \frac{u^2}{2^{2n}} + \frac{(n^2 + 9)q_{IC}^*}{2^n},
\]

\[(14)\]

since \( q_{IC}^* \leq 4\sigma + 6(q_e + q_d) + 2q_{IC}, \frac{8q_e(q_d + q_e)}{6 \cdot 2^{2n}} \leq \frac{8q_{IC}^2}{6 \cdot 2^{2n}} \leq \frac{q_{IC}^2}{2^n} \), and \( 2nq_d \leq \frac{n^2 q_{IC}^2}{2^n} \). Then, Eq. (14) and Eq. (11) yield Eq. (1):

\[
\text{Adv}_{\text{muCIML2}}^{\text{TEDT}, \text{IC}, \text{TRPFamily} || r, L^*, u} \leq \frac{u^2}{2^{2n}} + (2n^2 + 17)(4\sigma + 6(q_e + q_d) + 2q_{IC}) + \frac{1}{2n!} \left( \frac{4n}{2^n} \right)^n.
\]

\(^{14}\) Note that the pair \((v, h, g, g \oplus V), (v, h, g \oplus \theta, g \oplus \theta \oplus W||b)\) should be deemed as an **ordered** pair.
5.5 The Necessity of the Inverse

Upon each decryption query \( \text{Dec}_{K,PK}(N, A, c \parallel Z) \), after deriving the digest \( V \parallel W = H(\text{pad}(A, N, c, PK \parallel \{0\})) \), if we use the inverse-free checking “If \( TGF(K, W, V) \neq Z \) then return \( \perp \); else decrypt”, then the forgery attack turns clear: as mentioned, invalid decryption queries leak the correct tag \( TGF(K, W, V) \). Let’s see where our proof approach fails for this design. Concretely, when the query proceeds into the analogue of Case 1 in Section 5.4, the query \( (K, W[1], V^*, Z) \) may be a forward query created during decryption (which is exactly the case of the aforementioned obvious forgery attack), and we are unable to utilize the uniformness of \( V^* \) to complete this argument as we did in section 5.4. Overall, in our model, the inverse of the underlying primitives seems necessary to achieve integrity against decryption/verification leakage. In practice, this means the use of inverse increases robustness against decryption leakage.

6 Leakage CCA Security of TEDT

We now detail the leakage-resistant CCA security of TEDT. The leakage model and assumption are much more complicated than those in Section 5. Therefore, we first state the model in subsection 6.1, then specify the (mild) assumptions in subsections 6.2 and 6.3, present the security claims in subsection 6.4, and give the proof in subsection 6.5.

6.1 Modeling Leakage Functions

We model the leakage as probabilistic efficient functions manipulating and/or computing (partially) secret values. In TEDT, each computation of \( E \) (resp. \( \oplus \)) comes with some additional (internal) information given by \( L_E \) (resp. \( L_{\oplus} \)). However, we make a distinction between the leakages given by KDF and TGF and those given by the less protected calls to \( E \). Indeed, while KDF and TGF both use \( E \), the implementation of these algorithms might offer different levels of protection compared to all the other calls to \( E \) in TEDT, including those of the hash function in Figure 3.

We split the leakage trace resulting from the leaking execution of the TBC \( E \) between its input and output parts: if \( E^I_E(X) \rightarrow Y, L_E(K, T, X, Y) := (L^I_E(K, T; X), L^{out}_E(K, T; Y)) \) with semicolon. While this distinction appears a bit theoretical at first, it better reflects the designers’ implementation goals for each functions/calls and it allows interpreting the security bounds based on cryptanalytic experience (as explained later in section 6.2).

Finally, we insist one more time on the probabilistic feature of the leakage functions \( L^{I}_E, L^{out}_E \), and \( L_{\oplus} \) (which is indeed likely in practice): measuring \( p \) times the leakage from the same computation would not result in completely identical traces. Therefore, we will write \( [L_E]^p \) for the vector of \( p \) leakage traces of \( \oplus \) (and use similar notations for the other operations). Because of the plentiful possible uses of \( E \), we will next denote its input-output leakage function pair as \( (L^I_E, L^{out}_E) \) for simplicity.

Oracle-free leakage function. An artifact of modeling leakages as probabilistic functions is that \( L_E \) might contain “future” calls to \( E \) [DP08]. While benign as a first sight, the possibility for an adversary to call such a leakage function gives him the ability to mount a “future computation attack” [YSPY10]. For instance, if the leakage resulting from \( E(k_0, T, P_1(N)) \rightarrow k_1 \) in TEDT might also contain \( y_2 = E(k_1, T, Q_1(N)) \), the value \( y_2 \) cannot remain unpredictable in the next block. Preventing \( L_E(k_0, *, *) \) from calling \( E(k_1, *, *) \) cannot be achieved only from the tweakable pseudorandom permutation in the standard model. Hence, leakage security (mainly confidentiality) cannot follow. Consequently, for the natural single-pass re-keying encryption used in TEDT, there are only two reliable existing proof approaches which are (i) the leakage simulatability assumption in the standard model [SPY13], and (ii) the non-invertible leakage assumption (or variants)
in the ideal model [YSPY10]. Since we focus here on the ideal TBC model, and since the non-invertible leakages can be more easily measured by cryptanalytic practice (which we believe is important for designing modes), we start following this approach of Yu et al. Nevertheless, we also provide a standard model analysis (of muCCAmL2) based on the former approach in Appendix C.

With this in mind, and to prevent future computation attacks in the ideal model, we assume oracle-free leakage functions [YSPY10]: they cannot make any call to IC, since it is natural for an implementation not to evaluate computations that are unrelated to its current state — e.g., \( E(k_0, *, *) \) in our above example. Therefore, we will say that the leakage function associated to \( E \) is oracle-free, if \( \tau(L_{k}^{E}) = \tau(L_{k}^{\text{ext}}) = \emptyset \), where \( \tau(L_{k}) \) is the transcript of queries and answers made by \( L_{k}^{E} \) to IC when \( L_{k}^{E} \) is evaluated on its inputs.

As discussed in [YSPY10], this model appears to have a natural correspondence with concrete attacks on circuits implementing (tweakable) block ciphers, where the measured leakages can be interpreted as a simple function of the cipher’s input and key during the first few rounds of the computation, and/or as a simple function of the cipher’s output and key during the last few rounds of the computation, but where any useful function of the cipher input and output remains elusive (or is the sign of a completely broken implementation). Also, the use of ideal models does not result in trivial results, as the bounds essentially match simple side-channel attacks to some extent.

### 6.2 Non-Invertible Leakage Assumption

To achieve confidentiality, the leakages have to be somewhat “bounded” — unlike Section 5. To capture this, we follow [YSPY10] and require that the ephemeral TBC keys remain secret in the following sense: the probability that an adversary recovers an ephemeral key before it is being refreshed should be small.

Yu et al. required the adversary \( A \) to precisely output the secret [YSPY10, Definition 2]. We are more generous as we allow \( A \) to output a set of \( q \) guesses instead, and \( A \) wins as long as the secret is in this set. Clearly, this weakens the assumption. Yet, interestingly, this weaker assumption results in better bounds than Yu et al. [YSPY10] (which should be an artifact of the proof technique). More formally, we define

\[
\text{Adv}^{2-up[q]}(A) \quad (A)
\]

\[
def \text{Pr}_{\text{IC}, s_1} \left[ s_2 \leftarrow \text{IC}^{T}(P_A), z \leftarrow \text{IC}^{T}(P_B), \text{Guesses} \leftarrow A^{IC}(s_2, z; \text{leak}) : s_1 \in \text{Guesses} \right] ,
\]

where \( |\text{Guesses}| = q \), and \( A \)'s input \( \text{leak} \) is a list of leakages depending on values \( T, P_A, P_B, s_0 \) specified by \( A \):\(^{16}\)

\[
\text{leak} = \left[ L_{\text{out}}(s_0, T; s_1), L_{\text{in}}^{T}(s_1, T; P_A), L_{\text{out}}^{T}(s_1, T; s_2), L_{\text{in}}^{T}(s_1, T; P_B), L_{\text{out}}^{T}(s_1, T; z) \right]^{P}.
\]

Note that the challenge secret \( s_1 \) is uniformly picked. This emulates the real scenario, in which the target secret is (nearly) uniform in all the possibilities. We further define

\[
\text{Adv}^{2-up[q]}(p, q_{IC}, t) \quad (A)
\]

\[
def \max \left\{ \text{Adv}^{2-up[q]}(A) \right\} ,
\]

where the maximum is taken over all adversaries that repeat their measurements \( p \) times, makes \( q_{IC} \) IC-queries, and runs in time \( t \).

\(^{15}\) Using the standard model and PPT leakage functions, existing provably secure re-keying schemes typically require a more complicated alternating structure [DPF08, Pie09], or public randomness [YSPY10, FPS12] in order to overcome the “future computation” attack.

\(^{16}\) This is a simplified description of a challenge-response process. Granting \( A \) the freedom to choose this \( s_0 \) is for composability purpose (which appears in the proof): the same holds for \( s \) in Equation (19).
Encryption and decryption leakages

As a last step before stating the Theorem 2 allows deriving concrete limits on the capability of the implementation.

\[ \begin{array}{c}
\text{Figure 7: (Left) Illustration on the 2-up assumption; (Right) Basic message manipulation.}
\end{array} \]

Understanding 2-up[q] Advantage. Equation (15) defines a leakage property of a small “unit” of TEDT, pictured in Figure 7 (left). Concretely, it captures that the secret \( s_1 \) cannot be recovered from the involved leakages. Note that \( s_1 \) is used in two subsequent TBC-calls (this is why we call this assumption “2-up”), from which both the “input” and the “output” leakages may contain information on \( s_1 \); this clarifies the presence of \( L^{in}(s_1, T; P_A) \), \( L^{out}(s_1, T; k_2) \), \( L^{in}(s_1, T; P_B) \), and \( L^{out}(s_1, T; y_2) \) in Equation (16). On the other hand, \( s_1 \) itself is the output of the previous cipher-call, and this explains the presence of \( L^{out}(s_0, T; s_1) \), which may contain information on \( s_1 \) as well. The goal of repeating the measurements \( p \) times is to fit into the requirement of providing challenge decryption leakages several times in the muCCAmL2 security game. It should be noted that while this repetition may reduce the measurement noise and make an SPA attack easier, the corresponding advantage of (17) should still be smaller than that of a DPA.

Testers: Measuring in Practice. The concrete values of \( \text{Adv}^{2-up[q]} \) can be measured by running the following tester against the best known SCA adversary \( A \). This along with Theorem 2 allows deriving concrete limits on the capability of the implementation.

1. **Tester for UP \( \text{Adv}^{2-up[q]} \)**
2. Let the challenging adversary \( A \) specify \( s_0 \).
3. Pick the secret \( s_1 \leftarrow \{0,1\}^n \) and compute \( P_{\text{pre}} \leftarrow (E_{\text{pre}}^T)^{-1} (s_1) \).
4. Repeat \( s_1 \leftarrow E_{\text{pre}}^T (P_{\text{pre}}) \), \( s_2 \leftarrow E_{\text{pre}}^T (P_A) \), and \( z \leftarrow E_{\text{pre}}^T (P_B) \) for \( p \) times.
5. Pass the leakages of Step (4) to \( A \). In our model, this means \( L^{in}(s_0, T; s_1), L^{in}(s_1, T; P_A) \), \( L^{out}(s_1, T; s_2), L^{in}(s_1, T; P_B), L^{out}(s_1, T; z) \) are returned to \( A \).
6. Let the challenging adversary \( A \) output \( q \) guesses \( k_1, \ldots, k_q \), the adversary \( A \) wins as long as \( s_1 \in \{k_1, \ldots, k_q\} \).

6.3 Capturing the (In)Security of the XOR

As a last step before stating the muCCAmL2 bound, we have to measure the leakage resulting from XORing the random (looking) block stream with the message blocks in TEDT. To capture this concrete information we follow Pereira et al. [PSV15] and we define

\[ \text{Adv}^{LORL2} (A) \defeq \left| \Pr(_{\text{IC},z} [c^0 \leftarrow z \oplus m^0 : A^{IC}(c^0, \text{leak}_0) \Rightarrow 1] \right. \right. \]
\[ \left. \quad - \Pr(_{\text{IC},z} [c^1 \leftarrow z \oplus m^1 : A^{IC}(c^1, \text{leak}_1) \Rightarrow 1] \right) \right|, \] (18)

where \( \text{leak}_b \) again depends on values \( T, s \) specified by \( A \):

\[ \text{leak}_b = \left[ L^{out}(s, T; z)^p, L_{\oplus}(z, m^b), [L_{\oplus}(z, c^b)]^{p-1} \right]. \] (19)

In the abbreviation \( LORL2 \), the suffix \( L \) stands for “leaking”, and the suffix 2 indicates both encryption and decryption leakages are given. We also define

\[ \text{Adv}^{LORL2} (p, q_{\text{IC}}, t) \defeq \max_A \left\{ \text{Adv}^{LORL2} (A) \right\}. \] (20)
Understanding $LORL^2$ Advantage. Equation (18) defines the information that an adversary might extract from the “basic” message manipulation made in TEDT, which involves XORing as pictured in Figure 7 (right). Concretely, the sensitive point is the key stream block $z$. This block is the output of a TBC-call, hence the presence of $[L^{out}(s, T; z)]^p$ (repeated $p$ times as clarified before). Then, the block $z$ is used to mask the message block, and thus the leakage $L_\oplus(z, m^b)$ comes. Finally, the presence of $[L_\oplus(z, c^b)]^{p-1}$, the leakage from the decryption direction, still stems from the challenge decryption leakage requirements of $muCCAmL^2$.

Like in [HLWW16, PSV15], if a single XOR of the message leaks a single bit, then no $muCCAmL^2$ security would spring up. Thus, on the one hand it is legitimate to focus on this part of leaking implementations. But on the other hand, we cannot claim that $Adv^{LORL^2}(p, q, t)$ is negligible. So our goal here is to faithfully reduce the $muCCAmL^2$ to simple and precise pieces that are more easy to protect as isolated components. This type of methodology is not new in the theory community. For example, it is typically assumed that the PRP advantage of AES is concrete 0.01 rather than “negligible” [Tes11].

Yet, the more critical nature of physical leakages also make us deviate from these results. In some sense, we argue that the advantage degrades in an inevitable rate during the encryption, rather than to argue for some level of security amplification. So in this sense, TEDT is a security-preserving domain extender for a “single-block” encryption operation. In practice, the value $Adv^{LORL^2}$ can be similarly measured by a tester and it is easier to study (and to reduce with relevant protections) than to study entire modes.

1: Tester for $LORL^2$ $Adv^{LORL^2}$
2: Let the challenging adversary $A$ be given $s, T, (m^0, m^1)$.
3: Pick the secrets $y \in \{0, 1\}^n$, $b \in \{0, 1\}$, and compute $P_{pre} \leftarrow (E^T_y)^{-1}(y)$.
4: Compute $c \leftarrow y \oplus m^b$, and repeat $y \leftarrow E^T_y(P_{pre})$ for $p$ times while $m^b \leftarrow y \oplus c$ for $p - 1$ times.
5: Pass $c$ and the leakages of Step (4) to $A$. In our model, this means $(c, [L^{out}(s, T; y)]^p, L_\oplus(y, m^b), [L_\oplus(y, c)]^{p-1})$ are returned to $A$.
6: Let the challenging adversary $A$ output the guess $b'$.

6.4 $muCCAmL^2$ Analysis of TEDT

We define the leakage function $L = (L_{enc}, L_{dec})$ of TEDT as:

- $L_{enc}$, the leakages generated during the encryption:

  - $L^{in}(k, t; x) \& L^{out}(k, t; y)$ generated by internal calls to $E(k, t; x) \rightarrow y$ (excluding KDF- and TGF-calls which are again modeled as leak-free),
  - $L_\oplus(a, b)$ generated by the internal actions $a \oplus b$,
  - all the intermediate values involved in the computations of the hash functions (i.e., hash functions are non-protected, and leak everything).

- $L_{dec}$, the above that are generated during the decryption.

We denote $q = (g_m, q_c, q_d, p - 1, q_{\oplus})$ and we define

$$Adv^{muCCAmL^2}_{TEDT, L}(u, q, t, \sigma) \overset{\text{def}}{=} \max \left\{ Adv^{muCCAmL^2}_{A, TEDT, L, u} \right\},$$

where the maximum is taken over all $(q, t)$-bounded adversaries against $u$ users that have at most $\sigma$ blocks in all their (challenge & non-challenge) queries including AD.

**Theorem 2.** In the ideal TBC model, with the TEDT leakage functions $L = (L_{enc}, L_{dec})$ defined above, if the leakage functions $L^{in}, L^{out}, L_\oplus$ satisfy the assumptions specified by
Equation (15) and Equation (18), then the following holds:

$$\text{Adv}_{\text{TEDT}}^{\mu\text{CCAmL2}}(u, q, t, \sigma) \leq \frac{1}{n^2} \left( \frac{4u}{2^n} \right)^n + \frac{2u^2}{2^n} + \frac{4(n^2 + 9)q^*}{2^n} + \sigma \cdot \text{Adv}_{LORL}^{\mu\text{CCAmL2}}(p, q^*, t^*) + 2\sigma \cdot \text{Adv}_{\text{up}}^{\mu\text{CCAmL2}}(p, q^*, t^*).$$

(21)

where $\text{Adv}_{\text{up}}^{\mu\text{CCAmL2}}$ and $\text{Adv}_{LORL}^{\mu\text{CCAmL2}}$ are defined in Eq. (17) and Eq. (20) respectively, $q^* = 2q_c + 4\sigma + 6(q_c + q_d + q_m)$, $t^* = O(t + \sigma t_l)$, and $t_l$ is the total time to evaluate $L^n$ and $L^{\text{out}}$.

