On the Security of Triplex- and Multiplex-type Constructions with Smaller Tweaks

Nilanjan Datta[1], Avijit Dutta[1], Eik List[2] and Sougata Mandal[1,3]

1 Institute for Advancing Intelligence, TCG CREST, India
2 Nanyang Technological University, Singapore
3 Ramakrishna Mission Vivekananda Educational and Research Institute, India
nilanjan.datta@tcgcrest.org, avijit.dutta@tcgcrest.org,
eik.list@ntu.edu.sg, sougata.mandal@tcgcrest.com

Abstract. In TCHES'22, Shen et al. proposed Triplex, a single-pass leakage-resistant authenticated encryption scheme based on Tweakable Block Ciphers (TBCs) with $2n$-bit tweaks. Triplex enjoys beyond-birthday-bound ciphertext integrity in the CIML2 setting and birthday-bound confidentiality in the CCAmL1 notion. Despite its strengths, Triplex's operational efficiency was hindered by its sequential nature, coupled with a rate limit of 2/3. In an endeavor to surmount these efficiency challenges, Peters et al. proposed Multiplex, a variant of Triplex with increased parallelism and a flexible rate of $d/(d+1)$ that retains similar security guarantees. However, the innovation came at the price of requiring TBCs with dn-bit tweaks, which are unusual and potentially costly for $d > 3$. In this paper, we investigate the limits of generalized Triplex- and Multiplex-type constructions for single-pass leakage-resilient authenticated encryption. Our contributions are threefold. First, we show that such constructions cannot provide CIML2 integrity for any tweak lengths below $dn/2$ bits. Second, we provide a birthday-bound attack for constructions with TBCs of tweak lengths between $dn/2$ and $(d-1)n + n/2$ bits. Finally, on the constructive side, we propose a family of single-pass leakage-resilient authenticated ciphers, dubbed Tweplex, that uses tweaks of $dn/2$ bits and provides a rate of $d/(d+1)$ while providing $n/2$-bit CIML2 integrity and CCAmL1 confidentiality.

1 Introduction

The design and analysis of schemes for authenticated encryption (with associated data) has been a highly active research area since it had been postulated to be a primitive of its own kind [37] that shall protect both the confidentiality of the message and the integrity of the ciphertext. Throughout the decades, variants like online schemes [4, 28], nonce-based, or deterministic authenticated encryption [39] arose. With them, a vast number of designs have been proposed that tried to optimize various needs between efficiency and security, most recently as the outcomes of the CAESAR [7] and NIST Lightweight competitions [41]. Beyond those general aspects, a series of research has been addressing robustness aspects, such as security under nonce-misuse [39], accidental nonce

repetitions [18] or settings where unverified would-be plaintexts could be released [1, 11]. This led to the strongest theoretical security notions for AE in Robust [27] and Subtle AE [3].

1.1 Leakage-resilient Authenticated Encryption

The usual theoretical security notions of AE treat the primitives as black boxes whereas real-world adversaries are free to exploit additional information from side channels. Such leakage can include but is not limited to information about timing, power consumption, or electromagnetic radiation, which can leak significant amounts of information about the internal state of the mode including its keys. One differentiates between two main attack vectors in the context of power consumption: Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [29]. In SPA attacks, an adversary observes leakages from encrypting a single input message under potentially multiple measurements to remove noise. DPA attacks study leakage from encrypting multiple inputs which provides new information about the internals of a cipher. Thus, the privacy of internal states can be reduced at a rate exponential in the number of distinct inputs.

Widespread AEAD schemes such as OCB [30, 38], GCM [20, 32], or CCM [19], which invoke a block cipher multiple times with a single key, are typically susceptible to DPA attacks. However, the protection of the underlying block cipher against leakage is usually left to the implementors and engineers who implement the components of a scheme so to minimize leakages as much as possible. For example, on a hardware level, the usual approaches for preventing leakage are to blur the signal with noise or special circuits. In contrast, on the implementation level, countermeasures include masking [12, 21], where the internal state of the device is split into several shares, which are then used in individual computations, or shuffling [25, 42]. Strong protection often adds considerable amounts of additional area, power, or efficiency penalties. Protection against DPAs lowers the performance of both software and hardware implementations of the algorithm by several orders of magnitude compared to the unprotected implementations in standard metrics (e.g. [22]). Consequently, a line of research of developing more efficient schemes that provide trade-offs has emerged. For an in-depth survey, we refer the interested reader to [6].

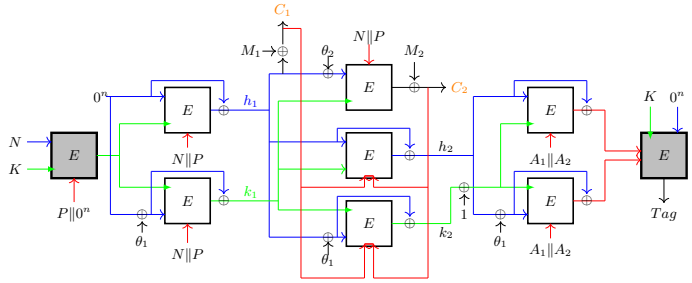
Instead of using a strongly protected block-cipher implementation for all invocations in an AEAD scheme, leakage-resistant modes of operations [3, 9, 10, 16] have shifted the paradigm to support dedicated leveled implementations. In this scenario, a crytographic primitive is called multiple times in an AEAD mode, but only certain calls to the primitive are strongly protected against DPA attacks while the remaining calls – that usually perform the majority of computations – are allowed to leak a certain amount of information to the adversary every time they are used. In summary, leakage-resilient AE schemes ensure security despite leakage at the cost of requiring a strong protection for a few primitive calls. This allows for reasonable efficiency with sufficient protection in practice.