**Interpreting the bound.** We focus on the last two terms since the others have been analyzed before. On the one hand, the term $\sigma \cdot \text{Adv}_{LORL}^{\mu\text{CCAmL2}}(p, q^*, t^*)$ corresponds to the reduction to the “minimal” message manipulation. On the other hand, the term $2\sigma \cdot \text{Adv}_{\text{up}}^{\mu\text{CCAmL2}}(p, q^*, t^*)$ captures the hardness of side-channel key recovery, and it is roughly of some birthday type, namely

$$O \left( \frac{\sigma \cdot q_c + \sigma + t^*}{c \cdot 2^n} \right) = O \left( \frac{(q_c + \sigma + t^*)^2}{c \cdot 2^n} \right),$$

for some parameter $c$ that depends on the concrete conditions. Yet, it is nowadays a common assumption that with such a small data complexity (only 3 relevant leakage traces), the value of $c$ should not be significant [MOP07, Pie09].

The term $\sigma \cdot \text{Adv}_{\text{up}}^{\mu\text{CCAmL2}}$ illustrates the security loss of “hybrid factor” mentioned in the introduction. A standard model-based proof for the black-box security of TEDT would also suffer from a similar term $\sigma \cdot \text{Adv}_{\text{TBC}}^{\mu\text{CCAmL2}}$: this is unavoidable for hybrid-based proofs, see [Men17]. However, for the black-box setting the term $\sigma \cdot \text{Adv}_{\text{TBC}}^{\mu\text{CCAmL2}}$ is not tight, and is merely an artifact of the technique. By contrast, here with leakage the term $\sigma \cdot \text{Adv}_{\text{up}}^{\mu\text{CCAmL2}}$ is tight. This is easy to see: this term captures the collision between the $\sigma$ keying actions, and such a collision allows the adversary to obtain more than 2 leakage traces about a single key, while our Equation (16) only assumed security with 2 traces. In all, our assumption Equation (16), though a bit conservative, tightly results in a birthday-type bound.

### 6.5 Proof of Theorem 2

Let’s see some intuition first. To prove muCCAmL2 security, we need to show:

(i) Messages encrypted by fresh nonce values have confidentiality, and

(ii) Leakages given by the decryption oracle would not affect the above confidentiality.

Since TEDT has been proved muCIML2 (Theorem 1), all non-trivial decryption queries will be invalid, and then the Encrypt-then-MAC design ensures that every such invalid decryption query only leaks a “useless” pseudorandom value, i.e. $(IC_K^{[1]})^{-1}(Z)$ for some $Z$. By these, the above (ii), harmfulness of decryption leakages, is clear. For the above (i), note that essentially, for each fresh nonce $N$, TEDT derives a new initial key $k_0 = E_K^\mu(N)$ (for some $K, T$) and takes it as the key of a TBC-based stream cipher $	ext{TStream}_{k_0}[\text{IC}]$ (which we formally describe in Fig. 8). This $k_0$ is somewhat independent from all the other information gained by the adversary. This enables a standard hybrid argument that reduces the left-or-right confidentiality of challenge messages to the indistinguishability of the leaking encryption of two messages $\text{LTStream}_{k_0}[\text{IC}](M^0)$ and $\text{LTStream}_{k_0}[\text{IC}](M^1)$ under a random initial key $k_0$. Formally, it requires to derive an upper bound for

$$\Pr[A^\mu([\text{LTStream}_{k_0}[\text{IC}])(M^0))] \Rightarrow 1] - \Pr[A^\mu([\text{LTStream}_{k_0}[\text{IC}])(M^1))] \Rightarrow 1]$$

for any adversary $A^\mu$ with bounded power. This indistinguishability resembles the classical eavesdropper security enhanced with leakages. To establish this “leakage eavesdropper security”, we rely on an ideal stream cipher $\text{IStream}$ for “relay”: $\text{IStream}$ encrypts a single message with a true random key stream and offers leakages. Now:
• The “leakage eavesdropper security” of \( IStream \) is easily reduced to \( \text{Adv}^{\text{LORL2}} \);

• The outputs of \( TStream_{k_0}[IC] \) and \( IStream \) can be proved indistinguishable, using the non-invertible assumption on the leakage functions.

The above presented a “top-down” style overview. For the proof, we proceed with a “bottom-up” style flow. In detail, we first formally define and study \( TStream_{k_0}[IC] \) and \( IStream \) in subsection 6.5.1, then bound the leakage eavesdropper security of \( IStream \) in subsection 6.5.2, and then the leakage eavesdropper security of \( TStream_{k_0}[IC] \) in 6.5.3, and finally derive the \( \text{muCCAmL2} \) bound in subsection 6.5.4.

### 6.5.1 TEDT versus the ideal stream cipher

**Description of** \( TStream_{k_0}[IC](T, N, M) \), \( M = m_1 \| \ldots \| m_{\ell} \):

• Initializes an empty list leak for the leakage;

• For \( i = 1, \ldots, \ell \), computes \( k_i \leftarrow \text{IC}_k^{-1}(P_i(N)) \), \( y_i \leftarrow \text{IC}_k(T(Q_{i-1}(N)), \text{c}_i \leftarrow y_i \oplus m_i \), and adds the leakage traces \( [L^\text{in}(k_i-1, T; P_i(N)), L^\text{out}(k_i-1, T; k_i), L^\text{in}(k_i-1, T; Q_{i-1}(N)), L^\text{out}(k_i-1, T; y_i), L_\oplus(y_i, m_i)] \) to the list leak.

• Returns \( (c_1, \ldots, c_{\ell}) \).

**Description of** \( IStream(T, N, M) \), \( M = m_1 \| \ldots \| m_{\ell} \):

• Initializes an empty list leak for the leakage;

• For \( i = 1, \ldots, \ell \), samples \( k_i \leftarrow \{0,1\}^n \) and \( y_i \leftarrow \{0,1\}^n \) such that \( k_i \neq y_i \), sets \( c_i \leftarrow y_i \oplus m_i \), and adds the leakage traces \( [L^\text{in}(k_i-1, T; P_i(N)), L^\text{out}(k_i-1, T; k_i), L^\text{in}(k_i-1, T; Q_{i-1}(N)), L^\text{out}(k_i-1, T; y_i), L_\oplus(y_i, m_i)] \) to the list leak.

• Returns \( (c_1, \ldots, c_{\ell}) \).

We define \( LTStream_{k_0}[IC](T, N, M) = (TStream_{k_0}[IC](T, N, M), \text{leak}) \) for the list leak standing at the end of the above process.

**Figure 8:** The real stream cipher \( TStream_{k_0}[IC] \) and its ideal counterpart \( IStream \).

Formally, the two stream ciphers as well as their leakages are described in Fig. 8. For their indistinguishability, we have the following lemma.

**Lemma 6.** For every \( \ell \)-block message \( M \), every \( T, N \), and every \((q_c, t)\)-bounded distinguisher \( D^{IC} \), it holds

\[
|\Pr[D^{IC}(LTStream_{k_0}[IC](T, N, M)) \Rightarrow 1] - \Pr[D^{IC}(LIStream(T, N, M)) \Rightarrow 1]| \\
\leq \ell \cdot \text{Adv}^{2\cdot \text{up}[q_c]}(p, q_c + 2\ell, t^*)) ,
\]

where \( t^* = O(t + \ell \cdot p \cdot t_l) \), \( t_l \) is the total time needed for evaluating \( L^\text{in} \) and \( L^\text{out} \).

There is no term capturing \( \text{Adv}^{\text{PRP}} \) since it has been captured by the term \( \text{Adv}^{2\cdot \text{up}[q_c]} \); if \( A \) predicates the secret then it clearly breaks the PRP security of \( IC \).

**Proof.** Consider the execution of \( D^{IC} \) upon the inputs \( LTStream_{k_0}[IC](T, N, M) \), \( M = m_1 \| \ldots \| m_{\ell} \). We define a bad event \( \text{BadQuery} \), which occurs when any of the internal keys \( k_0, k_1, \ldots, k_{\ell-1} \) appears in the key field of an \( IC \) query made by \( D^{IC} \). This event, once happens, would cause the key stream blocks lose randomness. We remark that in [YSPY10,
Appendix A], an event with the same meaning termed Query, was defined, and was later proved negligible. So here we just need to adapt Yu et al’s argument to our setting. In detail, given an adversary $\mathcal{D}^{ic}$, we construct an adversary $\mathcal{A}^{ic}$ such that

$$
\text{Adv}^{2\text{-up}[q]}(\mathcal{A}) \leq \Pr[\text{BadQuery in } \mathcal{D}^{ic}(\text{LTStream}_{k_0}[C](T, N, M))].
$$

(22)

Concretely, $\mathcal{A}^{ic}$ runs an instance of $\mathcal{D}$, and keeps record of $\mathcal{D}$’s queries to $\mathcal{C}$ in a set $\gamma_{\mathcal{C}}$ (as defined in previous proofs). $\mathcal{A}^{ic}$ simulates the following process against $\mathcal{D}$:

1. $\mathcal{A}^{ic}$ randomly guesses an index $i \in [0, \ell - 1]$, uniformly samples an initial key $k_0$, and initializes an empty list $\text{leak}$;

2. for $j = 1, \ldots, i - 1$, $\mathcal{A}^{ic}$ queries $\mathcal{C}$ to obtain $k_j \leftarrow \text{IC}_{k_{j-1}}(P_j(N))$ and $y_j \leftarrow \text{IC}^T_{k_{j-1}}(Q_j(N))$, and computes $c_j \leftarrow y_j \oplus m_j$. $\mathcal{A}^{ic}$ then adds the leakage traces $[\text{L}^{in}(k_{j-1}, T; P_j(N)), \text{L}^{out}(k_{j-1}, T; k_j), \text{L}^{out}(k_{j-1}, T; y_j)]^p$, $\text{L}^{\oplus}(y_j, m_j)$, and $[\text{L}_\oplus(y_j, c_j)]^{p-1}$ to leak;

3. $\mathcal{A}^{ic}$ queries $y_i \leftarrow \text{IC}_{k_{i-1}}(Q_{i-1}(N))$ and computes $c_i \leftarrow y_i \oplus m_i$. $\mathcal{A}^{ic}$ then submits $T, P_A = P_{i+1}(N), P_B = Q_i(N)$, and $s_0 = k_{i-1}$ to its $2\text{-up}[q]$ challenger, and (according to our convention) this results in the outputs $(s_2, z, \text{leak})$, where

$$
\text{leak} = \left( [\text{L}^{out}(k_{i-1}, T; s_1), \text{L}^{in}(s_1, T; P_A), \text{L}^{out}(s_1, T; s_2), \text{L}^{in}(s_1, T; P_B), \text{L}^{out}(s_1, T; z)]^p \right)
$$

for the challenge secret $s_1 \in \{0, 1\}^n$. $\mathcal{A}^{ic}$ adds $[\text{L}^{in}(k_{i-1}, T; P_1(N)), \text{L}^{out}(k_{i-1}, T; s_1), \text{L}^{in}(k_{i-1}, T; Q_{i-1}(N)), \text{L}^{out}(k_{i-1}, T; y_i), \text{L}^{\oplus}(y_i, m_i)],$ and $[\text{L}_\oplus(y_i, c_i)]^{p-1}$ to leak as the leakage of the $i$-th iteration.

4. $\mathcal{A}^{ic}$ sets $k_{i-1} \leftarrow s_2$ and $y_{i+1} \leftarrow z$, and computes $c_{i+1} \leftarrow y_{i+1} \oplus m_{i+1}$. It (conceptually) takes the challenge $s_1$ as the key $k_i$ and adds $[\text{L}^{in}(s_1, T; P_{i+1}(N)), \text{L}^{out}(s_1, T; k_{i+1}), \text{L}^{in}(s_1, T; Q_i(N)), \text{L}^{out}(s_1, T; z), \text{L}^{\oplus}(y_i, m_i)],$ and $[\text{L}_\oplus(y_i, c_i)]^{p-1}$ to leak as the leakage of the $i + 1$-th iteration.

5. Then $\mathcal{A}^{ic}$ starts from $k_{i+1}$ to emulate the remaining actions of $\text{LTStream}_{k_0}[C]$ encrypting the tail $m_{i+1} || \ldots || m_{\ell}$ to obtain $c_{i+1} || \ldots || c_\ell$. Eventually, $\mathcal{A}^{ic}$ serves $\text{BadQuery}$ as well as the leakage list $\text{leak}$ to $\mathcal{D}$, and outputs the set $\text{Guesses} = \{ k : (k, t, x, y) \in \gamma_{\mathcal{C}} \text{ for some } t, x, y \}$.

The strategy of $\mathcal{A}^{ic}$ is quite obvious: if $\mathcal{D}$ triggers the event $\text{BadQuery}$ then the key $k$ being queried must be in $\gamma_{\mathcal{C}}$. Therefore, $\mathcal{A}^{ic}$ makes a uniform guess on the position of the first key on which such a query is made; guessing the first queried key ensuring that the key will only be correlated to one thing: the corresponding leakages (and not any previous call on $\mathcal{C}$). This guess will be correct with probability $1/\ell$. Then, $\mathcal{A}^{ic}$ emulates the encryption process of $\text{TStream}_{k_0}[C]$ and provides the leakages to $\mathcal{D}$, except for the index, for which the leakages and $\mathcal{C}$ output are replaced by those obtained from a challenger for the seed-preserving property. If the guess on the index $i$ is correct, all the inputs sent to $\mathcal{D}$ are distributed exactly as those produced by $\text{TStream}_{k_0}[C](T, N, M)$. Therefore, when $\mathcal{D}$ halts, if $\mathcal{D}$ made a query on $s_1$, then simply outputting $\gamma_{\mathcal{C}}$ would break the game. So we have

$$
\Pr[s_1 \in \text{Guesses} | \text{BadQuery in } \mathcal{D}^{ic}(\text{LTStream}_{k_0}[C](T, N, M))] = \frac{1}{\ell}.
$$

Now, we observe that

$$
\Pr[s_1 \in \text{Guesses} | \text{BadQuery in } \mathcal{D}^{ic}(\text{LTStream}_{k_0}[C](T, N, M))]
\leq \frac{\Pr[\text{BadQuery in } \mathcal{D}^{ic}(\text{LTStream}_{k_0}[C](T, N, M))]}{\Pr[\text{BadQuery in } \mathcal{D}^{ic}(\text{LTStream}_{k_0}[C](T, N, M))]},
$$
And it can be seen $A$ is $(qc + 2\ell, t^*)$-bounded for $t^* = O(t + \ell \cdot p \cdot t)$: the factor $p$ before $t_l$ stems from the fact that the leakage functions are evaluated $p$ times in total. By this,

$$
\Pr[\text{BadQuery in } D^S(LTStream_{0}[IC](T, N, M))] \leq \ell \cdot \Pr[s_1 \in \text{Guesses}]
\leq \ell \cdot \text{Adv}_{2-up[qC]}(A) \quad (\text{Eq. 15})
\leq \ell \cdot \text{Adv}_{2-up[qC]}(p, q_C + 2\ell, t^*) \quad (\text{Eq. 17})
$$

During the real execution $D^S(LTStream_{0}[IC](T, N, M))$, as long as the event BadQuery never happens, all the keys and key stream blocks are fresh random values independent from $\gamma_C$ the transcript of IC queries of $D$, and have the same distribution as those in the ideal execution $D^S(LIStream(T, N, M))$. Therefore,

$$
|\Pr[D^S(LTStream_{0}[IC](T, N, M)) \Rightarrow 1] - \Pr[D^S(LIStream(T, N, M)) \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{2-up[qC]}(p, q_C + 2\ell, t^*)
$$

as claimed.

\hfill \Box

### 6.5.2 Leakage eavesdropper security of the ideal stream cipher

We then show how to reduce the leakage eavesdropper advantage of $IStream$ encrypting an $\ell$-block message to the defined term $\text{Adv}^{LORL2}$.

**Lemma 7.** For every pair of $\ell$-block messages $M^0$ and $M^1$ and $(q_C, t)$-bounded adversary $A^{IC}$, it holds

$$
|\Pr[A^{IC}(LIStream(T, N, M^0)) \Rightarrow 1] - \Pr[A^{IC}(LIStream(T, N, M^1)) \Rightarrow 1]| \leq \ell \cdot \text{Adv}^{LORL2}(p, q_C, t^*) + \frac{\ell}{2^n},
$$

where $t^*$ as defined in Lemma 6.