1.2 Security Models for Leakage-resilient Nonce-based Authenticated Encryption

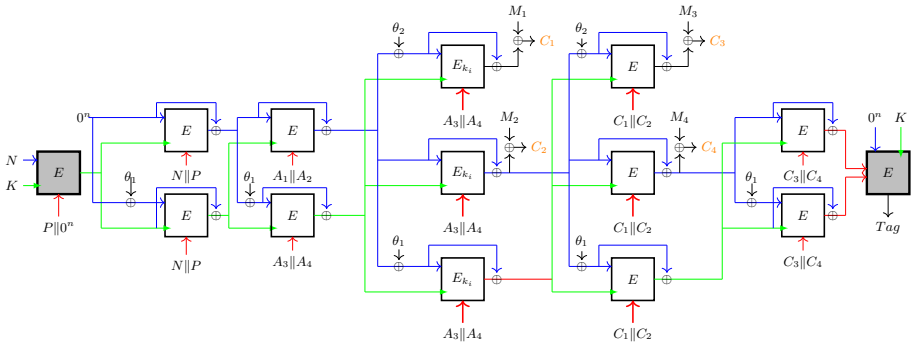
We consider the notions for leakage-resilient authenticated encryption by Guo et al. [9,10,23]. In [23], they proposed a comprehensive framework and the relations between them. As the strongest notions for AE, they identified (1) Ciphertext Integrity with Misuse-resistance and Leakage in encryption and decryption, or CIML2 [9, 10]; (2) chosen-ciphertext security with misuse-resilience and Leakage in encryption and decryption oracle, called CCAmL2 [23]; (3) moreover, schemes that process the messages in multiple passes could furthermore achieve the usual nonce-misuse resistance in the black-box setting without leakage [39].
In general, authenticated encryption schemes can be categorized into one- or multipass schemes. The latter can achieve both CIML2 and CCAmL2 security. However, in the context of lightweight cryptography single-pass modes are often prefered over two-pass modes. While nonce-based single-pass schemes can also achieve CIML2 security, CCAmL2 is out of range, but they can achieve CCA security with misuse-resilience and leakage in encryption, which was formulated as CCAmL1 [23].
A portfolio of leakage-resilient schemes for leveled implementations has been developed in the past few years. Misuse-resistant two-pass schemes include ISAP [16], ISAPv2 [15], or TEDT [8]. One-pass schemes include AEDT [10], or Triplex [40], and Multiplex [35]. All these constructions share a common design structure that consists of three independent modules [6, 14]: (i) the first module is called the *key-derivation function* (KDF) that employs a protected primitive to derive a session state from the nonce and the long-term master secret key; (ii) the second module is called the *message-processing function* (MPF), in which the plaintext (or the ciphertext) is encrypted (or decrypted) with a less protected primitive. In security treatments, the MPF module is often assumed to leak continuously. However, it adopts the idea of frequent rekeying to ensure that the security should not degrade badly. (iii) The third module is called the *tag-generation function* (TGF). It also employs a heavily protected primitive to derive the authentication tag. For nonce-based one-pass AE schemes, the MPF module will finally output a state as the input data to the TGF module, whereas for two-pass AE schemes [8,14,23,31] such a state is produced by hashing the ciphertext, nonce, and associated data with a less protected primitive. Following [23], Bellizia et al. [6] referred to leveled designs achieving both CIML2 and CCAmL1 security as *Grade-2-protected.* Recent lightweight one-pass leakage-resilient schemes, such as Ascon [17], Spook [5], or Triplex [40] are Grade-2 designs.

1.3 Revisiting Triplex and Multiplex

Leakage-resilient AE schemes are built primarily upon public permutations and tweakable block ciphers (TBCs). TBC-based leakage-resilient AE schemes require at least two calls to a primitive with n-bit block size for encrypting an n-bit message. While TEDT and TEDT2, which are two-pass AE schemes, achieve strong security guarantees (i.e., CIML2 and CCAmL2), they inherently offer a



(a) Encryption with Triplex [40].



(b) Encryption with Multiplex for $d = 2$ [35].

Fig. 1: Triplex and Multiplex. The darkened TBC calls in the KDF and TGF modules need strong protection whereas the white calls to E do not. The blue, green, and red lines represent block, key, and tweak input to the TBCs, respectively.

lower performance compared to single-pass modes. In this respect, Shen et al. [40] introduced a nonce-based single-pass AE scheme, called Triplex, which offered a rate of 2/3 while providing $n - \log_2(n)$-bit CCAmL1 and CIML2 security. Triplex is shown schematically in Fig. 1a.

Despite its efficiency under leakage, its design limits its throughput. In particular, one cannot make parallel TBC calls for en- or decryption. To address this shortcoming, Peters et al. proposed Multiplex [35], that allows a higher degree of parallelization at every round of the algorithm and offers a flexible rate of $d/(d+1)$, where d denotes the degree of parallelization, with $O(n - \log_2(dn))$-bits of security. In particular, at each round, it process dn bits message using $d+1$ TBC calls such that each one of them requires dn bits tweak. Despite of achieving a higher throughput, its primary disadvantage is the use of large tweak. Although, a few long-tweak variants of TBCs have been proposed [13, 33, 34], their security is far less understood compared to established designs and demands more cryptanalysis [24,36] to be stable. Moreover, instantiating Multiplex with TBCs of tweak lengths between $2n$ and $3n$ bits does not add anything extra over Triplex. Therefore, it is an interesting question how the security of Triplex and Multiplex is affected when they are instantiated with smaller tweak

size TBCs. In particular, our study is narrowed down to ask the following two questions:

1. *Can we instantiate* Triplex *using a TBC with* $< 2n$*-bit tweaks?*
2. *In the light of the importance of using TBCs with established tweak lengths, can we process* d *message blocks in* Multiplex *with* $(d+1)$ *calls to a TBC with* $< dn$*-bit tweaks?*

1.4 Our Contribution

Answering the question above is the central theme of the paper. More precisely, we tackle it in three steps as follows:

1. We show that for any choice of linear functions $f, g : \{0,1\} \to \{0,1\}^l$ that take $2n$-bit ciphertexts and produce l-bit outputs, if $l < n$, one can mount a forging attack with probability one on Triplex with a small constant number of queries. Furthermore, we show for tweak lengths of $l \in [n, 3n/2]$ bits, one can mount a forging attack on the construction with probabilty one in approximately $2^{n/2}$ queries. Finally, we show for tweak lengths of $l \in (3n/2, 2n)$ bits, an adversary can mount a forgery attack on the construction with success probability 2^{3n-2l} by making at least 2^{2n-l} queries.
2. For a given fixed parameter d, we show that for any choice of a pair of linear functions (f, g) that takes dn-bit ciphertexts and produces l-bits output, if $l < dn/2$, then one can mount a forging attack with probability 1 on Multiplex with a constant number of queries. We further show for tweak lengths of $l \in [dn/2, dn - n/2]$ bits, then one can mount a forging attack on the construction with probabilty 1 by making at least $2^{n/2}$ queries. Finally, we show for tweak lengths of $l \in (dn - n/2, dn)$ bits, an adversary can mount a forgery attack on the construction with success probability 2^{3n-2l} by making at least 2^{2n-l} queries. This provides an answer to our second question.
3. On a constructive side and to transform our theoretical results into practice, we propose an efficient Multiplex-type construction, dubbed Tweplex, which employs a TBC with $dn/2$-bits tweak and a rate of $d/(d+1)$. We show that Tweplex achieves the maximally possible $O(n/2)$-bit CIML2 and birthday-bound CCAmL1 security in the multi-user setting. Our construction maintains the rate of $d/(d+1)$ while using only half of the tweak size of Triplex and Multiplex. Hence, our construction provides higher throughput over Triplex and Multiplex while allowing still a rate of $4/5$ for $d = 4$ with established ciphers such as Deoxys-BC-128-384 or Skinny-128-384.

What remains is structured as follows: after briefly recalling the necessary notions, we study forgery results on Multiplex-type constructions in Section 3. We define Tweplex in Section 4 and prove its CIML2 and muCCAmL1 security under multiple users in Sections 5 and 6, respectively.

2 Preliminaries

NOTATIONS: For a finite set $\mathcal{X}$, we write $X \xleftarrow{\$} \mathcal{X}$ to denote that X is uniformly sampled from $\mathcal{X}$. We write $(X_1, X_2, \ldots, X_q) \xleftarrow{\$} \mathcal{X}$ to denote that each X_i is sampled uniformly at random from $\mathcal{X}$. For a set $\mathcal{X}$, we write $\mathcal{X} \xleftarrow{\cup} X$ to denote that $\mathcal{X} \leftarrow \mathcal{X} \cup \{X\}$. For a fixed n, we write the set of all n-bit binary strings as $\{0,1\}^n$, and $\{0,1\}^*$ denotes the set of all binary strings of arbitrary length. ε is used to denote the empty string. $|x|$ denotes the length of the bit string x. $\mathsf{msb}_c(Z)$ and $\mathsf{lsb}_c(Z)$ return the c most and least significant bits of a bit string Z, respectively. $x[i,j]$ denotes the substring from i-th bit to j-th bit of x. Concatenation of two strings x and y is denoted as $x\|y$. We also often write it as (x,y). If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}(x_1, x_2, \ldots, ; r)$ denotes running the algorithm $\mathcal{A}$ with randomness r on inputs $x_1, \ldots,$ and assigning the output to y. Equivalently, we can express the notation above as follows: let $y \xleftarrow{\$} \mathcal{A}(x_1, \ldots)$ be the result of picking r uniformly at random and then compute $\mathcal{A}(x_1, \ldots; r)$ and assign the result to the variable y. For an algorithm $\mathscr{A}$ and an oracle $\mathcal{O}$, we write $\mathscr{A}^{\mathcal{O}}$ to denote the output of $\mathscr{A}$ at the end of its interaction with $\mathcal{O}$.