**Proof.** Let $M^0 = m^0_0 \ldots m^0_\ell$ and $M^1 = m^1_0 \ldots m^1_\ell$. We start by building a sequence of $\ell + 1$ messages $M_{h,0}, \ldots, M_{h,\ell}$ starting from $M^0$ and modifying its blocks one by one until obtaining $M^1$. That is, $M_{h,i} := m^0_0 \| \cdots \| m^0_\ell \| m^1_0 \| \cdots \| m^1_\ell$. For any $i$, assuming a $(q_C, t)$-bounded adversary $A^{IC}$ against $IStream(T, N, M_{h,-1})$ and $IStream(T, N, M_{h,i})$, we build a $(q_C, t^*)$-bounded adversary $A^{IC}_2$ against the distribution defined in Eq. (18). In detail, $A^{IC}_2$ proceeds in four steps:

1. $A^{IC}_2$ uniformly samples $k_0$ and initializes an empty list leak;

2. for $j = 1, \ldots, i - 1$, $A^{IC}_2$ uniformly samples random values $k_j, y_j$ such that $k_j \neq y_j$, computes $c_j \leftarrow y_j \oplus m^1_j$, and adds the traces $[L^m(k_{j-1}, T; P_j), L^{out}(k_{j-1}, T; k_j)]^p$, $[L^m(k_{j-1}, T; Q_{j-1}), L^{out}(k_{j-1}, T; y_j)]^p$, $L^z(y_j, m^1_j)$, and $L^z(y_j, c_j)$ to leak;

3. $A^{IC}_2$ samples $k_i$, and submits $s = k_{i-1}, T$ to its LORL2 challenger. According to our convention, $A^{IC}_2$ will receive the outputs $(c_i, \text{leak}_i)$, where the leakage list $\text{leak}_i = ([L^{out}(s, T; z)]^p, L^z(z, m^b), [L^z(z, c^b)]^{p-1})$. $A^{IC}_2$ sets $y_i = z$, and adds the leakages $[L^m(k_{i-1}, T; P_i), L^{out}(k_{i-1}, T; k_i)]^p$, $[L^m(k_{i-1}, T; Q_{i-1}), L^{out}(k_{i-1}, T; y_i)]^p$, $L^z(y_i, m^b)$, and $L^z(y_i, c_i)$ to leak;

4. Then $A^{IC}_2$ starts from $k_i$ to emulate the remaining actions of $IStream$ encrypting the tail $m_{i+1}^0 \| \cdots \| m_{\ell}^0$ to obtain $c_{i+1} \| \cdots \| c_{\ell}$. Eventually, $A^{IC}_2$ serves the ciphertext $c_1 \| \cdots \| c_{i+1} \| c_{i+1} \| \cdots \| c_{\ell}$ as well as all the generated simulated leakages to $A^{IC}$, and outputs whatever $A^{IC}$ outputs.
It can be seen that, as long as \( k_i \neq z \), depending on whether the input tuple received by \( A_2^C \) captures the LORL2 challenger encrypting \( M^0 \) or \( M^1 \), the inputs to \( A^C \) capture \( \Pi \) encrypting \( M_{h,i}^{-1} \) or \( M_{h,i} \). Note that \( \Pr[k_i = z] = 1/2^n \). Moreover, \( A_2^C \) is \((q_C, O(t + \ell \cdot p \cdot t_1))\)-bounded if \( A^C_i \) is \((q_C, t)\)-bounded, which means

\[
\Pr[A^C_i(\Pi(N, M_{h,i}^{-1})) = 1] - \Pr[A^C_i(\Pi(N, M_{h,i})) = 1] \leq \text{Adv}^{\text{LORL2}}(p, q_C, O(t + \ell \cdot p \cdot t_1)) + \frac{1}{2^n}
\]

by Eq. (20). Taking a union bound over the \( \ell \) hybrids yields the main claim. \( \square \)

### 6.5.3 Leakage eavesdropper security of the stream cipher TStream[\( \pi \)]

For simplicity, we define

\[
\text{Adv}^{\text{eavl2}}_{\text{TStream}}(p, q_C, t, \ell) := \max_{A} \left\{ \left[ \Pr[A^C_i(\Pi(N, M)) = 1] - \Pr[A^C_i(\Pi(N, M^0)) = 1] \right] \right\},
\]

where the abbreviation \( \text{eavl2} \) stands for eavesdropper security with encryption and decryption leakages, and the maximal is taken over all adversaries making \( q_C \) queries to IC and running in time \( t \). Gathering Lemmas 6 and 7, we could obtain an upper bound on \( \text{Adv}^{\text{eavl2}}_{\text{TStream}}(p, q_C, t, \ell) \).

**Lemma 8.** For every pair of \( \ell \)-block messages \( M^0 \) and \( M^1 \), \((q_C, t)\)-bounded adversary \( A^C_i \) and \( t^* \) is as defined in Lemma 6, it holds

\[
\text{Adv}^{\text{eavl2}}_{\text{TStream}}(p, q_C, t, \ell) \leq \frac{\ell}{2^n} + \ell \cdot \text{Adv}^{\text{LORL2}}(p, q_C, t^*) + 2\ell \cdot \text{Adv}^{2\text{-ap}[q_C]}(p, q_C + 2\ell, t^*).
\]

**Proof.** By the previous lemmas, we have

\[
\Pr[A^C_i(\Pi(N, M^0)) = 1] - \Pr[A^C_i(\Pi(N, M^1)) = 1] \leq \Pr[A^C_i(\Pi(N, M^0)) = 1] - \Pr[A^C_i(\Pi(N, M^1)) = 1] \leq \frac{\ell}{2^n} \text{Adv}^{\text{LORL2}}(p, q_C, O(t + \ell \cdot p \cdot t_1)) + \frac{1}{2^n}
\]

(by Lemma 7)

\[
+ \sum_{b=0,1} \Pr[A^C_i(\Pi(N, M^b)) = 1] - \Pr[A^C_i(\Pi(N, M^b)) = 1] \leq 2\ell \text{Adv}^{2\text{-ap}[q_C]}(p, q_C + 2\ell, O(t + \ell \cdot p \cdot t_1)) \quad \text{(by Lemma 6)}
\]

Hence the claim. \( \square \)

### 6.5.4 Completing the muCCAmL2 Proof

The core step is a bit similar the standard hybrid-based, eavesdropper-to-CPA security reduction: to prove that encrypting \( M^0_1, M^0_2, \ldots \) is close to encrypting \( M^1_1, M^1_2, \ldots \), we will replace encrypting \( M^0 \) by \( M^1_1 \) one-by-one, and bound the gap introduced by each such replacement using Lemma 8. What makes the situation more complicated is the presence of the decryption queries (since we are considering CCA rather than CPA). Decryption queries may be trivial or non-trivial: recall from subsection 5.4.2 that a decryption query \( \text{LDec}_{\text{K}, \text{T}}(i, N, A, C) \) is trivial if the encryption \( \text{LEnc}_{\text{K}, \text{T}}(i, N, A, M) \to C \) happens before. The leakages of trivial decryption queries may serve new information, thus requiring some treatments. Formally, we start by defining \( G_0 \) as the game capturing the interaction between \( \mathcal{A} \) and \( (\text{LEnc}_{\text{K}, \text{T}}, \text{LDec}_{\text{K}, \text{T}}, \text{Ldecch}, \text{LDec}_{\text{K}, \text{T}}, \text{IC}) \) (cf. subsection 2.2), and \( G_0 \) as the game capturing the interaction between \( \mathcal{A} \) and \( (\text{LEnc}_{\text{K}, \text{T}}, \text{LEnc}_{\text{K}, \text{T}}, \text{Ldecch}, \text{LDec}_{\text{K}, \text{T}}, \text{IC}) \). The subsequent proof consists of several small steps presented in the following paragraphs.
Idealizing TEDT. We then define two intermediate games \( G_1 \) and \( G_1^* \): \( G_1 \), resp. \( G_1^* \), is obtained from \( G_0 \), resp. \( G_0^* \), by replacing all the KDF- and TGF-calls by calls to TRPFamily (as done in section 5.3). By Eq. (11) (although the formalisms are different), with \( q_{ic} = 4^\sigma + 6(q_m + q_c + q_d) + 2q_{c} \) (which is tweaked from Eq. (7)), we have

\[
| \Pr[\mathcal{G}_1 \Rightarrow 1] - \Pr[\mathcal{G}_0 \Rightarrow 1] | \leq \frac{(n^2 + 8)q_{ic}}{2^n} + \frac{1}{2n!} \cdot \left( \frac{4u}{2^n} \right)^n \tag{23}
\]

\[
| \Pr[\mathcal{G}_1^* \Rightarrow 1] - \Pr[\mathcal{G}_0^* \Rightarrow 1] | \leq \frac{(n^2 + 8)q_{i}^*}{2^n} + \frac{1}{2n!} \cdot \left( \frac{4u}{2^n} \right)^n \tag{24}
\]

Excluding non-trivial decryption queries. We then define two new intermediate games \( G_2 \) and \( G_2^* \): they differ from \( G_1 \) and \( G_1^* \) in that:

- They return \( \bot \) (as well as the decryption leakages) for all non-trivial decryption queries.
- They abort when \( \exists i \neq j : K_i || T_i = K_j || T_j \).

The gaps have been bounded by Eq. (14) in section 5.4 (with \( \Pr[\exists i \neq j : K_i || T_i = K_j || T_j] \) included). Therefore,

\[
| \Pr[\mathcal{G}_2 \Rightarrow 1] - \Pr[\mathcal{G}_1 \Rightarrow 1] | \leq \frac{n^2}{2^{2n}} + \frac{(n^2 + 8)q_{ic}^{***}}{2^n} \tag{25}
\]

\[
| \Pr[\mathcal{G}_2^* \Rightarrow 1] - \Pr[\mathcal{G}_1^* \Rightarrow 1] | \leq \frac{n^2}{2^{2n}} + \frac{(n^2 + 9)q_{i}^{***}}{2^n} \tag{26}
\]

The main hybrid-based argument. For this, we denote the \( q_e \) challenge tuples by

\[ (i_1, N_1, A_1, M^1_1, M^1_1), \ldots, (i_{q_e}, N_{q_e}, A_{q_e}, M^0_{q_e}, M^0_{q_e}). \]

Then, we use \( q_e \) hops to replace \( M^0_1, \ldots, M^0_{q_e} \) by \( M^1_1, \ldots, M^1_{q_e} \) in turn, to show that \( G_2 \) can be transited to \( G_2^* \). For convenience, we define \( G_{3,0} = G_2 \), and define a sequence of games

\[ G_{3,1}, G_{3,2}, \ldots, G_{3,q_e}. \]

such that in the \( j \)-th system \( G_{3,j} \), the first \( j \) messages processed by the challenge encryption oracle are \( M^0_1, \ldots, M^0_j \), while the remaining \( q_e - j \) messages being processed are \( M^1_{j+1}, \ldots, M^1_{q_e} \). It can be seen that \( G_{3,q_e} = G_2^* \).

We then show that for \( j = 1, \ldots, q_e, G_{3,j-1} \) and \( G_{3,j} \) are indistinguishable for \( A^{IC} \). For this, from \( A^{IC} \) we build an adversary \( A^{IC}_2 \), such that \( | \Pr[\mathcal{G}_{3,j-1} \Rightarrow 1] - \Pr[\mathcal{G}_{3,j} \Rightarrow 1] | \) is related to \( \text{Adv}_{\text{Test}}^{\text{IC}}(p,q_{IC},O(t),|M^0_j|/n) \). In detail, initially, \( A^{IC}_2 \) samples \( u \) distinct keys \( K_1 || T_1, \ldots, K_1 || T_u \) for subsequent simulations. It also keeps a pair of tables \( (\text{ETABLE}, \text{ETABLE}^{-1}) \), which have entries of the form \( \text{ETABLE}(K,T,X) = Y \) and \( \text{ETABLE}^{-1}(K,T,Y) = X \), to simulate the primitive TRPFamily via lazy sampling (note that TRPFamily is essentially an ideal cipher IC2 independent from IC). It then runs \( A \).

Upon \( A \)’s action, \( A^{IC}_2 \) reacts as follows:

- Upon a query to IC: \( A_2 \) simply relays the query to IC and the answer to \( A \).
- Upon a (non-challenge) encryption query \( (i^*,N^*,A^*,M^* \) from \( A \),
  - if \( (K_{i^*},T_{i^*},C_0(N^*)) \notin \text{ETABLE} \), \( A^{IC}_2 \) samples an initial key \( k_0 \) such that \( (K_{i^*},T_{i^*},k_0) \notin \text{ETABLE}^{-1} \), defines the two entries \( \text{ETABLE}(K_{i^*},T_{i^*},C_0(N^*)) \leftarrow k_0 \) and \( \text{ETABLE}^{-1}(K_{i^*},T_{i^*},k_0) \leftarrow C_0(N^*) \), and then emulates the encryption process \( \text{LTStream}_{k_0}[IC](T_{i^*},N^*,M^*) \) to get the ciphertext \( c^* \) and leakages. \( A^{IC}_2 \) then computes \( V^* || W^* \leftarrow \text{H}[IC](\text{pad}(A^*,N^*,c^*,T_{i^*})) \) and \( Z^* \leftarrow \)
ETABLE($K_{c}^{i}, W^{*}\|1, V^{*}$) (if $(K_{c}^{i}, W^{*}\|1, V^{*}) \notin \text{ETABLE}$ then $A_{\mathbb{C}}^{\ell}$ samples $Z^{*}$ and defines ETABLE($K_{c}^{i}, W^{*}\|1, V^{*} \leftarrow Z^{*}$ as before). For this entire process $A_{\mathbb{C}}^{\ell}$ has to make $4\ell^{*} + 6$ queries to IC with $\ell^{*} = |M^{*}|/n$ and costs $O(p \cdot \ell \cdot t_{i})$ time to evaluate the leakage functions. Finally, $A_{\mathbb{C}}^{\ell}$ returns the ciphertext $c||Z$ and the leakages to $A$:

- If $(K_{c}^{i}, T_{c}^{i}, C_{0}(N^{*})) \in \text{ETABLE}$, $A_{\mathbb{C}}^{\ell}$ simply emulates $\text{LTStream}_{k_{0}}(K_{c}^{i}, T_{c}^{i}, C_{0}(N^{*}))$ with $k_{0} = \text{ETABLE}(K_{c}^{i}, T_{c}^{i}, C_{0}(N^{*}))$, calls $V^{*}\|W^{*} \leftarrow H(\text{pad}(A^{*}, N^{*}, c^{*}, T_{c}^{i}))$ and computes the tag $Z^{*} \leftarrow \text{ETABLE}(K_{c}^{i}, W^{*}\|1, V^{*})$ on the obtained $c^{*}$, and returns $c^{*}\|Z^{*}$ and the leakages to $A$. The cost is similar to the above case.

- Upon a trivial decryption query $(i^{*}, N^{*}, A^{*}, C^{*})$ from $A$, $A_{\mathbb{C}}^{\ell}$ parses $C^{*} = c^{*}\|Z^{*}$, computes $k_{0} = \text{ETABLE}(K_{c}^{i}, T_{c}^{i}, C_{0}(N^{*}))$, and simply emulates the decryption of $\text{LTStream}_{k_{0}}[IC]$ on $(T_{c}^{i}, N^{*}, c^{*})$, and relays the outputs to $A$. The cost is similar to the encryption case.

- Upon a non-trivial decryption query $(i^{*}, N^{*}, A^{*}, C^{*})$ from $A$, $A_{\mathbb{C}}^{\ell}$ parses $C^{*} = c^{*}\|Z^{*}$, and computes $V^{*}\|W^{*} \leftarrow H(\text{pad}(A^{*}, N^{*}, c^{*}, T_{c}^{i}))$. Then,

- if $(K_{c}^{i}, W^{*}\|1, Z^{*}) \notin \text{ETABLE}$, $A_{\mathbb{C}}^{\ell}$ samples $V^{**}$ such that $(K_{c}^{i}, W^{*}\|1, V^{*}) \notin \text{ETABLE}$, and defines the two table entries $\text{ETABLE}(K_{c}^{i}, W^{*}\|1, V^{**}) \leftarrow Z^{*}$; $\text{ETABLE}^{-1}(K_{c}^{i}, W^{*}\|1, Z^{*}) \leftarrow V^{**}$;

- if $(K_{c}^{i}, W^{*}\|1, Z^{*}) \in \text{ETABLE}$, $A_{\mathbb{C}}^{\ell}$ just sets $V^{**} \leftarrow \text{ETABLE}^{-1}(K_{c}^{i}, W^{*}\|1, Z^{*})$.

Now $A_{\mathbb{C}}^{\ell}$ returns $(\perp, V^{**})$ to $A$ (regardless of $V^{**} = V^{*}$ or not).

- Upon $A$ submitting the $\ell$-th challenge tuple $(i_{\ell}, N_{\ell}, A_{\ell}, M_{\ell}^{0}, M_{\ell}^{1})$, it necessarily holds $(K_{i_{\ell}, T_{i_{\ell}}, C_{0}(N_{\ell})}) \notin \text{ETABLE}$, since:

- Previous encryption queries to the $i_{\ell}$-th user cannot define $\text{ETABLE}(K_{i_{\ell}, T_{i_{\ell}}, C_{0}(N_{\ell})}$ due to the challenge nonce-respecting restriction on $A$;

- Previous encryption queries to the $i_{\ell}$-th users cannot define $\text{ETABLE}(K_{i_{\ell}, T_{i_{\ell}, C_{0}(N_{\ell})}$ since $K_{i_{\ell}, T_{i_{\ell}} \neq K_{i_{\ell}, T_{i_{\ell}}}$.

Therefore, depending on $\ell$, $A_{\mathbb{C}}^{\ell}$ reacts as follows:

- When $\ell < j$, it encrypts $M_{\ell}^{0}$ and returns. In detail, $A_{\mathbb{C}}^{\ell}$ samples $k_{0}^{(\ell)}$, sets $\text{ETABLE}(K_{i_{\ell}, T_{i_{\ell}}, C_{0}(N_{\ell})} \leftarrow k_{0}^{(\ell)}$ and $\text{ETABLE}^{-1}(K_{i_{\ell}, T_{i_{\ell}}, k_{0}^{(\ell)} \leftarrow C_{0}(N_{\ell})$, and then emulates $\text{LTStream}_{k_{0}^{(\ell)}}[IC](T_{i_{\ell}, N_{\ell}, M_{\ell}^{0}) \rightarrow c_{\ell}$, performs the tag generation accordingly to produce $Z_{\ell}$ and returns $(c_{\ell}, Z_{\ell})$ and the leakages to $A_{\mathbb{C}}$. The cost is similar to the non-challenge encryption queries.

- When $\ell = j$, it relays $M_{\ell}^{0}$ and $M_{\ell}^{1}$ to its eavesdropper challenger to obtain $c_{\ell}$ and leakages $\text{ leak}_{\text{enc}}$ and $\text{ leak}_{\text{dec}}^{p-1}$, and then performs the tag generation accordingly to produce $Z_{\ell}$ and returns $C_{\ell} = (c_{\ell}, Z_{\ell})$ to $A$. Note that this means the relation $\text{ETABLE}(K_{i_{\ell}, T_{i_{\ell}}, C_{0}(N_{\ell}) = k_{0}^{(\ell)}$ is implicitly fixed, where $k_{0}^{(\ell)}$ is the secret key generated inside the eavesdropper challenger;

- When $\ell > j$, it simply encrypts $M_{\ell}^{1}$ and returns. The details are similar to the described case $\ell < j$.

- Upon $A$ making the $\lambda$-th query to $L_{\text{decch}}(\ell)$ ($1 \leq \lambda \leq p - 1$),

- When $\ell \neq j$, $A_{\mathbb{C}}^{\ell}$ performs the corresponding decryption and returns the obtained leakages to $A$;

- When $\ell = j$, $A_{\mathbb{C}}^{\ell}$ simply returns the $\lambda$-th trace in the vector $\text{ leak}_{\text{dec}}^{p-1}$ as the answer.
It can be seen that, depending on whether $b = 0$ or $1$, the whole process is the same as either $G_{3,j-1}$ or $G_{3,j}$. By the remarks before, besides running $\mathcal{A}$, $A_{2L}^C$ samples at most $2(q_m + q_e + q_d)$ random values (to emulate TRPFamily) and internally processes $q_m + q_e + q_d - 1$ queries (except for the query encrypted by the challenger). Therefore, $A_{2L}^C$ makes $q^* = 4\sigma + 6(q_m + q_e + q_d)$ additional queries to $IC$, and evaluates the leakage functions for $2p\sigma$ times, resulting in $O(p\sigma t_l)$ additional running time. By these, and since $q_C + q^* \leq q^*_C$, we have
\[
\left| \Pr[G_{3,i} \Rightarrow 1] - \Pr[G_{3,i-1} \Rightarrow 1] \right| \leq \text{Adv}^{\text{eval2}}_{\text{TStream}}(p, q^*_C, O(t + p\sigma t_l), \ell_i),
\]
where $\ell_i$ is the number of blocks in the $i$th challenge message. This means
\[
\left| \Pr[G_2 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1] \right| \leq \sum_{i=1}^{q_e} \left| \Pr[G_{3,i} \Rightarrow 1] - \Pr[G_{3,i-1} \Rightarrow 1] \right|
\leq \sum_{i=1}^{q_e} \text{Adv}^{\text{eval2}}_{\text{TStream}}(p, q^*_C, O(t + p\sigma t_l), \ell_i).
\]
(27)
Summing over Eqs. (23), (24), (25), (26), and (27), we eventually reach
\[
\left| \Pr[G_0 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1] \right| \leq \frac{1}{n!} \left( \frac{4u}{2^m} \right)^n + \frac{2u^2}{2^m} + \frac{4n^2 + 34)q^*}{2^n} + \sum_{i=1}^{q_e} \text{Adv}^{\text{eval2}}_{\text{TStream}}(p, q^*_C, O(t + p\sigma t_l), \ell_i),
\]
(28)
with the last term bounded by Lemma 8 as
\[
\frac{\sigma}{2^m} + \frac{\sigma \cdot \text{Adv}^{\text{LORL2}}_{\text{TStream}}(p, q^*_C, O(t + p\sigma t_l))}{2^m} + 2\sigma \cdot \text{Adv}^{2\text{up}[q^*_C]}_{\text{TStream}}(p, q^*_C, O(t + p\sigma t_l)).
\]
Using $\frac{(4n^2+34)q^*}{2^n} + \frac{\sigma}{2^m} \leq \frac{4(n^2+9)q^*_e}{2^n}$, we reach Eq. (21).

7 Performance Evaluation and Related Work

TEDT is designed in order to reach high security guarantees in the presence of leakage. The previous sections gave comprehensive theoretical analyzes of our security claims and physical assumptions. In this section, we finally discuss the cost of these guarantees.

In a first part, we provide performance estimations highlighting the interest of such a leakage-resistant mode of operation enabling leveled implementations, compared to a standard mode where all the blocks require to be uniformly protected against leakage.

In a second part, we provide a discussion of possible competitors. Our main conclusion is that published solutions illustrate an interesting tradeoff between mode-level and implementation-level side-channel resistance. In this respect, a fair comparison would require an evaluation of the performance overheads needed to satisfy the various assumptions used in these modes. We believe such an evaluation and a rigorous discussion of the best way to achieve given levels of confidentiality and integrity in the presence of leakage is among the most important open problems in this field of research.

7.1 Leveled versus Uniform Software Implementations

As a first step in our discussion, we compare two extreme situations with respect to leakage-resistance. First, TEDT reaches the top of the hierarchy in the definitional framework of Guo et al. [GPPS19a] (ignoring black box misuse-resistance) and is therefore expected to minimize the need of implementation-level countermeasures. For this purpose, TEDT
Figure 9: Performance estimations (I): uniformly masked implementation of OCB based on the AES versus leveled implementation of TEDT based on Deoxys-BC-256.

requires 4 TBC calls per message block and a decryption-based tag verification. (The first requirement is due to the need of a two-pass mode in order to ensure CCaml2, where the encryption pass embeds a re-keying process that requires 2 TBC calls per message block and the hashing, based on the Hirose construction, needs another two calls; the second requirement is necessary to reach Cimm2 in the unbounded leakage model). Next, OCB-AES provides no guarantees against leakage, but has an amortized cost of one TBC call per message block and is fully parallelizable (which is not possible with re-keying) [Rog04a]. The rationale behind this choice of modes is to avoid a comparative discussion of different leakage assumptions that is challenging given the current (lack of) understanding of the respective implementation cost needed to fulfill these assumptions.

We consider masking as implementation-level countermeasure, and base our estimations on the state-of-the-art solutions proposed by Goudarzi and Rivain in [GR17]. We specifically exploit Table 3 in this reference, which provides cycle counts in function of the number of shares in the masking scheme. We selected Deoxys-BC-256 as instance of TBC to use in TEDT, which was recently elected as a winner of the CAESAR competition [JNPS]. This choice is mostly motivated by the AES-based structure of this cipher, which enables exploiting the same masking countermeasure. In particular, compared to the AES, Deoxys-BC-256 implies performance overheads of a factor 1.4 to 1.6 for unprotected implementations, and has a total of 224 S-boxes compared to 200 for the AES (i.e., 10 × 16 for the rounds and 10 × 4 for the key rounds). By counting the fraction of time spent on S-boxes (also provided in the work by Goudarzi and Rivain), this allows us to estimate the respective cycle counts for Deoxys-BC-256 and the AES, for various number of shares (with a ratio 1.5 for one share, converging to \( \frac{224}{200} \) as the number of shares increases).

For illustration, we then compare implementations of OCB uniformly protected thanks to masking, and leveled implementations of TEDT. The first ones use \( \ell + 2 \) calls to a masked AES implementation, while the second ones uses 2 calls to masked Deoxys-BC-256 implementations and \( 4\ell \) calls to the unprotected Deoxys-BC-256 implementation. The results of these estimations can be found in Figures 9 and 10 for messages of various sizes. The most interesting conclusion is that starting from 2 shares (i.e., the minimum amount of masking), TEDT compares favorably to OCB, independent of the message size. This is easily observed thanks to the “factor of gain” on the right Y axis of the figures. It is explained by the large overheads paid when moving from an unprotected implementation (that can run in a few thousands of cycles) to a masked one (e.g., a 2-share masked AES takes > 50,000 cycles in [GR17]). The second important conclusion is that the factor of gain approximately converges towards \( \frac{4\ell}{2} \) as the number of shares increases. Quite naturally, the gains get larger and the full convergence requires more shares as the message sizes increase. Concretely, factors of gain larger than 10 can already be observed for medium size messages (e.g., \( \ell = 100 \) blocks) and 4 to 8 shares. Note that
Figure 10: Performance estimations (II): uniformly masked implementation of OCB based on the AES versus leveled implementation of TEDT based on Deoxys-BC-256.

The factor of gain is actually slightly lower than $\ell + 2$, which can be explained by the slightly worse performances (and larger number of S-boxes) of Deoxys-BC-256 compared to the AES. In this respect, it is worth mentioning that combining TEDT with an underlying cipher specifically optimized for efficient masked implementation, such as the LS-designs in [GLSV15], would actually allow even better results. Note also that these estimations ignore the cost needed to protect the whitening values $\Delta$’s in OCB.

We insist that these estimations are only used to give trends and show that mode-level side-channel resistance gains interest as physical security becomes a concern (exact figures require algorithm-specific optimizations that are out of scope here). We also mention that we expect similar (energy) gains in hardware. The only difference is that the energy overheads of the masking countermeasure may then be due to an increase of the cycle count (as in software) and to an increased area [GMK17]. Also, hardware implementations provide additional opportunities to implement the weakly protected implementation at minimum energy cost (e.g., thanks to low-latency designs [KDH+12]).

7.2 Competing Modes of Operation & Discussion

We conclude this work with a brief overview of other approaches to leakage-resistant (or resilient) authenticated encryption and emerging challenges for their fair comparison.

Starting with seed results, and as already mentioned in Section 3.1, Berti et al. [BKP+18, BPPS17] and Guo et al. [GPPS19a] introduced several security definitions and modes of operation for leakage-resistant AEADs. While their security definitions are the starting point of the TEDT design, we extend their work in several directions. (i) TEDT is designed to offer strong mu security, which is ignored in previous modes, and which, as mentioned before, is highly important for practical use. (ii) TEDT is fully specified, while their proposals left several questions open, including the way to implement a hash function. (iii) TEDT is analyzed in several security models, including the ideal cipher model based on very mild assumptions on leakages (one-wayness) with tight security bounds, while such a tight security analysis is not available for other modes.17

Following with competitors, the most relevant and conceptually closest proposal is ISAP, which is an elegant two-passes sponge-based AEAD originally aiming at mode-level DPA resistance [DEM+17]. Subsequently, two works independently proposed methodologies for formally analyzing the leakage security of sponge/duplex designs [GPPS19b, DM19]. While being based on different constructions and primitives, ISAP uses the Encrypt-then-MAC pattern that is similar to TEDT and the analysis in [GPPS19b] suggests that its leveled implementations achieve similar security guarantees as TEDT under different assumptions.

17 While tight security proofs under idealized assumptions must admittedly always be interpreted with care, it remains that they are desirable targets in general, and practically-relevant in many cases.
Namely, its strongly protected components depend on a rate-1 sponge-based re-keying function (rather than a masked TBC) and its tag verification is not inverse-based so has to be protected against DPA to avoid forgeries. Analyzing ISAP and TEDT reveals why comparing leakage-resistant modes of operation is challenging and requires more research. For example, it depends on (i) whether the sponge-based re-keying of ISAP gives better or worse concrete security guarantees than the masked TBC of TEDT for a given cost/performance target, which additionally depends on whether encryption or decryption leakages are considered, and (ii) whether a DPA-resistant tag verification is preferable or not to an inverse-based tag verification which can leak in full.

Yet, one concrete advantage of ISAP over TEDT is that it anyway enjoys simplified passes: two TBC calls are replaced by a single permutation calls. Several other modes take advantage of the appealing structure of sponges for leakage-resistance (or resilience). For example, Guo et al. proposed a sponge-based variant of TEDT that they denote as TEDTSponge and (as key contribution) a single-pass design TETSponge that only guarantees CCAmL1 (and CIML2) security under similar assumptions as TEDT. It serves as a basis for the Spook candidate to the NIST lightweight cryptography standardization effort (https://www.spook.dev). The Ascon cipher, both elected as a winner of the CAESAR competition and candidate to the NIST lightweight cryptography standardization effort (https://ascon.iak.tugraz.at) provides similar guarantees (CCAmL1 and CIML2) under different assumptions: its leveled implementations use masking as TETSponge but its tag verification is not inverse-based [GPPS19b]. In general, the sponge duplex construction (without complementary ingredients) also provides some (weaker) security guarantees (formally, CPA security and ciphertext integrity with leakage in encryption only) at low(er) cost [GPPS19b, DM19], which means that implementation-level countermeasures are then needed if stronger security targets must be reached. So overall, comparing the security that these proposals in a fair manner requires understanding the implementation overheads needed to fulfill their respective physical assumptions. We believe the exhibition of this mode-level vs. implementation-level side-channel security tradeoff is an important conclusion of this work and opens interesting avenues for further investigations.

We finally mention the work of Barwell et al. [BMOS17] who propose a mode of operation and a choice of PRF that satisfy a complementary definition of leakage-resilient and misuse-resistant AEAD (rather than leakage-resistant and misuse-resilient for TEDT). Their security definition is different from ours, notably in the fact that they do not offer to the adversary any leakage that would come from queries in which the game challenger would encrypt either a real or a random message. Such a definition could classify as secure an implementation that leaks plaintexts in full but, on the other hand, makes it feasible to require misuse-resistance instead of misuse-resilience. Their mode of operation is also incompatible with leveled implementations: each block-cipher call needs to be equally well protected, and they propose using, for that purpose, a PRF that is based on pairings. As such, the (e.g., energy) efficiency of a uniformly protected implementation of their mode would be considerably lower than a leveled implementation of TEDT as we propose.

Acknowledgments. Chun Guo was partly supported by the Program of Qilu Young Scholars of Shandong University. Thomas Peters is a postdoctoral researcher and François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the

---

18 Its confidentiality relies on oracle-free leakages but since permutation-based and contrary to TEDT, it is unknown whether its leakage-resistance can be proven without such an idealized assumption.

19 As for black box security guarantees, ISAP has not been designed for mu security either, and is actually quite sensitive to mu attacks on its authentication part. It is unknown if our techniques could be used in ISAP to avoid mu security degradation without increasing the secret key-size.

20 A similar single-pass design can also be derived from TEDT with a similar limitation to CCAmL1, which we illustrate in supplementary material, Appendix E for completeness.
ERC project SWORD (724725), the INNOVIRIS projects SCAUT and C-Cure, and the European Union and Walloon Region FEDER USERMedia project 501907-379156.

References


[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijay-

[Rog04a] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refine-

[Rog04b] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refine-


Supplementary Material

Common belief arguing about the infeasibility to achieve nonce-misuse resistance with (non-restricted) leakage is given in Appendix A. Appendix B shows that XEX is not an ideal TBC. In the misuse-resilience black-box setting (i.e. without leakage), TEDT competes with standards and even outperforms them in the multi-user setting. Unfortunately, the details and the proofs of its optimal security must have been deferred to Appendix D due to space constraints. Eventually, Appendix C shows that even in the standard model TEDT keeps its leakage confidentiality. This final analysis simply gives us more confidence in the rationale of our mode.

A Full Nonce Robustness (Misuse-Resistance) is Hard to Achieve Facing Leakage

(Very informal) The hardness of realizing misuse-resistance in the leakage setting was argued from two different ideas due to Berti et al. [BKP+18] and Guo et al. [GPPS19a]. Below we elaborate in detail. We ignore the associated data $A$ since it’s irrelevant here. Concentrating on the re-keying designs, if nonce $N$ is fixed in CPA attacks, then the initial state of the re-keying is fixed, and likely a same ephemeral key will appear in each encryption process. This allows the adversary to observe long-term information about this ephemeral key and recover it via SPA [BKP+18]. Such a “state-fixing” attack is always possible unless the AEAD scheme first derives a digest $D \leftarrow f(N, M)$ that depends on both $N$ and $M$. But in the latter case, the scheme has to rely on: (i) a variable input length PRF $f_K$, which again faces the problem of “state-fixing” attacks, or (ii) a keyless hash function $f$ (like that used in SIVAT [BMOS17]), which leaks easy-to-compare information about the message $M$ and ruins confidentiality (we stress that SIVAT was designed for applications without challenge leakage, so our observation does not contradict the security claim in [BMOS17]). The above “state-fixing” attack won’t be effective on the leakage-resilient PRF/PRP-based DAE or MR AE schemes (such as those proposed by Barwell [BMOS17]): it’s effective on re-keying designs just because the latter are expected to use PRF/PRPs that are weak w.r.t. leakage-resilience. But for these DAE schemes, another possibility for distinguishing occurs. Concretely, virtually all (D)AE perform actions on the message block-by-block. For example, in the SIV(-like) composition, message is absorbed block-by-block by a universal hash function (SCT [PS15] and ZAE [IMPS17] are two concrete examples); in sponge, message is absorbed block-by-block by a iterative process (e.g., SUNDAE [BBLT18]). In all, the prefix of the leakage $L_{enc}(m_1 \parallel m_2 \parallel \ldots \parallel m_\ell)$ only depends on $m_1$ (or $m_1$ and $m_2$, when Feistel networks are used). Therefore, using $L_{enc}(m_1 \parallel \ldots \parallel m_\ell)$ as a template, for $M^0 = m_1 \parallel \ldots$, and $M^1 = m'_1 \parallel \ldots$, $m'_1 \neq m_1$, it’s easy to distinguish $L_{enc}(M^0)$ from $L_{enc}(M^1)$ and break the LOR game. In all, while there isn’t any formal analysis, “full” robustness against nonce-misuse in the presence of challenge query leakage seems impossible. We further stress that: (1) The (informal) attacks described here never contradict the security claims in the mentioned papers [PS15, IMPS17, BBLT18, BMOS17]: the former three only claimed security without leakage, while the last [BMOS17] is only tailored to the setting without challenge query leakage; (2) The impossibility of “full” robustness against nonce-misuse with challenge query leakage isn’t an artifact of the security model. Instead, it’s the reality. It shows if challenge query leakage cannot be avoided, then one has to manage the nonce more carefully.
B  XEX is not an Ideal TBC

We first recall the definition of XEX, which was introduced by Rogaway [Rog04b] as an instance of a construction of [LRW11]. XEX uses $\Delta = x_1^{a_1} \cdots x_t^{a_t} E_k(T)$ to mask the input and output of a classical block cipher $E$, where $x_1, \ldots, x_t \in \mathbb{GF}(2^n)$ are some pre-defined generators. Formally,

$$XEX_k((\alpha_1, \ldots, \alpha_t, T), m) = E_k(m \oplus \Delta) \oplus \Delta.$$  

If the generators and the tweak space are defined such that the $x_1^{a_1} \cdots x_t^{a_t}$ are unique and unequal to 1 for all tweaks, XEX achieves birthday bound security [Rog04b]. For example, practical applications with $n = 128$ often take $\ell \leq 3$ and $(x_1, x_2, x_3) = (2, 3, 7)$, and an allowed tweak space would be $[1, 2^{n/2}] \times [0, 10] \times [0, 10] \times \{0, 1\}^n$.