2.1 Security Notions

A distinguisher $\mathscr{A}$ is an algorithm that tries to distinguish between two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$ via black-box interaction with one of them. At the end of its interaction, it returns a bit $b \in \{0,1\}$. The distinguishing advantage of $\mathscr{A}$ against $\mathcal{O}_0$ and $\mathcal{O}_1$ is defined as

$$\Delta_{\mathscr{A}}[\mathcal{O}_0; \mathcal{O}_1] \triangleq \left|\Pr[\mathscr{A}^{\mathcal{O}_0} = 1] - \Pr[\mathscr{A}^{\mathcal{O}_1} = 1]\right| ,$$

where the probabilities depend on the random coins of $\mathcal{O}_0$ and $\mathcal{O}_1$ and the random coins of the distinsguisher $\mathscr{A}$. The time complexity of the adversary is defined over the usual RAM model of computations. We call $\mathscr{A}$ a (q,t)-adversary if it asks at most q queries and runs in time at most t. We augment this notation by parameters e. g. in settings, where queries consist of en- and decryption oracles, we consider (q_e, q_d, t)-adversaries, assuming it asks at most q_e en- and q_d decryption queries, respectively. When queries consist of multiple blocks or bits, we augment it by σ for the number of blocks an adversary asks, and by p if an additional oracle to a primitive is given.

2.2 Tweakable Block Cipher

A tweakable block cipher with key space $\{0,1\}^\kappa$, tweak space $\{0,1\}^t$ and domain $\{0,1\}^n$ is a function $\widetilde{E} : \{0,1\}^\kappa \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that for each key $k \in \{0,1\}^\kappa$ and each tweak $\mathsf{t} \in \{0,1\}^t$, the function $\widetilde{E}(k, \mathsf{t}, \cdot)$ is a permutation over $\{0,1\}^n$. We call such TBCs (κ, t, n)-TBCs and define $\mathsf{TBC}(\kappa, t, n)$ for the set of all (κ, t, n)-TBCs. We call a function $\mathsf{IC} : \{0,1\}^\kappa \times \{0,1\}^t \times \{0,1\}^n$ an ideal TBC if $\mathsf{IC} \xleftarrow{\$} \mathsf{TBC}(\kappa, t, n)$. In this case, $\mathsf{IC}_k^{\mathsf{t}}$ is a random independent permutation

over $\{0,1\}^n$, for each $(k,\mathsf{t}) \in \{0,1\}^\kappa \times \{0,1\}^t$, even if k is public. We write $\widetilde{E}$ for TBCs, and in our ideal TBC-based security proofs, we use the notation IC. $\mathsf{TP}(t,n)$ denotes the set of all functions $\widetilde{\pi} : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that for all $\mathsf{t} \in \{0,1\}^t$, $\widetilde{\pi}(\mathsf{t},\cdot)$ is a permutation over $\{0,1\}^n$. We define the strong tweakable pseudorandom permutation (stprp) advantage of $\mathscr{A}$ against $\widetilde{E}$ as

$$\mathbf{Adv}^{\mathsf{stprp}}_{\widetilde{E}}(\mathscr{A}) \triangleq \Delta_{\mathscr{A}}\left[(\widetilde{E}_k, \widetilde{E}_k^{-1}); (\widetilde{\pi}, \widetilde{\pi}^{-1})\right],$$

where $k \xleftarrow{\$} \{0,1\}^\kappa$ and $\widetilde{\pi} \xleftarrow{\$} \mathsf{TP}(t,n)$.

2.3 Nonce-based Single-pass Authenticated Encryption

Let $\mathcal{K}, \mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{C}, \mathcal{T}$ be non-empty sets for keys, nonces, associated data, messages, ciphertext, and authentication tags, respectively. A nonce-based authenticated encryption scheme (nAE) consists of a pair of deterministic algorithms, called the encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$ and the decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}$. The correctness condition of a nAE scheme states that for every $K \in \mathcal{K}, N \in \mathcal{N}, A \in \mathcal{A}$, and $M \in \mathcal{M}$, we have $\mathcal{D}(K,N,A,\mathcal{E}(K,N,A,M)) = M$ and the tidiness condition of the nAE scheme states that for all $(K,N,A,C,T) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T}$, where $\exists M \in \mathcal{M}$ such that $\mathcal{E}(K,N,A,M) = (C,T)$, it holds that $\mathcal{E}(K,N,A,\mathcal{D}(K,N,A,(C,T))) = (C,T)$. Let $\Pi = (\mathcal{E},\mathcal{D})$ be a nAE scheme.

In this work, we focus on various security models of single-pass AEAD schemes under leakage; in particular, we limit our interest to notions for AEAD with nonce-misuse-resistant integrity and nonce-misuse-resilient confidentiality under potential leakage in encryption queries. Note that nonce-misuse-resistant confidentiality in the context of both en- and decryption-oracle leakage is impossible to achieve for a single-pass AE mode. We refer to the interested reader to the Appendix A of [8] for a detailed discussion.

Leakage depending on the implementation of an AEAD scheme can be viewed as two functions: leakage during encryption queries and leakage during decryption queries. In [9], Berti et al. have defined the leakage integrity notion with nonce-misuse-resistant by allowing only encryption leakage, which is referred to as CIML [9] notion, in which an adversary makes encryption and decryption queries, and obtains the corresponding responses. Along with that, the adversary also obtains leakages corresponding to the encryption queries. The final goal of the adversary in this model is to forge the construction with a valid tuple. However, this security notion has been extended from CIML [9] to $\mathsf{CIML2}$ [10], where the latter allows leakage from not only encryption, but also from decryption queries. Berti et al. [8] defined $\mathsf{muCIML2}$ as a multi-user distinguishing version of $\mathsf{CIML2}$. We focus on the strong multi-user notions (in their non-distinguishing variants) $\mathsf{muCIML2}$ for integrity and $\mathsf{muCCAmL1}$ for confidentiality both under leakage.

2.4 (Multi-user) Ciphertext Integrity under Misuse Leakage

We consider Ciphertext Integrity under Misuse Leakage ($\mathsf{CIML2}$) under leakage in both en- and decryption queries. In this security notion, an adversary

$\mathscr{A}$ is allowed to make queries to the encryption $L\mathcal{E}_{K_i}$ for any user i, and the decryption oracle $L\mathcal{D}_{K_i}$, and obtains the corresponding responses along with the leakages corresponding to the encryption and the decryption. Finally, the adversary submits a forging tuple (i, N, A, M, C, Tag) such that it is fresh. The forging advantage of the muCIML2 notion is then defined as the probability that (i, N, A, M, C, Tag) is valid. Formally, we define the muCIML2 notion following algorithm 2 as follows:

Definition 1. *Let the game of muCIML2 be defined in Algorithm 2. Let $\mathscr{A}$ be a (q, t)-muCIML2 adversary on an AEAD scheme $\Pi := (\mathcal{E}, \mathcal{D})$. Then, the advantage $\mathscr{A}$ is defined as*

$$\mathbf{Adv}_{\Pi}^{muCIML2}(\mathscr{A}) \triangleq \Pr\left[\mathscr{A}^{L\mathcal{E}_{\mathbf{K}}, L\mathcal{D}_{\mathbf{K}}, \widetilde{E}, \widetilde{E}^{-1}} \text{ forges}\right],$$

where the probability is taken over the u manu user keys $\mathbf{K} = (K_1, \ldots, K_u)$, randomness of $\mathscr{A}$, and the ideal TBC $\widetilde{E}$.