Note that the security of XEX crucially relies on the secretness of the key $k$: otherwise, regardless of the choice of $x_1, \ldots, x_t$, the adversary could find $\alpha_1, \ldots, \alpha_t, T, \alpha_1', \ldots, \alpha_t', T'$ such that $x_1^{a_1} \cdots x_t^{a_t} E_k(T) = x_1^{a_1} \cdots x_t^{a_t} E_k(T')$ via querying $E_k^{-1}$, which then satisfy $XEX_k((\alpha_1, \ldots, \alpha_t, T), m) = XEX_k((\alpha_1', \ldots, \alpha_t', T'), m)$ for any $m$. By this, XEX is not secure in the known-key setting [KR07] and thus not ideal.

C  An Analysis in the Standard Model

Compared to subsection 6.5, the analysis is even closer to the FEMALE proof in [GPPS19a]. To ease comparison, in this subsection we adopt the notations of [GPPS19a].

Assumptions. In this section, we model the TBC $E$ as a strong tweakable pseudorandom permutation, defined as follows.

Definition 1 (Strong Tweakable Pseudorandom Permutation). A function $E : K \times TW \times M \rightarrow M$ is a $(q, t, \varepsilon_E)$-strong tweakable pseudorandom permutation (STPRP) for a security parameter $n$ if, for all $(q, t)$-bounded adversaries $A$, we have

$$\Pr[k \xleftarrow{} K : A^{E_k \cdot E_k^{-1}}(1^n) \Rightarrow 1] - \Pr[P \xleftarrow{} \mathcal{T}P : A^{P \cdot P^{-1}}(1^n) \Rightarrow 1] \leq \varepsilon_E,$$

where $\mathcal{T}P$ denotes the set of all tweakable permutations on $M$ and with tweak space $TW$ so that for any tweakable permutation $P$, and for any tweak $tw$, $P^{tw} = P(tw, \cdot)$ and $P^{tw \cdot 1}$ are the inverse of each other.

Further, we assume the leakages are “recyclably simulatable”, which was first introduced in [GPPS19a] and is defined below (adapted to TPRP setting) based on $(p, q)$-rsim-game in Table 2. We abbreviate it as $(p, q)$-recyclable-simulatability. This $(p, q)$-recyclable-simulatability assumption is an extension of the $q$-simulatability notion [SPY13], by allowing each of the $q$ leakages in $q$-simulatability to be repeated $p$ times.

<table>
<thead>
<tr>
<th>Game $(p, q)$-rsim$(\mathcal{A}, E, L, \Delta, b)$.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The challenger selects two random keys $k, k^* \xleftarrow{} K$. The output of the game is a bit $b'$ computed by $\mathcal{A}^k$ based on the challenger responses to a total of at most $q$ adversarial queries of the following type, each repeated at most $p$ times:</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>Response if $b = 0$</td>
</tr>
<tr>
<td>$\text{TEnc}(T, x)$</td>
<td>$E_k^L(x), L(k, T, x)$</td>
</tr>
<tr>
<td>and one query of the following type, repeated at most $p$ times:</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>Response if $b = 0$</td>
</tr>
<tr>
<td>$\text{Gen}(k_{pre}, T_{pre}, x)$</td>
<td>$S^L(k_{pre}, T_{pre}, x, k)$</td>
</tr>
</tbody>
</table>
Definition 2 ((p, q)-recyclable-simulatability of leakages). Let $E$ be a TPRP with $L$ as its leakage function. Then the leakages of $E$ are said to have $(q_S, t_S, q_t, t, \varepsilon_{(p,q)\text{-}rsim})$-recyclable-simulatability, if there exists a $(q_S, t_S)$-bounded simulator $S^k$ such that, for every $(q_t, t)$-bounded adversary $A^k$ (making at most $q_t$ queries to $L$ and running in time $t$), we have:

$$\left| \Pr[(p, q)\text{-}sim(A, E, L, S, 1) = 1] - \Pr[(p, q)\text{-}sim(A, E, L, S, 0) = 1] \right| \leq \varepsilon_{(p,q)\text{-}rsim}.$$

Throughout the remaining, we would simply call such leakages $R$-simulatable. It isn’t hard to see $(p, q)$-recyclable-simulatability captures very similar SPA security setting as $\text{Eq. (15)}$, the non-invertible leakage assumption. Please see [GPPS19a] for additional discussion on this assumption.

To the muCCAmL2 Advantage. We define:

$$\text{Adv}_{\text{muCCAmL2}}^{\text{TEDT}}(q, t, \sigma) \overset{\text{def}}{=} \max \left\{ \text{Adv}_{\text{TEDT}}^k(A) \right\},$$

where $q = (q_m, q_e, q_d, p-1, q_t)$, and the maximal is taken over all $(q, t)$-bounded adversaries: that make $q_m$ queries to the non-challenge encryption oracle, $q_d$ queries to the decryption oracle, $p - 1$ queries to $L_{\text{dec}}$, $q_t$ queries to the challenge encryption oracle, and $q_e$ queries to $L$, run in time $t$, and have at most $\sigma$ blocks in its queries. For the formal analysis, we define the leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ of TEDT as:

- $L_{\text{enc}}$ consists of the follows that are generated during the encryption:
  - the leakages $L_{\text{enc}}(s, T, x)$ generated by all the internal calls to $E_{ch}(T, x)$, and
  - the leakages $L_{\text{aux}}(a, b)$ generated by all the internal actions $a \oplus b$, and
  - all the intermediate values involved in the computations of the hash functions.

- $L_{\text{dec}}$ consists of the above that are generated during the decryption.

Our security reduction is made against (i) the simulatability of the leaking blocks, (ii) the security of the encryption of one single block with a fresh key. In detail, following Pereira et al.’s approach [PSV15], we consider a Leaking Real Single-block Encryption scheme LRSE defined in Fig. 11 as the basic unit of TEDT, and as the leakage confidentiality assumption for TEDT in the standard model. Since for each generated key $k_{\text{ch}}$ LRSE will be used to encrypt a single message $m$ composed of a single block, we assume that given a security parameter $n$, LRSE is $(p, q_t, t, \varepsilon_{\text{s-block}})$ secure in the following sense: for any $(q_t, t)$-bounded eavesdropper adversary $A^{\text{eave}}$ choosing $T, p_A, p_B, m, m^1 \in \{0, 1\}^n$ with $p_A \neq p_B$, it holds

$$\left| \Pr[A^{\text{eave}}(LRSE_{k_{\text{ch}}}^+(m^0)) = 1] - \Pr[A^{\text{eave}}(LRSE_{k_{\text{ch}}}^-(m^1)) = 1] \right| \leq \varepsilon_{\text{s-block}}. \tag{29}$$

where $LRSE_{k_{\text{ch}}}^+(m^0) = (LRSE_{k_{\text{ch}}}^-(m^0), [L_{\text{dec}}(k_{\text{ch}}, c^b)]^{p-1}, k_{\text{pre}}, k_{\text{up}})$ for the pair of outputs $(c^b, k_{\text{up}}) = RSE_{k_{\text{ch}}}^+(m^0)$. The reason why the adversary also gets the auxiliary outputs $k_{\text{pre}}$ and $k_{\text{up}}$ is for composability purpose which will be apparent in the proof. Similar to Eq. (15), $\varepsilon_{\text{s-block}}$ may not be negligible.

Based on the assumption (29), the muCCAmL2 advantage bound could be established below.

Theorem 3. Let $E : \{0, 1\}^n \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a $(2q_e + 2q_d + 2q_m, t', \varepsilon_{E'})$-STPRP. Assume that $E$ has two implementations: a strongly protected implementation $E^*$ underlying $KDF$ and TGF is leak free, and a plain implementation $E$ have leakage function $L_E$ that is $(q_S, t_S, q_t, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p, 2)$-$R$-simulatable. Then the TEDT implementation with leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ defined before has:

$$\text{Adv}_{\text{muCCAmL2}}^{\text{TEDT}}(q, t, \sigma) \leq 2(q_e + q_d + q_m)\varepsilon_E + \text{Adv}_{\text{muCCAmL2}}^{\text{TEDT}}(q_e + q_d + q_m, t')$$

$$+ 4\sigma(\varepsilon_E + \varepsilon_{(p,2)\text{-}rsim}) + \sigma \cdot \varepsilon_{s\text{-block}}, \tag{30}$$
Description of LRSE scheme: (tool for the proof and for capturing the confidentiality advantage)

\[ \text{RSGen}(1^n) \] picks \( k_{\text{ch}} \leftarrow \{0,1\}^n, M, C = \{0,1\}^n (T, p_A, p_B \in \{0,1\}^n, p_A \neq p_B) \)

\[ \text{RSEnc}_{k_{\text{ch}}}(m) \] returns \( (k_{\text{up}}, c) \), where \( c = y_{\text{ch}} \oplus m, y_{\text{ch}} = E_{k_{\text{ch}}}(T, p_B) \), and \( k_{\text{up}} = E_{k_{\text{ch}}}(T, p_A) \). (The term “up” is short for “update”)

\[ \text{RSDec}_{k_{\text{ch}}}(c) \] proceeds in the natural way.

The leakage \( L_{\text{RSE}} = (L_{\text{rsenc}}, L_{\text{rsdec}}, k_{\text{pre}}) \) resulting from the LRSE implementation is defined as

- \( L_{\text{rsenc}}(k_{\text{ch}}, m) = (L_E(k_{\text{ch}}, T, p_A), L_E(k_{\text{ch}}, T, p_B), L_{\oplus}(y_{\text{ch}}, m), S^t(k_{\text{pre}}, T, p_A, k_{\text{ch}})) \),

- \( L_{\text{rsdec}}(k_{\text{ch}}, c) = (L_E(k_{\text{ch}}, T, p_A), L_E(k_{\text{ch}}, T, p_B), L_{\oplus}(y_{\text{ch}}, c), S^t(k_{\text{pre}}, T, p_A, k_{\text{ch}})) \)

for a fixed random \( k_{\text{pre}} \leftarrow \{0,1\}^n \). As usual we denote \( \text{LRSEnc}_{k_{\text{ch}}}(m) = (\text{RSEnc}_{k_{\text{ch}}}(m), L_{\text{rsenc}}(k_{\text{ch}}, T, p_A, p_B, m)) \).

The proof flow for Theorem 3 is very similar to Section 6.5. We first prove that based on the pseudorandomness of \( E \) and the simulatability of the leakage, the “real-leaking world” \( (E_T^T(p), L(k, T, p)) \) is indistinguishable from the “ideal-simulating” world \( (\$; S^t(k, T, p, \$)) \). This is actually a lemma of Guo et al. [GPPS19a] adapted to our TPRP setting.

Lemma 9 (Indistinguishability of Real-Leak and Ideal-Simulation). Let \( E : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n \) be a \((2, t, \varepsilon_E)\)-TPRP, whose implementation has a leakage function \( L_E \) having \((q_S, t_S, q_d, l, \varepsilon_{(2,2)-\text{rsim}}) \) \((2, 2)\)-R-simulatable leakages, and let \( S^t \) be an appropriate \((q_S, t_S)\)-bounded leakage simulator. Then, for every \( k_{\text{pre}}, T, p_A, p_B, z \in \{0,1\}^n \) and every \((q - q^*, t - t^*)\)-bounded distinguisher \( \mathcal{D}^t \), the following holds:

\[
\left| \Pr_{k_{\text{ch}}} \left[ k_{\text{ch}} \leftarrow \{0,1\}^n : \mathcal{D}^t(E_{k_{\text{ch}}}(p_A), E_{k_{\text{ch}}}(p_B), L_E(k_{\text{ch}}, T, p_A), L_E(k_{\text{ch}}, T, p_B), S^t(k_{\text{pre}}, T, z, k_{\text{ch}}))^p \right] \right| \leq \varepsilon_E + \varepsilon_{(2,2)-\text{rsim}}.
\]

Here \( q^* = 3p \cdot q_S \), while \( t^* = \text{Max}\{t_r, t_{\text{sim}}\} \), in which \( t_r \) is equal to \( 3p \cdot t_S \) augmented with the time needed to make 2 oracle queries to the TPRP challenger and select a uniformly
random key in \( \{0,1\}^n \), and \( t_{\text{sim}} \) is the time needed to relay the content of 2p \( T\text{Enc} \) and \( p \) \( \text{Gen} \) queries from and to a \( (p,2)\)-rsim challenger.

Proof. The proof consists of two simple transitions: first replace the real leakages by simulated ones, relying on recyclable simulatability assumption, then replace \( E_{k_{ch}}(T,p_A) \) and \( E_{k_{ch}}(T,p_B) \) by two distinct random values to obtain the target inputs, relying on the assumption that \( E \) is a TPRP.

We then define the tweakable variant of the “single-block scheme” in [GPPS19a]: roughly, all the intermediate values are replaced by random and all the leakages are replaced by simulation. The result is algorithm in Fig. 12. We also define \( \text{LISEnc}_{k_{ch}}(m) = (\text{LISEnc}_{k_{ch}}(m), [L_{\text{dec}}(k_{ch}, c)]^{n-1}, k_{pre}) \) for \( (c, k_{up}) = \text{LISEnc}_{k_{ch}}(m) \).

**Lemma 10 (Indistinguishability of \( \text{ISEnc} \) and \( \text{RSEnc} \)).** Let \( E : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n \) be a \( (2, t, \varepsilon_E) \)-TPRP, whose implementation has a leakage function \( L_E \) having \( (q_S, t_S, q_t, t, \varepsilon_{(p,2),\text{rsim}}) \) \( (p,2)\)-R-simulatable leakages, and let \( S_k \) be an appropriate \((q_S, t_S)\)-bounded leakage simulator. Then, for every \( T, p_A, p_B \in \{0,1\}^n \), \( p_A \neq p_B \), and every \((q_t - q^*, t - t^*)\)-bounded distinguisher \( D_k \), the following holds:

\[
\Pr[D_{\text{isec}}(m, L_{\text{RSEnc}}_{k_{ch}}(m)) \Rightarrow 1] - \Pr[D_{\text{isec}}(m, \text{LISEnc}_{k_{ch}}(m)) \Rightarrow 1] \leq \varepsilon_E + \varepsilon_{(p,2),\text{rsim}}.
\]

Here \( q^* = 3p \cdot q_S + p \), while \( t^* = \max\{t_r, t_{\text{sim}}\} \), in which \( t_r \) is equal to 3p \( \cdot t_S + 2t_{\oplus} \) augmented with the time needed to make 2 oracle queries to the TPRP challenger and select a uniformly random key in \( \{0,1\}^n \), \( t_{\oplus} \) is the time needed to evaluate the \( \oplus \) action on an \( n\)-bit input, and \( t_{\text{sim}} \) is the time needed to relay the content of four \( T\text{Enc} \) and two \( \text{Gen} \) queries from and to a \((p,2)\)-rsim challenger.

Proof. Follows the same line as Lemma 9.

We then define \((L)T\text{Stream} \) and \((L)I\text{Stream} \) in the standard model in Fig. 13 and prove their indistinguishability.

**Lemma 11 (Indistinguishability of \( \text{LTStream} \) and \( \text{LIStream} \)).** Let \( E : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n \) be a \( (2, t, \varepsilon_E) \)-TPRP, whose implementation has a leakage function \( L_E \) having \( (q_S, t_S, q_t, t, \varepsilon_{(p,2),\text{rsim}}) \) \( (p,2)\)-R-simulatable leakages, and let \( S_k \) be an appropriate \((q_S, t_S)\)-bounded leakage simulator. Then, for every \( \ell\)-block message \( m \), every \( T, N, p_A, p_B \), and every \((q_t - q^*, t - t^*)\)-bounded distinguisher \( D_k \), the following holds:

\[
\Pr[D_{\text{lisec}}(m, \text{LTStream}_{k_{ch}}(T, N, m)) \Rightarrow 1] - \Pr[D_{\text{lisec}}(m, \text{LIStream}_{k_{ch}}(T, N, m)) \Rightarrow 1] \leq \ell(\varepsilon_E + \varepsilon_{(p,2),\text{rsim}}).
\]

Here \( q_t = \ell(2q_S + 3) \), \( q^* \) and \( t^* \) are as defined in Lemma 10, and \( t_r = 2(t_S + t_S + t_E) + \ell \cdot t_{\oplus} \), where \( t_S \) is the time needed for evaluating \( E \) once, \( t_S \) is the time needed for randomly sampling a value from \( \{0,1\}^n \), and \( t_{\oplus} \) is the time needed for evaluating \( \oplus \) once.
Description of $T\text{Stream}$:
- $\text{Gen}$ picks $k_0 \xleftarrow{} \{0,1\}^n$
- $T\text{Stream}_{k_0}(T, N, m_1, \ldots, m_t)$ proceeds in two steps:
  1. Initializes an empty list $\text{leak}$ for the leakage;
  2. For $i = 1, \ldots, t$, computes $k_i \leftarrow E_{k_{i-1}}(C_i(N)), y_i \leftarrow E_{k_{i-1}}(T, D_{i-1}(N))$, and $c_i \leftarrow y_i \oplus m_i$, and adds $|L_{D}(k_{i-1}, T, C_i(N)), L_{E}(k_{i-1}, T, D_{i-1}(N))|^p$, $L_{\oplus}(y_i, m_i)$, and $|L_{\oplus}(y_i, c_i)|^{p-1}$ to the list leak.