The algorithmic description of muCIML2 is given in Supporting Material A.

2.5 (Multi-user) Chosen-ciphertext Indistinguishability under Nonce Misuse and Leakage

For privacy under nonce repetitions, we follow [2]. In this context, an adversary $\mathscr{A}$ is allowed to make encryption queries and ideal TBC queries. We split the encryption oracle into two categories: $\mathcal{E}_1$ and $\mathcal{E}_2$, where $\mathscr{A}$ is allowed to repeat the nonce for same user during $\mathcal{E}_1$ queries but has to use a fresh nonce for every query to $\mathcal{E}_2$. Ultimately, the adversary has to distinguish between $\mathcal{E}_2$ and a random function. We define the advantage as follows:

$$\mathbf{Adv}_{\Pi}^{\mathsf{conf}}(\mathscr{A}) \triangleq \Delta_{\mathscr{A}}\left[(\mathcal{E}_{\mathbf{K}}^1, \mathcal{E}_{\mathbf{K}}^2, \widetilde{E}, \widetilde{E}^{-1}); (\mathcal{E}_{\mathbf{K}}^1, \$, \widetilde{E}, \widetilde{E}^{-1})\right].$$

We extend the notion above to incorporate leakage. Then, the adversary interacts with the en- and decryption oracles with possibly repeating nonces. It obtains the corresponding en- or decryption responses plus potential leakage during the encryption queries, i.e., the *decryption oracle does not leak*. Finally, the adversary submits a challenge encryption query corresponding for some user under a fresh nonce N. The challenger either encrypts that query or encrypts a randomly chosen message and responds with the corresponding ciphertext-tag pair. The security advantage of muCCAmL1 is then defined as the distinguishing advantage. We define muCCAmL1 security of an authenticated encryption scheme Π as in Algorithm 3 with respect to leakage and nonce-misuse resilience. An adversary $\mathscr{A}$ can query four oracles: a primitive oracle, the decryption oracle without leakage, the encryption oracle $L\mathcal{E}^1$ with a leakage function $\mathfrak{L}^{\mathcal{E}}$, and another encryption oracle $L\mathcal{E}^2$ with a leakage function $\mathfrak{L}^{\mathcal{E}}$. Using these queries, $\mathscr{A}$ has to distinguish between $L\mathcal{E}^2$ and a random funtion. We define the muCCAmL1 advantage as

$$\mathbf{Adv}_{\Pi}^{\mathsf{muCCAmL1}}(\mathscr{A}) \triangleq \Delta_{\mathscr{A}}\left[(L\mathcal{E}_{\mathbf{K}}^1, L\mathcal{E}_{\mathbf{K}}^2, \mathcal{D}, \widetilde{E}, \widetilde{E}^{-1}); (L\mathcal{E}_{\mathbf{K}}^1, L\$, \mathcal{D}, \widetilde{E}, \widetilde{E}^{-1})\right],$$

where the probability is taken over the u user keys $\mathbf{K} = (K_1, \ldots, K_u)$, the randomness of $\mathscr{A}$, and the ideal TBC $\widetilde{E}$. Note that we have considered leakage from only the encryption oracle. This is a weaker notion as compared to muCCAmL2, where both en- and decryption oracles leak. However, it is well-known that the notion muCCAmL2 is impossible to achieve for single-pass authenticated encryption schemes. Thus, we restrict our intesrest to muCCAmL1. An alternative game-based description of muCCAmL1 security notion is given in Supporting Material A.

3 Forgery Complexity on Triplex- and Multiplex-type Constructions

In this section, we show forging attacks on Triplex- and Multiplex-type constructions based on TBCs with varying tweak lengths. Since our attack algorithms and the corresponding analysis depend on a well-known combinatorial result, we briefly recall it here.

Theorem 1. *Let A and B be linear spaces and $f : A \to B$ be a linear map. Then, if $dim(A)$ is finite, we have $dim(A) = rank(f)+nullity(f)$, where $rank(f) = dim(f(A))$ and $nullity(f) = dim\{x : x \in A \wedge f(x) = 0\}$.*

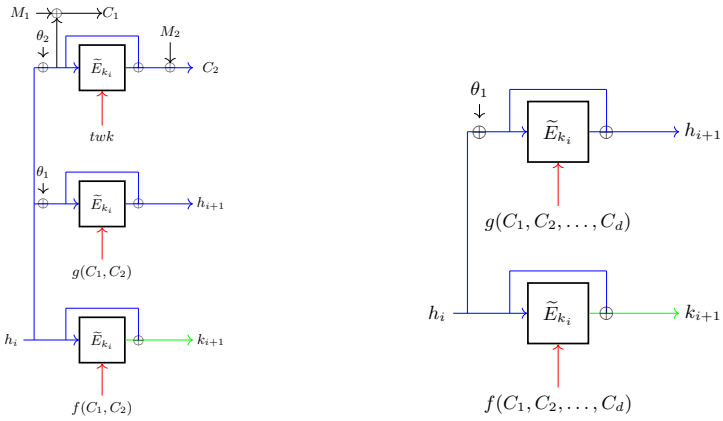
3.1 Forging Attack on Triplex with smaller Tweak

We start with the attack algorithm on Triplex. In each iteration of its message-processing module, Triplex encrypts two n-bit message blocks M_{2i}, M_{2i+1} and produces two ciphertext blocks C_{2i}, C_{2i+1}. Moreover, it generates the successive key and chaining-value pair (h, k) using two tweakable block ciphers with $2n$-bit tweak using C_{2i}, C_{2i+1} as the tweak. It has been shown in [40] that Triplex achieves n-bit CIML2 security. Moreover, it can be easily seen from the design of Triplex that it is impossible to process more than $2n$-bit message material in one iteration.

In the following, let those ciphertext blocks be denoted as C_1 and C_2. Let further t_1 be an l-bit tweak to the bottom TBC and t_2 be the other l-bit tweak to the middle TBC call in a round of Triplex with $l < 2n$. Those tweaks are used for creating the successive pair of key and chaining value. Assume that these two tweaks t_1, t_2 are created using two linear transformations $f, g : \{0,1\}^{2n} \to \{0,1\}^l$ such that $t_1 = f(C_1, C_2)$ and $t_2 = g(C_1, C_2)$. Let $\mathsf{Ker}(f) = \{x \in \{0,1\}^{2n} : f(x) = 0^l\}$ be the set of all preimages of f which are mapped to 0^l. $\mathsf{Ker}(g)$ is defined similarly. There are three distinct cases in which we attempt forgery on Triplex with l-bit tweak. The cases are as follows:

- Case A: There exist $x, y \in \mathsf{Ker}(f)$ such that $g(x) = g(y)$.
- Case B: There exist $x, y \in \mathsf{Ker}(g)$ such that $f(x) = f(y)$.
- Case C: None of the two conditions above holds.

We describe forgery-attack algorithms for each case in the following.



(a) Core of Triplex. (b) Core of Multiplex.

Fig. 2: Core of the Triplex and Multiplex constructions. where $C_1, C_2, \ldots$ stem from the preceding iteration.

□ *Forgery Algorithm for Case A:* The attack algorithm for Case A is depicted in Fig. 3. Let us briefly justify how the attack works. Since $x, y \in \mathsf{Ker}(f)$ such that $g(x) = g(y)$, we have $f(x \oplus z) = f(y \oplus z)$ and $g(x \oplus z) = g(y \oplus z)$ due to the linearity of f and g. We will use this fact to mount the forgery attack. Note that we have used the intermediate values (e_1 and e_2) of the first encryption query to set up the second encryption query such that the tweak used at the core component of the same round of the construction for generating the ciphertext blocks (C_1', C_2') is $f(C_1' \| C_2') = f(M_1' \oplus e_1 \| M_2' \oplus e_2) = f(x \oplus z)$. A similar argument holds for the tweak used at the middle TBC call $g(x \oplus z)$. Next, we have expilicitly used $y \oplus z$ as the two respective ciphertext blocks in the forging attempt that ensure the tweaks used to generate the successive chaining and updated key values are $f(y \oplus z)$ and $g(y \oplus z)$, respectively. The property that $f(x \oplus z) = f(y \oplus z)$ and $g(x \oplus z) = g(y \oplus z)$ ensures the forgery.

□ *Forging Algorithm for Case B:* The attack algorithm for Case B is similar to that of Case A, but concentrates on $f(x) = f(y)$ instead of $g(x) = g(y)$. When the boxed statements are included, Fig. 3 also describes Case B. The analysis of this case is then naturally almost identical to that of Case A. Note that similarly to Case A, the success probability depends on the proper choice of x, y, and z such that $f(x \oplus z) = f(y \oplus z)$, and $g(x \oplus z) = g(y \oplus z)$.

□ *Forging Algorithm for Case C:* In this case, we also try to achieve an internal two-block (h, k) collision. While Cases A and B achieved this by a collision in the tweak values, Case C constructs a collision in the tweak for one TBC call (that generates the chaining value). For the other calls, we make several queries with different tweaks and expect to get a collision. Thus, our attack will succeed

CASE A: $\exists\ x, y \in \mathsf{Ker}(f) : g(x) = g(y)$	CASE B: $\exists\ x, y \in \mathsf{Ker}(g) : f(x) = f(y)$
1 : `Choose` $z \in \{0,1\}^{2n} : f(z) \neq 0^l$;	`Choose` $z \in \{0,1\}^{2n} : g(z) \neq 0^l$;
2 : `Make an Encryption Query` $(N, A, M = (M_1 \| M_2))$;	
3 : `Let the response be` $(C = C_1 \| C_2,\ T)$;	
4 : `Compute` $e_1 = C_1 \oplus M_1,\ e_2 = C_2 \oplus M_2$;	
5 : `Compute` $M_1' \| M_2' = (x \oplus z) \oplus (e_1 \| e_2)$;	
6 : `Make an Encryption Query` $(N, A, M' = (M_1' \| M_2'))$;	
7 : `Let the response be` $(C',\ T')$;	
8 : `Forge` $(N, A, C'' = (y \oplus z),\ T')$;	

Fig. 3: Forgery algorithm in Cases A (without the boxed statements) and B (with the boxed statements) for Triplex.

probabilistically depending on the tweak length l. The algorithm for Case C is given in Fig. 4.

Let us briefly discuss how the attack works. Since $x_1, \ldots, x_a \in \mathsf{Ker}(g)$, we have $g(x_1 \oplus z) = \cdots = g(x_a \oplus z) = g(z)$ due to the linearity of g. Now we try to see if we could find (i, j) such that $k_i = k_j$, then we would be done and could set up the forgery in a similar manner that we have done for Case A. Now let us look at the probability of matching two keys k_i and k_j. Note that the k_i values are generated from the invocation of the TBC with the same key and input but with different tweaks. Hence,

$$\Pr[\exists\ i, j \in \{1, 2, \ldots, a\} : k_i = k_j] \leq a^2 \cdot 2^{-n}\,.$$

It is easy to see that a successful forgery happens in this case whenever we obtain a pair (i, j) with $k_i = k_j$. Hence, the probability of getting a successful forgery is bounded by $a^2 \cdot 2^{-n}$.

Now let us summerize the different settings depending on the used tweak lengths:

(i) $\boxed{l < n.}$ From the rank-nullity theorem, we have $\dim(\mathsf{Ker}(f)) \geq 2n - l > n$ and the same follows for $\ker(g)$. Let $\mathcal{B}_f$ and $\mathcal{B}_g$ be two bases for $\ker(f)$ and $\mathsf{Ker}(g)$, respectively. Then $|\mathcal{B}_f| \geq n + 1$ and $|\mathcal{B}_g| \geq n + 1$. Moreover, the dimension of $\{0,1\}^{2n}$ ensures that one of following properties will be satisfied: $\exists\ x, y \in \mathcal{B}_f$ which are linearly dependent to $\mathcal{B}_g$ which satisfy Case A or $\exists\ x, y \in \mathcal{B}_g$ which are linearly dependent on $\mathcal{B}_f$ which satisfies Case B. Thus, for $l < n$, at least one of Case A or B will happen. Hence, for tweaks of less than n bits, we can forge successfully with a constant number of queries.

(ii) $\boxed{n \leq l \leq 3n/2.}$ Here, we can have multiple cases: if any of the first two cases is satisfied for f and g, we will have successful forgery with only three queries. Otherwise, we consider Case C. Since $a = \min\{2^{2n-l}, 2^{\frac{n}{2}}\} =$

Case C: Cases A and B do not hold

1 : `Make an Encryption Query` $(N, A, M = (M_1 \| M_2))$;
2 : `Let the resonse be` $(C = (C_1 \| C_2), T)$;
3 : `Note internal round pair` (h_α, k_α), `used to create` $C_1 \| C_2$;
4 : `Choose` $z \in \{0,1\}^{2n} : \ g(z) \neq 0$;
5 : `Let` $a = \min\{2^{2n-l}, 2^{\frac{n}{2}}\}$;
6 : `Choose distinct` $x_1, x_2, \ldots, x_a \leftarrow \mathsf{Ker}(g)$;
7 : `Compute` $k_i \leftarrow \widetilde{E}(k_\alpha, t_1^i, h_\alpha)$, `where` $t_1^i \leftarrow f(x_i \oplus z)$, `for` $i = 1, \ldots, a$;
8 : `Find` (i, j) `such that` $k_i = k_j$;
9 : `Compute` $e_1 := C_1 \oplus M_1, \ e_2 := C_2 \oplus M_2$;
10 : `Compute` $M_1' \| M_2' = \ (x_i \oplus z) \oplus (e_1 \| e_2)$;
11 : `Make an Encryption Query` $(N, A, M' = (M_1' \| M_2'))$;
12 : `Let the response be` (C', T');
13 : `Forge with` $(N, A, C'' = (x_j \oplus z), T')$;

Fig. 4: Forgery algorithm in Case C for Triplex.

$2^{\frac{n}{2}}$, from the analysis of Case C, the adversary will be successful with probability one using at most $2^{\frac{n}{2}}$ queries.