Description of $I\text{Stream}$:
- $I\text{Stream}_{k_0}(T, N, m_1, \ldots, m_t)$ proceeds in two steps:
  1. Initializes an empty list $\text{leak}$ for the leakage;
  2. For $i = 1, \ldots, t$, samples $k_i \xleftarrow{} \{0,1\}^n$ and $y_i \xleftarrow{} \{0,1\}^n$ such that $k_i \neq y_i$, sets $c_i \leftarrow y_i \oplus m_i$, and adds $|S^1(k_{i-1}, T, C_i(N), k_i), S^1(k_{i-1}, T, D_{i-1}(N), y_i)|^p$, $L_{\oplus}(y_i, m_i)$, and $|L_{\oplus}(y_i, c_i)|^{p-1}$ to the list leak.

\text{LTStream}_{k_0}(T, N, m) = (T\text{Stream}_{k_0}(T, N, m), \text{leak})$ for the list leak standing at the end of the above process.

\text{LIStream}_{k_0}(m) = (I\text{ESM}_{k_0}(m), \text{leak})$ for the list leak standing at the end of the above process.

**Figure 13:** The $T\text{Stream}$ and $I\text{Stream}$ scheme.

\textbf{Proof.} We define $G_0$ as the security game in which $A^k$ receives $T\text{Stream}_{k_0}(T, N, m)$ as the input, and $G_r$ as the game in which $A^k$ receives $I\text{Stream}_{k_0}(T, N, m)$ as the input. We show that $G_0$ could be transited to $G_r$ via a sequence of games $G_1, G_2, \ldots, G_{\ell-1}$. In detail, for $i = 1$ from $1$ to $\ell$, we consider the game $G_{i-1}$: we replace the two intermediate values $E^T_{k_{i-1}}(C_i(N))$ and $E^T_{k_{i-1}}(D_{i-1}(N))$ by two distinct random values $k_i$ and $y_i$, and replace the leakages $|L_{D}(k_{i-1}, T, C_i(N)), L_{E}(k_{i-1}, T, D_{i-1}(N))|^p$, $L_{\oplus}(E^T_{k_{i-1}}(D_{i-1}(N), m_i))$, and $|L_{\oplus}(E^T_{k_{i-1}}(D_{i-1}(N), c_i))|^{p-1}$ by the simulated $|S^1(k_{i-1}, T, C_i(N), k_i), S^1(k_{i-1}, T, D_{i-1}(N), y_i)|^p$, $L_{\oplus}(y_i, m_i)$, and $|L_{\oplus}(y_i, c_i)|^{p-1}$. This yields the game $G_i$.

We next derive an upper bound for $|\text{Pr}[(D^k)^{G_\ell} \rightarrow 1] - \text{Pr}[(D^k)^{G_{\ell-1}} \rightarrow 1]|$. For this, we assume a $(q_y - p \cdot q_y - q_t', t - p \cdot t', t')$-bounded distinguisher $D^k$ against $G_i$, and $G_{\ell-1}$, and we build a distinguisher $D^k$ against the real-leaking and the ideal-simulation-world. Assume $D^k$ receives $(c_A, c_B, [\text{leak}_1, \text{leak}_2], S^k(k_{\text{pre}}, T, C_{i-1}(N), k_{ch})^p, k_{\text{pre}})$ as inputs, with $c_A \neq c_B$. $D^k$ proceeds in six steps:

1. $D^k$ first uniformly samples $k_0$;
2. For $j = 1, \ldots, i-3$, $D^k$ uniformly samples 2 random values $k_j \neq y_j$, simulates the traces $|S^1(k_{j-1}, T, C_j(N), k_j), S^1(k_{j-1}, T, D_{j-1}(N), y_j)|^p$, computes $c_j \leftarrow y_j \oplus m_j$ and $m_j \leftarrow y_j \oplus c_j$ and obtains the traces $L_{\oplus}(y_j, m_j)$ and $|L_{\oplus}(y_j, c_j)|^{p-1}$;
3. $D^k$ then sets $y_{i-2} \leftarrow k_{\text{pre}}$, samples $y_{i-2} \neq k_{\text{pre}}$, obtains $|S^1(k_{i-3}, T, D_{i-3}(N), y_{i-2})|^p$, computes $c_{i-2} \leftarrow y_{i-2} \oplus m_{i-2}$ and $m_{i-2} \leftarrow y_{i-2} \oplus c_{i-2}$ and obtains the traces $L_{\oplus}(y_{i-2}, m_{i-2})$ and $|L_{\oplus}(y_{i-2}, c_{i-2})|^{p-1}$. These along with $|S^1(k_{\text{pre}}, T, C_{i-1}(N), k_{ch})|^p$ are used as the leakages of the $i-2$ th iteration;
4. Then it uniformly samples $y_{i-1}$, computes $c_{i-1} \leftarrow y_{i-1} \oplus m_{i-1}, m_{i-1} \leftarrow y_{i-1} \oplus c_{i-1}$, and uses $|S^1(k_{i-2}, T, C_{i-1}(N), k_{ch}), S^1(k_{i-2}, T, D_{i-2}(N), y_{i-1})|^p$, $L_{\oplus}(y_{i-1}, m_{i-1})$, $|L_{\oplus}(y_{i-1}, c_{i-1})|^{p-1}$ as the traces of the $i-1$ th iteration in the first pass;
5. Sets $k_i \leftarrow c_A$ and $y_i \leftarrow c_B$, computes $c_i \leftarrow y_i \oplus m_i$, and uses $|\text{leak}_1, \text{leak}_2|^p$, $L_{\oplus}(y_i, m_i)$, $|L_{\oplus}(y_i, c_i)|^{p-1}$ as the corresponding leakages;
(6) Takes $k_i$ as the starting point and emulates the remaining part of the execution of $\mathsf{TStream}$ encryption. Eventually, $\mathcal{D}^{L'}$ serves the obtained ciphertext $c_1|\ldots|c_\ell$, as well as the leakage traces to $\mathcal{D}^{L'}$, and outputs whatever $\mathcal{D}^{L}$ outputs.

It can be seen that depending on whether the input tuple received by $\mathcal{D}^{L'}$ is real-leaking or ideal-simulation, $\mathcal{D}^{L}$ is interacting with $G_{i-1}$ or $G_i$.

We further show that to perform the additional operations, $\mathcal{D}^{L'}$ makes at most $p \cdot s_r$ additional queries to $L$ and spend $p \cdot t_r$ additional time. To this end, we note that the encryption process of $\mathsf{TStream}$ involves $2\ell - 1$ calls to $E$ and $\ell$ xor operations. Moreover, to emulate the “hybrid” encryption process once, $\mathcal{D}^{L'}$ needs at most $(2\ell - 1)(q_S + 1) + \ell = q_r$ queries to $L$ and $(2\ell - 1)(t_E + t_S + t_{\oplus}) + \ell \cdot t_{\oplus} = t_r$ running time. To obtain the required decryption leakage traces, $\mathcal{D}^{L'}$ has to additionally perform the “hybrid” decryption process for $p - 1$ times, which contributes to $(p - 1)q_r$ more queries and $(p - 1)t_r$ more time. Therefore, as claimed, $\mathcal{D}^{L'}$ makes at most $p \cdot q_r$ additional queries to $L$ and spends $p \cdot t_r$ additional time for the additional operations. By the above and Lemma 9, we have

$$|\Pr[\{(\mathcal{D}^{L})^{G_i} \Rightarrow 1]\} - \Pr[\{(\mathcal{D}^{L})^{G_{i-1}} \Rightarrow 1]\}] \leq \varepsilon_E + \varepsilon_{(p,2),\text{rsim}}.$$ 

Therefore, the $\ell$ transitions yield

$$|\Pr[\mathcal{D}^{G_i} \Rightarrow 1] - \Pr[\mathcal{D}^{G_0} \Rightarrow 1]| \leq \ell(\varepsilon_E + \varepsilon_{(p,2),\text{rsim}})$$

in total. \hfill $\square$

Then is the standard model version of Lemma 7. Unlike Lemma 7, there is no query to the ideal TBC here, and thus the notion $q_0c$ disappears in $\text{Adv}^{\text{eval2}}_{\mathsf{TStream}}(p, t, \ell)$.

**Lemma 12** (1-Block Advantage to $\ell$-Block). For every pair of $\ell$-block messages $m^0$ and $m^1$ and $(q_t, t)$-bounded adversary $\mathcal{A}^E$, there exists a $(q_t + 2q_r, t + 2t_r)$-bounded adversary $\mathcal{A}'$ such that

$$\text{Adv}^{\text{eval2}}_{\mathsf{TStream}}(p, t, \ell) := |\Pr[\mathcal{A}^E(\mathsf{ISStream}_{k_0}(T, N, m^0)) \Rightarrow 1] - \Pr[\mathcal{A}'(\mathsf{ISStream}_{k_0}(T, N, m^1)) \Rightarrow 1]|$$

$$\leq \sum_{i=1}^\ell |\Pr[\mathcal{A}'(\mathsf{LISEnc}^+_{k_{i-1}}(m^0_i)) \Rightarrow 1] - \Pr[\mathcal{A}'(\mathsf{LISEnc}^+_{k_{i-1}}(m^1_i)) \Rightarrow 1]|,$$

where $k_0, \ldots, k_{\ell-1}$ are chosen uniformly at random, and $m^0_i$ and $m^1_i$ are the $i$-th block of $m^0$ and $m^1$ respectively. Here $q_t = \ell(2q_S + 1)$ and $t_r = \ell(2t_S + 2t_S + t_{\oplus})$, where $t_E$, $t_S$, and $t_{\oplus}$ are as assumed in Lemma 11.

**Proof.** The proof follows the same line as Lemma 7. \hfill $\square$

Gathering Lemmas 10, 11, and following the same line as Lemma 8 we obtain

**Lemma 13.** Let $E : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation has a leakage function $L_E$ having $(q_S, t_S, q_t, t, \varepsilon_{(p,2),\text{rsim}})$ $(p, 2)$-R-simulatable leakages, and let $\mathcal{S}^L$ be an appropriate $(q_S, t_S)$-bounded leakage simulator. Then, for every pair of $\ell$-block messages $m^0$ and $m^1$ and $(q_t - 2q_r - q^*, t - 2t_r - t^*)$-bounded adversary $\mathcal{A}',$ it holds

$$|\Pr[\mathcal{A}'(\mathsf{TStream}_{k_0}(T, N, m^0)) \Rightarrow 1] - \Pr[\mathcal{A}'(\mathsf{TStream}_{k_0}(T, N, m^1)) \Rightarrow 1]|$$

$$\leq \delta(\varepsilon_E + \varepsilon_{(p,2),\text{rsim}}) + \ell \cdot \varepsilon_{\text{block}},$$

where $q_r, t_r$ are as defined in Lemma 11, and $q^*, t^*$ are as defined in Lemma 10.
Completing the muCCAmL2 Proof. We start by defining $G_0$ as the game $\text{PrivK}^{\mu\text{CCAmL2,0}}$, and $G_0^*$ as the game $\text{PrivK}^{\mu\text{CCAmL2,1}}$. From $G_0$ we obtain $G_1$ by replacing the $u$ instances of $E_{K_1}, \ldots, E_{K_u}$ invoked during the execution by $u$ independent tweakable random permutations $\tilde{P}_1, \ldots, \tilde{P}_u$. For each such $K_i$, it’s a quite standard trick to build a $(2q_e + 2q_d + 2q_m, t')$-bounded adversary $B_{\text{STPRP}}$ against the STPRP security of $E^*$, with $t' = q_e + q_d + q_m + t_{\text{pass}}$. The number of such $K_i$ involved in the execution is $\leq q_e + q_d + q_m$, thus

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq (q_e + q_d + q_m)\varepsilon_E.$$  

follows from the assumption that $E^*$ is a $(2q_e + 2q_d + 2q_m, t', \varepsilon_E)$-secure STPRP. Similarly, we replace $E_{K_1}, \ldots, E_{K_u}$ by $\tilde{P}_1, \ldots, \tilde{P}_u$ to turn $G_0^*$ into $G_1^*$, which also introduces a gap of $(q_e + q_d + q_m)\varepsilon_E$.

Then, following the proof of Eq. (27), the gap between $G_1$ and $G_1^*$ can be bounded:

$$|\Pr[(A^4)^{G_1} \Rightarrow 1] - \Pr[(A^4)^{G_1^*} \Rightarrow 1]| \leq \text{Adv}_{\text{muCCAmL2}}^{\text{TEDT}}(q_e + q_d + q_m, t') + \frac{q_e + q_m - 1}{2^n} + \sum_{i=1}^{\sigma_m} \text{Adv}_{\text{STPRP}}^{\text{E}}([p, t, \ell]),$$

with $\ell_i$ the number of blocks in the $i$th challenge message. This plus the above gap $2(q_e + q_d + q_m)\varepsilon_E$ yield the final claim.

D Black-box CCA Security of TEDT

In this section, we prove asymptotically optimal CCA security (without leakage) bounds for TEDT. We first present the security model in subsection D.1. The result and proof are later given in subsection D.2.

D.1 CCA Security with Misuse-Resilience

Ashur et al. [ADL17] proposed a strong indistinguishability notion for authenticated encryption which divides adversarial encryption queries into challenge and non-challenge ones, and only requires the adversary to be nonce-respecting among the former type of queries. The nonce-misuse in non-challenge queries should not affect the pseudorandomness of the responses to the challenge queries, i.e. of the challenge ciphertexts. To avoid confusion with misuse-resistance [RS06] we will not refer to misuse-resilience with its initials but as CCA$\mathcal{M}$ since it is a “real-or-random” indistinguishability game between the real world $(\text{Enc}_K, \text{Enc}_i, \text{Dec}_K)$ and the random (or ideal) world $(\text{Enc}_K, \mathcal{S}, \downarrow)$, hence the $\mathcal{S}$, where the second oracle is the challenge oracle. Formally, given a nonce-based authenticated encryption $\text{AEAD} = (\text{Enc}, \text{Dec})$, the multi-user chosen ciphertext misuse resilience advantage of an adversary $\mathcal{A}$ against $\text{AEAD}$ with $u$ users is

$$\text{Adv}_{\text{AEAD,AEAD}}^{\text{muCCAmS}} \overset{\text{def}}{=} |\Pr[A^{\text{Enc}, \text{Enc}_i, \text{Dec}_K, \mathcal{IC}, \mathcal{IC}^{-1} \Rightarrow 1}] - \Pr[A^{\text{Enc}_i, \mathcal{S}, \downarrow, \mathcal{IC}, \mathcal{IC}^{-1} \Rightarrow 1}]|,$$

where the probability is taken over the $u$ user keys $K = (K_1, \ldots, K_u)$, with $K_i \leftarrow \mathcal{K}$, over $\mathcal{A}$’s random tape and the ideal TBC IC and where $\text{Enc}_K(i, N, A, M)$: if $1 \leq i \leq u$, outputs $\text{Enc}_{K_i}(N, A, M)$; $\mathcal{S}(i, N, A, M)$ outputs and associates a fresh random ciphertext $C \leftarrow C_{[M]}$ to fresh input; $\text{Dec}(i, N, A, C)$ outputs $\text{Dec}_{K_i}(N, A, C)$; $\downarrow(i, N, A, C)$ outputs $\downarrow$; for each user $i$: (i) nonce $N$ cannot be used both in query to $O_1(i, N, *, *)$ and $O_2(i, N, *, *)$; (ii) $O_2(i, *, *, *)$ is nonce-respecting; (iii) if $C$ is returned by $O_1(i, N, A, M)$ or $O_2(i, N, A, M)$ query $O_3(i, N, A, C)$ is forbidden; (iv) a nonce used twice with $O_1$ cannot be used for an
$O_3$ query. The extended muCCAm$^*$ advantage muCCAm$^*$ is defined as the muCCAm$^*$ one where the last restriction (iv) is waived from the definition. We stress that all these conditions were already introduced in [ADL17], mainly to enable standard to fall into misuse-resilient AEAD.

### D.2 muCCAm$^*$ Security of TEDT

To simplify the notations, we define

$$\text{Adv}_{\text{TBC}, \mu\text{CCAm}^*}^{\text{muCCAm}^*}(u, \sigma, q) \overset{\text{def}}{=} \max \left\{ \text{Adv}_{\text{TBC}, \mu\text{CCAm}^*}^{\text{TBC}}(u, \sigma) \right\},$$

where $q = (q_{m}, q_{c}, q_{d}, q_{t})$, and the maximum is taken over all $(q, t)$-bounded adversaries against $u$ users that have at most $\sigma$ blocks in all their queried plaintext (both challenge and non-challenge) and ciphertext including AD.