(iii) $\boxed{l > 3n/2.}$ Again, if any of first two cases is satisfied, we will have successful forgery with only three queries. Otherwise, as the analysis of Case C in Fig. 4 states, the success probability of the forgery is at least $\frac{a^2}{2^n}$ for the value of a as in Line 5 of Fig. 4. Clearly, the value of a decreses as l increases. Thus, the security increases as the tweak length increases.

3.2 Forgery Attacks on Multiplex with $< dn$-bit TBCs

Multiplex encrypts d n-bit message blocks using $d+1$ TBC calls in each iteration. It processes these d blocks in the next iteration using them as the tweak of the TBC calls. Multiplex uses the same Hirose's compression function as Triplex for generating the subsequent key and chaining value. Unlike Triplex, Multiplex allows different values of d while achieving almost n-bit CIML2 security. In this section we will analyze the security of Multiplex when instantiated with a TBC with a tweak length of $< dn$ bits.

Similarly as for Triplex, the core component of Multiplex (see Fig. 2b) processes d ciphertext blocks (from the preceding iteration). Let t_1 and t_2 be the tweaks to two TBC calls in an iteration of the compression function to create the pair of key and chaining value. These two tweaks t_1, t_2 are created using two linear transformations $f, g : \{0,1\}^{dn} \to \{0,1\}^l$ such that $t_1 = f(C_1, C_2, \ldots, C_d)$ and $t_2 = g(C_1, C_2, \ldots, C_d)$. Let $\mathsf{Ker}(f) = \{x \in \{0,1\}^{dn} : f(x) = 0^l\}$ be the set of

Table 1: Forgery-attack properties for instantiations of Triplex and Multiplex with TBCs of varying tweak lengths.

(a) For Triplex.

Tweak length l	#Queries q	Success prob.
$l < n$	2	1
$n \leq l \leq 3n/2$	$2^{n/2}$	1
$3n/2 < l < 2n$	2^{2n-l}	2^{3n-2l}

(b) For Multiplex.

Tweak length l	#Queries q	Success prob.
$l < dn/2$	2	1
$dn/2 \leq l \leq dn - n/2$	$2^{n/2}$	1
$dn - n/2 < l < dn$	2^{2n-l}	2^{3n-2l}

all preimages of f which are mapped to 0^l. $\mathsf{Ker}(g)$ is defined similarly. Similarly as for Triplex, there are three distinct cases in which we attempt a forgery on Multiplex when instantiated with an l-bit TBC. Now we will analyze the security for three cases depending on f ands g as follows:

- Case A: There exist $x, y \in \mathsf{Ker}(f)$ such that $g(x) = g(y)$.
- Case B: There exist $x, y \in \mathsf{Ker}(g)$ such that $f(x) = f(y)$.
- Case C: None of the two conditions above holds.

Similarly as for Triplex, we can mount forgery attacks for the three cases above. For completeness, we present the attacks in Supporting Material D.
According to the cases mentioned above, one can mount the following generic attacks based on the length of the tweaks used:

(i) $\boxed{l < dn/2:}$ In this case, one can show that one of Case A or Case B gets satisfied, and hence, we can forge successfully with only two encryption queries.

(ii) $\boxed{dn/2 \leq l \leq dn - n/2:}$ In this case, if one of the first two cases gets satisfied for f and g, we will have a successful forgery with only three queries. Otherwise, we will mount an attack following Case C with $a = \min\{2^{dn-l}, 2^{n/2}\} = 2^{n/2}$. From the analysis of Case C, the adversary will be successful with probability one using at most $2^{n/2}$ queries.

(iii) $\boxed{l > dn - n/2:}$ Similarly as in the previous settings, it holds that if any of the first two cases is satisfied, we will have a successful forgery with three queries. Otherwise, the analysis of Case C in Fig. 7 states that the success probability of the forgery is at least $a^2 \cdot 2^{-n}$ for the value of a. Clearly, the value of a decreses as l increases. Thus, the security will increase as the tweak length increases.

4 The Tweplex Authenticated Cipher

Tweplex follows the three-step model suggested in [6] for designing leakage-resilient AEAD schemes. It uses a TBC with dn-bit tweaks as the underlying

Algorithm 1 Encryption Algorithm of Tweplex

11: **function** $\mathcal{E}(i, K_i, P_i, N, A, M)$
12: $d' \leftarrow dn/2$
13: $h_1 \| k_1 \leftarrow \mathrm{KDF}(i, K_i, P_i, N)$
14: $(C, twk) \leftarrow \mathrm{MPF}(h_1, k_1, A, M)$
15: $\mathsf{T} \leftarrow \mathrm{TGF}(i, K_i, twk, 0^n)$
16: Return (C, T)

21: **function** $\mathrm{KDF}(i, K_i, P_i, N)$
22: $k_0 \leftarrow \widetilde{E}(K_i, P_i \| 0^{d'-n}, N)$
23: $a \leftarrow \widetilde{E}(k_0, N \| P_i \| 0^{d'-2n}, 0^n)$
24: $b \leftarrow \widetilde{E}(k_0, N \| P_i \| 0^{d'-2n}, \theta_1) \oplus \theta_1$
25: **return** (a, b)

41: **function** $\mathrm{TGF}(i, K_i, twk, X)$
42: **return** $\widetilde{E}(K_i, twk \| 0^{d'-2n}, X)$

31: **function** $\mathrm{MPF}(h_1, k_1, A, M)$
32: $A_1 \| A_2 \| \cdots \| A_{da} \leftarrow PAD(A)$
33: $M_1 \| M_2 \| \cdots \| M_{dm} \leftarrow PAD(M)$
34: **for** $i \leftarrow 2 \ldots a$ **do**
35: $t_{i,1} \leftarrow \mathsf{msb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
36: $t_{i,2} \leftarrow \mathsf{lsb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
37: $k_i \leftarrow \widetilde{E}(k_{i-1}, t_{i,1}, h_{i-1}) \oplus h_{i-1}$
38: $h_i \leftarrow \widetilde{E}(k_{i-1}, t_{i,2}, h_{i-1} \oplus \theta_1) \oplus h_{i-1} \oplus \theta_1$
39: $X_1 \leftarrow \mathsf{msb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
40: $X_2 \leftarrow \mathsf{lsb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
41: $X_3 \leftarrow X_1 \oplus X_2$
42: **for** $i \leftarrow 1 \ldots m$ **do**
43: $k_{a+i} \leftarrow \widetilde{E}(k_{a+i-1}, X_1, h_{a+i-1}) \oplus h_{a+i-1}$
44: $h_{a+i} \leftarrow \widetilde{E}(k_{a+i-1}, X_2, h_{a+i-1} \oplus \theta_1) \oplus h_{a+i-1} \oplus \theta_1$
45: $C_{d(i-1)+1} \leftarrow M_{d(i-1)+1} + h_{a+i}$
46: **for** $j \leftarrow 2 \ldots d$ **do**
47: $e_j \leftarrow \widetilde{E}(k_{a+i-1}, X_3, h_{a+i-1} \oplus \theta_j) \oplus h_{a+i-1} \oplus \theta_j$
48: $C_{d(i-1)+j} \leftarrow M_{d(i-1)+j} + e_j$
49: $X_1 \leftarrow \mathsf{msb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
50: $X_2 \leftarrow \mathsf{lsb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
51: $X_3 \leftarrow X_1 \oplus X_2$
52: $twk_1 \leftarrow \widetilde{E}(k_{a+m}, X_1, h_{a+m}) \oplus h_{a+m}$
53: $twk_2 \leftarrow \widetilde{E}(k_{a+m}, X_2, h_{a+m} \oplus \theta_1) \oplus h_{a+m+1} \oplus \theta_1$
54: $twk \leftarrow twk_1 \| twk_2$
55: $C = C_1 \| C_2 \| C_3 \| \cdots \| C_{dm}$
56: **return** (C, twk)

primitive. Like Triplex and Multiplex, Tweplex consists of three distinct modules called key-derivation (KDF), message-processing (MPF) and tag-generation function (TGF), respectively. In a broader sense, the KDF module consists of a call to a TBC implementation that is strongly protected against DPA. It uses the secret key, the public key of the user, and the nonce as inputs and produces a pair of an initial key and a chaining-value (h_1, k_1). The MPF module takes (h_1, k_1) as its input and processes the associated data using two tweakable block-cipher calls in each iteration. This module is structurally very similar to Hirose's compression funtion [26]. Thereafter, it computes a multihash based on the MBL compression function, which is used in the design of the multihash function of Multiplex. Recall that the MBL compression function used in Multiplex consists of $(d+1)$ calls to a tweakable block cipher to encrypt a dn-bit part of the message such that each of the $(d+1)$ TBC calls uses a dn-bit tweak. Our design does the same but uses a TBC with only $dn/2$-bit tweaks. The final two TBC calls produce the pair of successive key and chaining value (h, k) for the next iteration. Finally, the TGF module consists again of a strongly protected call to the TBC, taking the fixed input 0^n, the secret key, and the output of the multihash concatenated with the required number of zeroes as the tweak and outputs T.

An algorithmic description of the construction is given in Fig. 1. An illustration for $d = 4$, with $8n$-bit message and associated data each and tweak sizes of $2n$ bit is shown in Fig. 5.

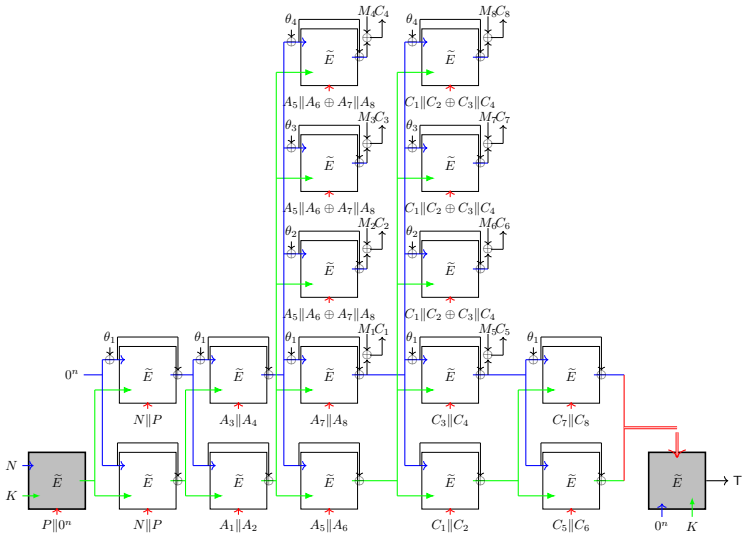


Fig. 5: Tweplex for $d = 4$.

5 Authentication Security of Tweplex

In this section, we show that Tweplex achieves ciphertext integrity in the muCIML2 setting for up to $2^{n/2}$ queries.

Theorem 2. *Consider an adversary $\mathscr{A}$ that tries to break the* muCIML2 *security of* Tweplex*. Assuming that $\mathscr{A}$ makes at most q_e encryption queries, q_d decryption queries with at most σ blocks in total, q_p primitive queries, over at most u users. Then, we have*

$$\mathbf{Adv}^{\mathsf{muCIML2}}_{\mathsf{Tweplex}}(\mathscr{A}) \leq \frac{u^2}{2^{2n+1}} + \frac{4(q^2+q)}{2^n} + \frac{q^2}{2^{2n}}\,.$$

where $q_c = q_e + q_d$, $q = \max\{q_p, q_c\}$ and $u \leq q$.

5.1 Query Types and Responses

The adversary $\mathscr{A}$ can make three types of queries: primitive, encryption, and decryption (or forging) queries. Here, we describe how the challenger responds to each query:

□ **Primitive Query:** For a primitive (ideal TBC) query $(J, t, x, \rightarrow)$, $\mathscr{A}$ will obtain $y = \widetilde{E}_J(t, x)$ and for query of the form $(J, t, y, \leftarrow)$, $\mathscr{A}$ gets $x = \widetilde{E}_J^{-1}(t, y)$. For both query types, the transcript will store an entry $(\texttt{Prim}, J, t, x, y, dir)$, where $dir \in \{\leftarrow, \rightarrow\}$. Hereafter, we will use $\star$ to denote that the direction is arbitrary.

□ **Encryption Query:** For an encryption query of the form (i, N, A, M), $\mathscr{A}$ will obtain a ciphertext-tag pair $C\|\mathsf{T}$ such that $C\|\mathsf{T} \leftarrow \mathcal{E}(i, N, A, M)$. Let $\mathbf{h}$ be the set of all h_i's generated during the production of $C\|\mathsf{T}$ and $\mathbf{k}$ be the set of all k_i's generated during the production of $C\|\mathsf{T}$. Due to the unbounded leakage assumption $\mathscr{A}$ will have full access to $\mathbf{h}$ and $\mathbf{k}$. An encryption query is stored in the form $(\mathsf{Enc}, i, N, A, M, C\|\mathsf{T}, \mathbf{h}, \mathbf{k})$. In addition, all internal primitive in- and outputs are stored in τ_p.

□ **Decryption Query:** For an decryption query of the form $(i, N, A, C\|\mathsf{T})$, $\mathscr{A}$ will get M such that $M \leftarrow \mathcal{D}(i, N, A, C\|\mathsf{T})$. Note that M can be a message or $\perp$ depending on authentication is successful or not, respectively. Let $\mathbf{h}$ be the set of all h_i's generated during decryption and $\mathbf{k}$ be the set of all k_i's generated during the same. Due to unbounded leakage assumption $\mathscr{A}$ will have $\mathbf{h}$ and $\mathbf{k}$. This query will be stored in the form $(\mathsf{Dec}, i, N, A, M, C\|\mathsf{T}, \mathbf{h}, \mathbf{k})$. Moreover, all internal primitive in- and outputs are stored in τ_p.
Note that we consider only non-trivial adversaries that do not make any decryption query of the form $(i, N, M, C\|\mathsf{T})$, after having observed $C\|\mathsf{T}$ as the output of an encryption query (i, N, A, M). We will try to bound $\mathscr{A}$'s successful forging probability. We will employ the standard technique: (i) define some bad events and show that the probability of having the bad events is low, and (ii) show that the forging probability of $\mathscr{A}$ is low if the defined bad events do not occur.

5.2 Defining Bad Events and Bounding Their Probabilities

We define a set of bad events as follows.