**Theorem 4.** When the $u$ public-keys $T_1, \ldots, T_u$ are uniformly distributed, $n \geq 6$, and $2\sigma + 3(q_{c} + q_{d}) + q_{t} \leq 2n/8$, then the following holds in the ideal TBC model:

$$\text{Adv}_{\text{TBC}, \mu\text{CCAm}^*}^{\text{muCCAm}^*}(u, \sigma, q) \leq 2u^2 \frac{2^u}{2^{2n}} + \frac{1}{n!} \left( \frac{4u}{2^n} \right)^n + \left( \frac{7n^2 + 26}{2^n} \right)^n \frac{(4\sigma + 6(q_{c} + q_{d}) + 2q_{t})}{n^2}.$$

Partly thanks to the GCM-like counters the muCCAm$^*$ bound is asymptotically optimal $O\left(\frac{u^2}{2^{2n}} + \frac{\sigma + n^2 q_{t}}{2^n}\right)$. Moreover, the terms $\frac{2u^2}{2^{2n}}$ and $\frac{1}{n!} \left( \frac{4u}{2^n} \right)^n$ disappear if the $u$ public-keys are distinct. While the bound seems not to depend on $q_{m}$, the non-challenge queries actually affect $\sigma$ which, in turn, affects the bound. The bound of GCM-SIV with KDF is at best $\frac{\sigma \ell_{max} + q_{t}}{2^n}$, which we surpass when the maximal query length $\ell_{max}$ exceeds $n^2$: see Appendix D.3 for the details.

In the remaining of this subsection we prove Theorem 4. We rely on a balls-in-bin lemma from [PS15] presented as follows.

**Lemma 14 (Balls-in-Bin).** Consider throwing a ball into a bin that is chosen independently uniformly at random from $2^n \geq 8$ bins. Then the probability that, after throwing $\sigma$ balls with $8 \leq \sigma \leq 2^n$, any bin contains $2\log_2 \sigma$ balls or more, is less than $\frac{1}{2^n}$.

**Proof.** See [PS15, Appendix A]. That proof covered more general cases which we don’t rely on. \(\square\)

Then, note that in the misuse resilience setting, schemes which achieve both CPA confidentiality and authenticity also achieve CCA confidentiality [ADL17]:

$$\text{Adv}_{A, \text{AEAD}, u}^{\text{muCCAm}^*} = \left| \Pr \left[ A^{\text{Enc}_{K,T}, \text{Enc}_{K,T}, \text{Dec}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] - \Pr \left[ A^{\text{Enc}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] \right| \leq \left| \Pr \left[ A^{\text{Enc}_{K,T}, \text{Enc}_{K,T}, \text{Dec}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] - \Pr \left[ A^{\text{Enc}_{K,T}, \text{Enc}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] \right|$$

$$+ \left| \Pr \left[ A^{\text{Enc}_{K,T}, \text{Enc}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] - \Pr \left[ A^{\text{Enc}_{K,T}, \perp, \perp, I, C, \perp, I, C \leftarrow 1} \right] \right|. \quad (31)$$

Clearly, $\text{Adv}_{A, \text{TBC}, u}^{\text{muINT-CTXT}} \leq \text{Adv}_{A, \text{TBC}, u}^{\text{muCCAm}^*}$. Therefore, we focus on the CPA advantage $\text{Adv}_{A, \text{TBC}, u}^{\text{muCCAm}^*}$ switching to the CPA setting greatly simplifying the setting as well as the notations. Again we employ the H-coefficient technique, and present the two steps in two subsections.
D.2.1 Bad Transcripts

Following section 5.3, during the interaction, we also reveal some underlying IC queries to \(D\) and include them in the transcript. In detail,

- First, we reveal all the IC queries underlying the non-challenge encryption queries (i.e., queries to the first encryption oracle);
- Second, for the challenge encryption queries, we emulate the corresponding hash evaluations, and reveal all the induced IC queries.

We merge these queries with the adversarial queries to obtain a set \(\tau^*_{IC}\). Thus we have \(|\tau^*_{IC}| \leq q^*_c\) which is as defined by Eq. (7). Note that this upper bound is a bit coarse, but it’s enough for the remaining argument.

We also organize the hash query transcript \(\tau^*_{H}\). Besides, the set

\[
\tau = (\tau^*_{H}, \tau_e, \tau^*_{IC}, T, K).
\]

For an encryption query \((u^{(i)}, N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)})\), we denote

\[
M^{(i)} = m_1^{(i)} || \ldots || m_{\ell_i}^{(i)},
\]

and

\[
C^{(i)} = c_1^{(i)} || Z^{(i)} = c_1^{(i)} || \ldots || c_{\ell_i}^{(i)} || Z^{(i)},
\]

i.e., \(m_j^{(i)}\), resp. \(c_j^{(i)}\), denotes the \(j\) th \(n\)-bit block of \(M^{(i)}\), resp. \(C^{(i)}\). Wlog assume that \(|m_j^{(i)}| = n\) for any block. And we define an auxiliary quantity

\[
\mu_Y := \max_{y \in \{0,1\}^n} |\{(i, j) : m_j^{(i)} \oplus c_j^{(i)} = y\}|.
\]

Note that in the ideal world, all the blocks in \(C^{(1)}, \ldots , C^{(q^*_c)}\) are uniformly distributed in \(\{0,1\}^n\). Therefore,

\[
\Pr[T_{id} = \tau] = \Pr(K, T) \cdot \Pr[IC \vdash \tau^*_{IC}] \cdot \left(1 - \frac{1}{2^n}\right)^{q^*_c + \sum_{i=1}^{q^*_c} \ell_i}.
\]  

(32)

With the above, a transcript \(\tau\) is bad if one of the following is fulfilled:

- (B-1) there exists two user indices \(i, j\) such that \(K_i || T_i = K_j || T_j\);
- (B-2) \(\mu_T \geq n\), or \(\mu_W \geq n\);
- (B-3) there exists an encryption query \((u, N, A, M, C), C = c || Z\), such that \((K_u, T_u, x, y) \in \tau^*_{IC}\) or \((K_u, W || 1, x, y) \in \tau^*_{IC}\) for some \(x, y\), where \(W\) comes from its corresponding hash record \(\text{pad}(A, N, c, T_u), V || W) \in \tau^*_{H}\);
- (B-4) \(\mu_Y \geq 2 \log_2 \sigma\);
- (B-5) there exists distinct queries \((u^{(i)}, N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)})\) and \((u^{(j)}, N^{(j)}, A^{(j)}, M^{(j)}, C^{(j)})\) with the corresponding hash records \((U^{(i)}, V || W^{(i)})\) and \((U^{(j)}, V || W^{(j)})\) in \(\tau^*_{H}\) satisfying \((U^{(i)} = \text{pad}(A^{(i)}, N^{(i)}, c^{(i)}, T_u^{(i)}), U^{(j)} = \text{pad}(A^{(j)}, N^{(j)}, c^{(j)}, T_u^{(j)}))\).
We remark that the event of nonce-reusing for a single user, i.e., there exists two queries of the form

\[ W^{(i)} = W^{(j)} \] or

contradiction: \[ W^{(i)} = W^{(j)} \] and \[ Z^{(i)} = Z^{(j)} \].

The first three conditions have been analyzed in subsection 5.3.2. We recycle the results

\[ \Pr[(B-1)] \leq \frac{u^2}{2^n} , \quad \Pr[(B-3) \mid (B-2)] \leq \frac{n^2 q_\mathcal{C}^*}{2^n} , \]

\[ \Pr[(B-2) \lor \text{hash collision}] \leq \frac{1}{2n!} \cdot \left( \frac{4u}{2^n} \right)^n \! + \! \frac{8q_\mathcal{C}^*}{2^n} , \]

where \( q_\mathcal{C}^* \) is defined by Eq. (7).

For (B-4), in the ideal world \( \epsilon_j^{(i)} \) and thus \( m_j^{(i)} \oplus \epsilon_j^{(i)} \) is uniformly and independent from anything else. In addition \( |\{(i, j)\}| \leq \sigma_2 \leq \sigma \ll 2^n \) with \( \sigma_2 \) denoting the number of blocks in queries to the challenge encryption oracle. So Lemma 14 yields

\[ \Pr[(B-4)] \leq \Pr[\mu_Y \geq 2 \log_2 \sigma] \leq \frac{1}{2^n} . \]

For (B-5), the collision event has been included in the above bounds. On the other hand, for any two indices \( i, j \), it can be seen from the proof in Section 5.2 that \( \Pr[W^{(i)} = W^{(j)}] \leq \frac{2}{2^n - q_\mathcal{C}^*} \). On the other hand, the tags \( Z^{(i)} \) and \( Z^{(j)} \) are uniform in the ideal world, and thus \( \Pr[Z^{(i)} = Z^{(j)}] = \frac{1}{2^n} \). Since we have \( \leq q_\mathcal{C}^*/2 \) such pairs of indices \((i, j)\), it holds

\[ \Pr[\text{contradiction in (B-5)}] \leq \frac{q_\mathcal{C}^*}{2n(2^n - q_\mathcal{C}^*)} \leq \frac{2q_\mathcal{C}^*}{2^n} \leq \frac{q_\mathcal{C}^*}{2^n}. \]

In all,

\[ \Pr[T_{id} \in \mathcal{T}_{bad}] \leq \frac{u^2}{2^{2n}} + \frac{1}{2n!} \cdot \left( \frac{4u}{2^n} \right)^n \! + \! \frac{(n^2 + 9)q_\mathcal{C}^* + 1}{2^n} . \quad (33) \]

### D.2.2 Ratio of Probabilities of Good Transcripts

For a good transcript \( \tau \), by \( \neg(B-3) \), for any \((u^{(i)}, N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)}) \in \tau_e \) the initial session key \( k_0^{(i)} = IC_{K_{K_u^{(i)}}}^{K_{\mathcal{T}_{k_0^{(i)}}}}(P_0(N^{(i)})) \) is uniform. With this observation, we define the first predicate \text{BadKD}(\mathcal{IC}) \) to capture the “badness” of this key. Formally, \text{BadKD}(\mathcal{IC}) \) is fulfilled if one of the following conditions is fulfilled:

- (C-1) “none-freshness” of the key: there exists \((u^{(i)}, N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)}) \in \tau_e \) such that the key \( k_0^{(i)} = IC_{K_{K_u^{(i)}}}^{K_{\mathcal{T}_{k_0^{(i)}}}}(P_0(N^{(i)})) \) satisfies \((k_0^{(i)}, T_{u^{(i)}}, P_1(N^{(i)}), y) \in \tau_C^* \), or \((k_0^{(i)}, T_{u^{(i)}}, Q_0(N^{(i)}), y) \in \tau_C^* \) for some \( y \), or \((k_0^{(i)}, T_{u^{(i)}}, x, y_1^{(i)}) \in \tau_C^* \) for some \( x \);

- (C-2) nonce-reuse across different users: there exists two queries \((u^{(i)}, N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)}) \) and \((u^{(j)}, N^{(j)}, A^{(j)}, M^{(j)}, C^{(j)}) \) in \( \tau_e \), such that \( T_{u^{(i)}} = T_{u^{(j)}} = T, N^{(i)} = N^{(j)} = N, \)
and the two keys \( k_0^{(i)} = IC_{K_{K_u^{(i)}}}^{K_{\mathcal{T}_{k_0^{(i)}}}}(P_0(N^{(i)})) \) and \( k_0^{(j)} = IC_{K_{K_u^{(j)}}}^{K_{\mathcal{T}_{k_0^{(j)}}}}(P_0(N^{(j)})) \) are identical.

We remark that the event of nonce-reusing for a single user, i.e., there exists two queries of the form \((u, N, A, M, C), (u, N, A', M', C') \) in \( \tau_e \), is also bad for the subsequent probability calculation. But this is forbidden by the muCCAM$^\$ security definition.

For a specific public-key \( T \), a nonce \( N \), and an index \( j \), we define an auxiliary set of keys

\[ \tau_C^*[T, j, N] := \left\{ k : (k, T, P_{j+1}(N), y) \in \tau_C^* \text{ or } (k, T, Q_j(N), y) \in \tau_C^* \text{ for some } y \right\} . \]
In addition, for $T$ and a key stream block $y \in \{0,1\}^n$, define

$$\tau^*_C[T,y]^{-1} := \{k : (k,T,x,y) \in \tau^*_C \text{ for some } x\}.$$ 

Conditioned on $IC \vdash \tau^*_C$ and the values of

$$k_0^{(i)} = IC_{K^{(i)}}(N^{(i)}), \ldots, k_{n-1}^{(i)} = IC_{K^{(i-1)}}(N^{(i-1)}),$$

the key $k_0^{(i)} = IC_{K^{(i)}}(N^{(i)})$ is uniform in $\geq 2^n - q_C - q_e$ possibilities: the first half is due to $\neg(B-3)$, while the second half is due to the non-repeating property of the triple $(K,T,N)$ (for the same user $N$ can’t be repeated, while for different users $i,j$ we’ve $K_i || T_i \neq K_j || T_j$ by $\neg(B-1)$). Therefore,

$$\Pr[(C-1)] \leq \sum_{i=1}^{q_e} \frac{\tau^*_C[T_{u,i},0,N^{(i)}]}{2^n - q_C - q_e} + \sum_{i=1}^{q_e} \frac{\tau^*_C[T_{u,i},y^{(i)}]^{-1}}{2^n - q_C - q_e}.$$ 

For (C-2), the number of choice for $(u^{(i)},N^{(i)},A^{(i)},M^{(i)},C^{(i)})$ is $\leq q_e$. For each such choice, the number of choice for the $(u^{(j)},N^{(j)},A^{(j)},M^{(j)},C^{(j)})$ is $\leq \mu_T$ due to the restriction $T_{u^{(i)}} = T_{u^{(j)}}$. Thus

$$\Pr[(C-2)] \leq \frac{\mu_T q_e}{2^n - q_C - q_e} \leq \frac{2\mu_T q_e}{2^n}.$$ 

Thus when $q_C + q_e \leq 2^n/2$, we have

$$\Pr_{IC}[BadKD(IC) \mid IC \vdash \tau^*_C] \leq \frac{2\mu_T q_e}{2^n} + \sum_{i=1}^{q_e} \frac{2\tau^*_C[T_{u,i},0,N^{(i)}]}{2^n} + \sum_{i=1}^{q_e} \frac{2\tau^*_C[T_{u,i},y^{(i)}]^{-1}}{2^n}.$$ 

(34)

We then analyze the $q_e$ encryption queries in turn, and define a sequence of bad predicates

$$BadE^{(i)}_1, BadE^{(i)}_2, \ldots, BadE^{(i)}_{\ell_i-1},$$

$$\ldots$$

$$BadE^{(q_e)}_1, BadE^{(q_e)}_2, \ldots, BadE^{(q_e)}_{q_{e-1}}.$$ 

(35)

As will be seen, each predicate concerns with the encryption of a specific plaintext block. Formally, for $1 \leq i \leq q_e$, consider the $i$-th query $(u^{(i)},N^{(i)},A^{(i)},M^{(i)},C^{(i)})$, and for $1 \leq j \leq \ell_i - 1$, let

$$k_0^{(i)} = IC_{K^{(i)}}(P_0(N^{(i)})), k_1^{(i)} = IC_{K^{(i)}}(P_1(N^{(i)})), \ldots, k_j^{(i)} = IC_{K^{(i)}}(P_j(N^{(i)}))$$

be the derived intermediate values. Then $BadE^{(i)}_j(IC)$ is fulfilled, if at least one of the following conditions is fulfilled:

- (C-ij1): $k_j^{(i)} \in \tau^*_C[T_{u,(i)}, j, N^{(i)}], \text{ or}$
- (C-ij2): there exists $s < i$ such that for the $s$-th encryption query $(u^{(s)},N^{(s)},A^{(s)},M^{(s)},C^{(s)})$, it holds:
  
  - $T_{u^{(s)}} = T_{u^{(i)}}$ and $N^{(s)} = N^{(i)}$, and
  
  - $k_j^{(i)}$ equals the intermediate value $k_j^{(s)}$ derived correspondingly.
(C-ij3): \( k_j^{(i)} \in \tau_c^\ast \left[ T_{u^{(i)}}, y_{j+1}^{(i)} \right]^{-1} \), or

(C-ij4): there exists two indices \((s, t)\) such that either \(s < i\), or \(s = i\) and \(t < j\), and:

\[
\begin{align*}
- y_{t+1}^{(s)} &= y_j^{(i)}, \\
- k_j^{(i)} &= \text{the intermediate value } k_j^{(s)} \text{ derived correspondingly.}
\end{align*}
\]

(C-ij5): \( k_j^{(i)} = y_j^{(i)} = m_j^{(i)} \circ e_j^{(i)} \).

It isn’t hard to see conditioned on IC \( \vdash \tau_c^\ast \) and \( \neg \text{BadKD}(\text{IC}) \) and \( \neg \text{BadE}_{j-1}^{(i)}(\text{IC}) \land \ldots \land \neg \text{BadE}_1^{(i)}(\text{IC}) \), the value \( k_j^{(i)} \) is uniform in \( \geq 2^n - q_c^\ast - q_e \) possibilities. Therefore,

\[
\Pr[(\text{C-ij1}) \lor (\text{C-ij3})] \leq \frac{\tau_c^\ast \left[ T_{u^{(i)}}, j, N^{(i)} \right] + \left| \tau_c^\ast \left[ T_{u^{(i)}}, y_{j+1}^{(i)} \right]^{-1} \right|}{2^n - q_c^\ast - q_e}.
\]

For (C-ij2), due to the restriction \( T_{u^{(i)}} = T_{u^{(i)}} \) and the design of TEDT, it can be seen the number of such index \( s \) is at most \( \mu_T - 1 \). Similarly, for (C-ij4), the restriction \( y_{t+1}^{(s)} = y_j^{(i)} \) indicates the number of such pairs of indices \((s, t)\) is at most \( \mu_Y - 1 \). Therefore,

\[
\Pr[(\text{C-ij2}) \lor (\text{C-ij4}) \lor (\text{C-ij5})] \leq \frac{\mu_T - 1 + \mu_Y - 1 + 1}{2^n - q_c^\ast - q_e}.
\]

Thus when \( q_c^\ast + q_e \leq 2^n/2 \) we have

\[
\Pr[\text{BadE}_j^{(i)}(\text{IC}) \mid \neg \text{BadE}_{j-1}^{(i)}(\text{IC}) \land \ldots \land \neg \text{BadE}_1^{(i)}(\text{IC}) \land \neg \text{BadKD}(\text{IC}) \land \text{IC} \vdash \tau_c^\ast] \leq \frac{2^n}{2^n}.
\]

For \( 1 \leq i \leq q_e \) and \( 1 \leq j \leq \ell_i \), conditioned on \( \neg \text{BadE}_j^{(i)}(\text{IC}) \land \neg \text{BadE}_{j-1}^{(i)}(\text{IC}) \land \ldots \land \neg \text{BadE}_1^{(i)}(\text{IC}) \land \neg \text{BadKD}(\text{IC}) \land \text{IC} \vdash \tau_c^\ast \), it can be seen the value \( y^\dagger = \text{IC}^\ast \left[ T_{u^{(i)}}(Q_{j-1}^{(i)}(N^{(i)})) \right] \) is uniform in \( \geq 2^n - q_c^\ast - q_e \) possibilities, and these possibilities include \( y_j^{(i)} \). Therefore,

\[
\Pr[y^\dagger = y_j^{(i)}] \geq \frac{1}{2^n}.
\] (36)

The probabilities of the predicates cumulate to

\[
\Pr[\text{BadE}_{\ell_{q_e}-1}^{(q_e)}(\text{IC}) \lor \ldots \lor \text{BadE}_1^{(1)}(\text{IC}) \lor \neg \text{BadKD}(\text{IC}) \mid \text{IC} \vdash \tau_c^\ast] \leq \sum_{i=1}^{q_e} \sum_{j=1}^{\ell_i} \frac{\left( \tau_c^\ast \left[ T_{u^{(i)}}, j, N^{(i)} \right] + \left| \tau_c^\ast \left[ T_{u^{(i)}}, y_{j+1}^{(i)} \right]^{-1} \right| \right)}{2^n} + \mu_T + \mu_Y.
\]

This plus Eq. (34) yields

\[
\Pr[\text{Bad}(\text{IC}) \mid \text{IC} \vdash \tau_c^\ast] \leq \sum_{i=1}^{q_e} \sum_{j=0}^{\ell_i-1} \frac{\left( \tau_c^\ast \left[ T_{u^{(i)}}, j, N^{(i)} \right] + \left| \tau_c^\ast \left[ T_{u^{(i)}}, y_{j+1}^{(i)} \right]^{-1} \right| \right)}{2^n} + \mu_T + \mu_Y.
\]
It’s easy to see \( \sum_{i=1}^{q_e} \sum_{j=0}^{\ell_i-1} (\mu_T + \mu_Y) \leq (\mu_T + \mu_Y) \sigma \). On the other hand, for the summation \( \sum_{i=1}^{q_e} \sum_{j=0}^{\ell_i-1} \tau \mathbb{E} [T_{u(i)}, j, N(i)] \), we reorganize it according to different \( T \) values. In this vein, we have

\[
\sum_{i=1}^{q_e} \sum_{j=0}^{\ell_i-1} \tau \mathbb{E} [T_{u(i)}, j, N(i)] \leq \mu_T \cdot \sum_{\tau \in \{0, 1, \ldots, \ell_i-1\}} \tau \mathbb{E} [T, j, N] \leq \mu_T |\tau \mathbb{E} | \leq \mu_T q_\tau \mathbb{E}.
\]

Similarly,

\[
\sum_{i=1}^{q_e} \sum_{j=1}^{\ell_i} \tau \mathbb{E} [T_{u(i)}, q_j(i)]^{-1} \leq \mu_Y q_\tau \mathbb{E}.
\]

Gathering the above and Eq. (36) yields

\[
\Pr \left[ \text{TDE0} | \mathcal{IC}, \text{Enc}_{K, T}(u(i), N(i), A(i), M(i)) = c(i), \text{ for all } i \in \{1, \ldots, q_e\} \mid \mathcal{IC} \vdash \tau \mathbb{E} \right] \geq \Pr \left[ -\text{Bad}(\mathcal{IC}) \mid \mathcal{IC} \vdash \tau \mathbb{E} \right] \left( \frac{1}{2^n} \right)^{\sum_{i=1}^{q_e} \ell_i} \geq \left( 1 - \frac{2\mu_T \sigma + 2\mu_Y \sigma + 2\mu_Y q_\tau \mathbb{E} + 2\mu_Y q_\tau \mathbb{E}}{2^n} \right) \left( \frac{1}{2^n} \right)^{\sum_{i=1}^{q_e} \ell_i} \cdot.
\]

(37)

It remains to analyze the produced tags. Let the hash query record corresponding to \((u(i), N(i), A(i), M(i), C(i)) = (U(i), V(i)||W(i))\). Therefore, the event that the \( q_e \) tags equal \( Z(1), \ldots, Z(q_e) \) is equivalent to \( q_e \) equalities as follows:

\[
\mathcal{IC}^{|W(i)}_K (V(1)) = Z(1), \ldots, \mathcal{IC}^{|W(i)}_K (V(q_e)) = Z(q_e).
\]

Consider the first equality. The entry \( \mathcal{IC}^{|W(i)}_K (V(1)) \) may have been rendered non-random due to the condition \( \text{TDE0} | \mathcal{IC}, \text{Enc}_{K, T}(u(i), N(i), A(i), M(i)) = c(i), \text{ for all } i \in \{1, \ldots, q_e\} \) or due to \( \mathcal{IC} \vdash \tau \mathbb{E} \). However, the former condition only affects entries with the tweak \( T, 0 \), while the latter would not affect \( \mathcal{IC}^{|W(i)}_K (V(1)) \) due to \( \neg(B-3) \). Therefore, \( \Pr[\mathcal{IC}^{|W(i)}_K (V(1)) = Z(1)] = \frac{1}{2^n} \).

In a similar vein, for any index \( i \in \{1, \ldots, q_e\} \), under the conditions that \( \mathcal{IC} \vdash \tau \mathbb{E} \) and \( \neg(B-3) \), “\( \neg(B-5) \)”, “\( V(i)|W(i) = Z(i) \)”, “\( Z(i)|W(i) \neq Z(i)|W(i) \)”, and “\( Z(i)|W(i) \neq Z(i)|W(i) \)” for any \( j < i \). By \( \neg(B-5) \), \( V(i)|W(i) \neq Z(i)|W(i) \) and \( Z(i)|W(i) \neq Z(i)|W(i) \). By this, \( \Pr[\mathcal{IC}^{|W(i)}_K (V(i)) = Z(i)] \leq \frac{1}{2^n} \), and thus

\[
\Pr[\mathcal{IC}^{|W(i)}_K (V(i)) = Z(i)] = 1 - \frac{1}{2^n \cdot q_e}.
\]

Gathering Eq. (32), (37), and (38), and with \( \sigma \leq 2^n/48 \Rightarrow \log_2 \sigma \leq n \), we have

\[
\frac{\Pr[T_{T_e} = |T|]}{\Pr[T_{T_id} = |T|]} \geq \left( 1 - \frac{2\mu_Y q_\tau \mathbb{E} + 2\mu_Y q_\tau \mathbb{E}}{2^n} \right) \left( \frac{1}{2^n} \right)^{q_e + \sum_{i=1}^{q_e} \ell_i} \left( \frac{1}{2^n} \right)^{q_e + \sum_{i=1}^{q_e} \ell_i} \geq 1 - \frac{2n(\sigma + q_\tau \mathbb{E}) + 4nq_\tau \mathbb{E}}{2^n}, \quad (\mu_T \leq n, \mu_Y \leq 2 \log_2 \sigma \leq 2n)
\]

\[
\geq 1 - \frac{6n(q_\tau \mathbb{E} + q_\tau \mathbb{E})}{2^n} \geq 1 - \frac{8nq_\tau \mathbb{E}}{2^n}, \quad (\sigma \leq \frac{q_\tau \mathbb{E}}{4})
\]

(39)
This plus Eq. (33) yield

$$\text{Adv}_{D,TEDT,u}^{\text{muCPAM}^*} \leq \frac{u^2}{2^{2n}} + \frac{1}{2n!} \cdot \left(\frac{4u}{2^n}\right)^n + \frac{(n^2 + 8n + 9)q_c + 1}{2^n}.$$ 

By Eq. (31), this plus the bound in Eq. (1) yields

$$\leq \frac{2u^2}{2^{2n}} + \frac{1}{n!} \cdot \left(\frac{4u}{2^n}\right)^n + \frac{1}{2n!} \cdot \left(\frac{4u}{2^n}\right)^n + \frac{(7n^2 + 26)(4\sigma + 6(q_e + q_d) + 2q_c)}{2^n} \quad (n \geq 2).$$

### D.3 Detailed Comparison with GCM-SIV with KDF

GCM-SIV with nonce-based KDF was recently proposed by Gueron and Lindell [GL17] as a very elegant and efficient BBB secure variant of GCM-SIV. Its security bounds are further improved by Bose et al. [BHT18] in the ideal cipher model.

In summary, AES-128-GCM-SIV with KDF uses $n = 128$ key bits, and its advantages are:

(i) Misuse-resistance (in black-box setting)

(ii) Asymptotically optimal mu security when nonce is random

(iii) Much less calls to the block cipher

For $n = 128$ TEDT uses 255 key bits but only 128 bits are secret. Its advantages are:

(i) Asymptotically optimal mu security for fresh nonce (misuse-resilience), say allowing arbitrary nonce reuse across different users, and slightly better bounds than AES-128-GCM-SIV (see below)

(ii) Provable side-channel security.

Concretely, in the mu setting, if nonce is arbitrarily reused across many users, then security of AES-128-GCM-SIV is capped at $u^2$. While Bose et al. proved that using random nonce could forbid arbitrarily reuse and achieve mu BBB, this method turns ineffective when the nonces in use are the same in each session (e.g., TLS 1.3 uses the sequence number to compute the nonces). Also according to Rogaway random nonce may be more prone to misuse. In comparison, in TEDT we employ additional 127-bit public randomness to achieve mu BBB. Though don’t support full nonce misuse, TEDT does support arbitrarily reusing the same nonce across different users.

In terms of bounds, with 128-bit secret keys, the mu bound of AES-128-GCM-SIV is

$$\frac{\sigma \ell_{\text{max}} + d(q_c + \sigma)}{2^{128}},$$

where $\ell_{\text{max}}$ and $d$ are upper bounds, respectively, the number of blocks encrypted per user-nonce pair, and of the number of users that re-use a particular nonce value. Setting $d = 1$ recovers its su bound. By this, TEDT probably achieves slightly better bounds $\frac{2^{128} \sigma + 2^{128} q_c}{2^{128}}$ even in the su setting (revisiting the calculations in Eq. (9) shows that the factor $n^2$ could be improved to $n$ in the su setting, so its su bound could be better $\frac{128 \sigma + 128 q_c}{2^{128}}$).

As mentioned, we believe this shows the benefits of protocol-level leakage-resilient designs.
E A single-pass TBC Mode TET

The single-pass mode TET is defined in Figure 14 and illustrated in Figure 15. It achieves $\mu_C^\text{IML2}$ with nearly optimal bounds as TEDT. Yet, since it has only one pass, the consensus is that the confidentiality is unavoidably altered given decryption leakages. Consequently, TET only ensures $\mu_C^\text{CAmL1}$ security, with bounds comparable to Theorem 2.
algorithm TET[E].Enc\(\mathcal{K}\)(N, A, M)

1. \(\ell \leftarrow \lceil |M|/n \rceil\)
2. \(\nu \leftarrow \lceil |A|/n \rceil\)
3. parse \(M\) as \(M[1] \ldots M[\ell]\), with \(|M[1]| = \ldots = |M[\ell - 1]| = n\) and \(1 \leq |M[\ell]| \leq n\)
4. \(M[\ell] \leftarrow M[\ell]0^{\ast}\)
5. parse \(A\) as \(A[1] \ldots A[\nu]\), with \(|A[1]| = \ldots = |A[\nu - 1]| = n\) and \(1 \leq |A[\nu]| \leq n\)
6. \(A[\nu] \leftarrow A[\nu]0^{\ast}\)
7. \(T \leftarrow PK0\)
8. \(S \leftarrow E^T_{\mathcal{K}}(N0^{\ast})\)
9. \(a \leftarrow E^PK10(N0^{\ast})\)
10. \(b \leftarrow E^PK11(N0^{\ast})\)
11. \(h_1 \leftarrow b_{n-1}(a \oplus b)\)
12. \(k_1 \leftarrow b \oplus (N0^{\ast})\)
13. for \(i = 1\) to \(\ell\) do
14. \(C[i] \leftarrow E^{h_i}_{k_i0}(M[i])\)
15. \(c_i^* \leftarrow E^{h_i}_{k_i0}(M[i])\)
16. \(h_{i+1} \leftarrow b_{n-1}(C[i] \oplus c_i^*)\)
17. \(k_{i+1} \leftarrow c_i^* \oplus M[i]\)
18. for \(i = 1\) to \(\nu\) do
19. \(j \leftarrow i + \ell\)
20. \(d_i \leftarrow E^{h_i}_{k_i0}(A[i])\)
21. \(d_i^* \leftarrow E^{h_i}_{k_i0}(A[i])\)
22. \(h_{j+1} \leftarrow b_{n-1}(d_j \oplus d_j^*)\)
23. \(k_{j+1} \leftarrow d_j^* \oplus [A[i]]\)
24. \(j \leftarrow \ell + \nu + 1\)
25. \(tail \leftarrow \lceil |M|/n/2 \rceil ||A||_{n/2}\)
26. \(d \leftarrow E^{h_i}_{k_i0}(tail)\)
27. \(d^* \leftarrow E^{h_i}_{k_i0}(tail)\)
28. \(V \leftarrow b_{n-1}(d \oplus d^*)\)
29. \(U \leftarrow d^* \oplus tail\)
30. \(Z \leftarrow E^{V}_{\mathcal{K}}(U)\)
31. \(C \leftarrow C[1] \ldots C[|M|]Z\)
32. return \(C\)

algorithm TET[E].Dec\(\mathcal{K}\)(N, A, C)

1. \(\ell \leftarrow \lceil |C|/n \rceil\)
2. \(\nu \leftarrow \lceil |A|/n \rceil\)
3. parse \(C\) as \(C[1] \ldots C[\ell]\), with \(|C[1]| = \ldots = |C[\ell - 1]| = |Z| = n\) and \(1 \leq |C[\ell]| \leq n\)
4. parse \(A\) as \(A[1] \ldots A[\nu]\), with \(|A[1]| = \ldots = |A[\nu - 1]| = n\) and \(1 \leq |A[\nu]| \leq n\)
5. \(A[\nu] \leftarrow A[\nu]0^{\ast}\)
6. \(T \leftarrow PK0\)
7. \(S \leftarrow E^T_{\mathcal{K}}(N0^{\ast})\)
8. \(a \leftarrow E^{PK10}(N0^{\ast})\)
9. \(b \leftarrow E^{PK11}(N0^{\ast})\)
10. \(h_1 \leftarrow b_{n-1}(a \oplus b)\)
11. \(k_1 \leftarrow b \oplus (N0^{\ast})\)
12. for \(i = 1\) to \(\ell\) do
13. \(M[i] \leftarrow (E^{h_i}_{k_i10})^{-1}(C[i])\)
14. \(c_i^* \leftarrow E^{h_i}_{k_i10}(M[i])\)
15. \(h_{i+1} \leftarrow b_{n-1}(C[i] \oplus c_i^*)\)
16. \(k_{i+1} \leftarrow c_i^* \oplus M[i]\)
17. for \(i = 1\) to \(\nu\) do
18. \(j \leftarrow i + \ell\)
19. \(d_i \leftarrow E^{h_i}_{k_i10}(A[i])\)
20. \(d_i^* \leftarrow E^{h_i}_{k_i10}(A[i])\)
21. \(h_{j+1} \leftarrow b_{n-1}(d_j \oplus d_j^*)\)
22. \(k_{j+1} \leftarrow d_j^* \oplus [A[i]]\)
23. \(j \leftarrow \ell + \nu + 1\)
24. \(tail \leftarrow \lceil |M|/n/2 \rceil ||A||_{n/2}\)
25. \(d \leftarrow E^{h_i}_{k_i10}(tail)\)
26. \(d^* \leftarrow E^{h_i}_{k_i10}(tail)\)
27. \(V \leftarrow b_{n-1}(d \oplus d^*)\)
28. \(U \leftarrow d^* \oplus tail\)
29. \(Z \leftarrow (E^V_{\mathcal{K}})^{-1}(U)\)
30. if \(U^* \neq U\) then return 1
31. \(M \leftarrow M[1] \ldots M[\ell]\)
32. return \(M\)

Figure 14: Definition of the TET mode, using a TBC E.
Figure 15: Illustrating the TET AEAD. The two dark blocks are “leak-free” TBC-calls. The other TBC-calls are leaking. For each square, the input to the triangle denotes the key input. The “tail” block is $tail = ||M||_{n/2}||A||_{n/2}$, and $T = PK0$. 