- Bad1: There are two different users sharing same secret key and same public constant, i.e., $\exists\ i \neq j,\ i, j \in \{1, 2, \ldots, u\} :\ K_i = K_j,\ P_i = P_j$.
- Bad2 : Two distinct primitive queries lead to a collision after their corresponding feed-forward operations, i.e $\exists$ $(\mathtt{Prim}, J, t, x, y, \star)$ and $(\mathtt{Prim},\ J',\ t',\ x',\ y',\ \star)$ such that $x \oplus y = x' \oplus y'$.
- Bad3: There is a collision in one of the KDF calls with a primitive query, i.e., $\exists\ (\mathtt{Prim}, J, t, x, y, \star) :\ J = K_i,\ t = P_i\|0^{(dn/2-n)}$, for some user i.
- Bad4: There is a collision in the output of a protected block of two different users having the same public constant and the same nonce, i.e., $\exists\ i \neq j,\ (\mathcal{D}, i, N_i^a, A_i^a, M_i^a,\ C_i^a\|\mathsf{T}_i^a, \mathbf{h}_i^a, \mathbf{k}_i^a),\ (\mathcal{E}, j, N_j^b, A_j^b, M_j^b, C_j^b\|\mathsf{T}_j^b, \mathbf{h}_j^b, \mathbf{k}_j^b)$ such that $k_{i,0}^a = k_{j,0}^b,\ N_i^a\|P_i = N_j^b\|P_j$. This condition considers the case of having same input for hash for two different users.
- Bad5: There is collision between tweak of two TGF calls, i.e., $\exists\ (i,\ N_i^a,\ A_i^a,\ M_i^a,\ C_i^a\|\mathsf{T}_i^a,\ \mathbf{h}_i^a, \mathbf{k}_i^a) \in \tau_d,\ (j, N_j^b, A_j^b, M_j^b, C_j^b\|\mathsf{T}_j^b, \mathbf{h}_j^b, \mathbf{k}_j^b)) \in \tau_{e/d}$ such that $h_{l_a+v_a+1}\|k_{l_a+v_a+1} = h_{l_b+v_b+1}\|k_{l_b+v_b+1}$, where $|A_i^a| = 4l_a,\ |C_i^a| = 4v_a,\ |A_j^b| = 4l_b,\ |C_j^b| = 4v_b$.
- Bad6: There is key-tweak pair collision among a KDF and TGF query, i.e., $\exists\ (i, N_i^a, A_i^a, M_i^a, C_i^a\|\mathsf{T}_i^a, \mathbf{h}_i^a, \mathbf{k}_i^a) \in \tau_d,\ (j, N_j^b, A_j^b, M_j^b, C_j^b\|\mathsf{T}_j^b, \mathbf{h}_j^b, \mathbf{k}_j^b)) \in \tau_c$ such that $K_i = K_j,\ h_{l_a+v_a+1}\|k_{l_a+v_a+1} = P_j\|0^n$, for some $j \in \{1, 2, \ldots, u\}$.

- Bad7 : There is a collision between a TGF call and a primitive query, i.e., $\exists\ (i, N_i^a, A_i^a,\ M_i^a, C_i^a \| Tag_i^a, \mathbf{h}_i^a, \mathbf{k}_i^a) \in \tau_c,\ (J, t, x, y, \star) \in \tau_p$ such that $J = k_i$, $t = h_{l_a+v_a+1} \| k_{l_a+v_a+1} \| 0^{(dn/2-2n)}$, for some user i.

We define an event Bad that is true if and only if any of the conditions above is satisfied. The following lemma bounds the probability of the event Bad:

Lemma 1. *It holds that*

$$\Pr[\mathtt{Bad}] \leq \frac{u^2}{2^{2n+1}} + \frac{4q^2}{2^n} + \frac{2q}{2^n} + \frac{q^2}{2^{2n}}\,.$$

Its proof can be found in Supporting Material C.1. Lemma 2 upper bounds the probability of forging when Bad did not occur:

Lemma 2. *It holds that*

$$\Pr[\mathscr{A}\ \textit{forges} \wedge \overline{\mathtt{Bad}}] \leq \frac{2q}{2^n}\,.$$

The proof of this lemma can be found in Supporting Material C.2. The result in Theorem 2 follows directly from Lemma 1 and 2.

6 Confidentiality Analysis of Tweplex

For privacy, we first define nonce-misuse-resilient security under the black-box assumption. Then, we will show muCCAmL1 security in the presence of leakage under the bounded-leakage assumption following a standard way as it does not have much technical novelty.

Theorem 3. *Suppose, an adversary $\mathscr{A}$ makes at most q_e encryption queries, that contain a total number of primitive calls of at most q_p, over at most u users. Then, for $q = \max\{q_p, q_e\}$, we have*

$$\mathbf{Adv}^{\mathsf{conf}}_{\mathsf{Tweplex}}(\mathscr{A}) \leq \frac{u^2}{2^{2n+1}} + \frac{4q^2+2q}{2^n} + \frac{q^2}{2^{2n}}\,.$$

Implication of the bound: This bound shows that we can achieve confidentiality for an adversary under the black-box assumption and up to $2^{n/2}$ queries. We will not consider security beyond the birthday bound since it is out of reach anyways. Hereafter, we focus on showing that Tweplex achieves muCCAmL1 security when the number of queries is roughly $2^{n/2}$.

Theorem 4. *Let $\widetilde{E} \in \mathit{TBC}(\{0,1\}^n, \mathcal{T}, \{0,1\}^n)$. Let $\mathscr{A}$ be a muCCAmL1 adversary on $\Pi[\widetilde{E}]_K = \mathsf{Tweplex}[\widetilde{E}]_K$ that is allowed to ask at most q_e encryption queries of at most σ dn-bit blocks and q primitive queries in total. Let $\mathcal{F}[\widetilde{E}]$ denote an iteration of Tweplex for message encryption, and $\mathcal{L}^{in}$, $\mathcal{L}^{out}$, and $\mathcal{L}^{\oplus}$ be leakage functions. Then*

$$\begin{aligned}\mathbf{Adv}^{\mathit{muCCAmL1}}_{\Pi[\widetilde{E}]}(\mathscr{A}) \leq\ & \mathbf{Adv}^{\mathit{conf}}_{\Pi[\widetilde{E}]}(q,\sigma) + 2\sigma \cdot \mathbf{Adv}^{\mathit{LUP\text{-}d}}_{\mathcal{F}[\widetilde{E}],\mathcal{L}^{in},\mathcal{L}^{out}}(p,q) + \\ & \sigma \cdot \mathbf{Adv}^{\mathit{XOR\$}}_{\mathcal{F}[\widetilde{E}],\mathcal{L}^{out},\mathcal{L}^{\oplus}}(q) + \mathbf{Adv}^{\mathit{muCIML2}}_{\Pi[\widetilde{E}]}(q,\sigma)\,,\end{aligned}$$

where the LUP-d and XOR$ games are given in Algorithms 4 and 5, respectively.

The proof follows similar steps as the qCPAmL2 proof by [31] on TEDT2 and the muCCAmL2 proof on TEDT [8]. However, we consider a multi-user version and assume that the adversary can query its decryption oracle only with some previous outputs from the encryption oracle due to the muCIML2 assumption. Moreover, we assume that the decryption oracle does not give any leakage.
For the proof of the black-box privacy game conf, we first define some Bad events in the nonce-misuse setting. Then, assuming those events did not happen, we will study the distinguishing probability under bounded leakage. Let $\mathscr{A}$ be a muCCAmL1 adversary for Tweplex that wants to win the game in Algorithm 3. $\mathscr{A}$ is allowed to make three kinds of queries.

6.1 Query Types and Responses

□ **Primitive Query:** For a primitive query $(J, t, x, \rightarrow)$, $\mathscr{A}$ will get $y = \widetilde{E}_J(t, x)$ and for a query of the form $(J, t, y, \leftarrow)$, $\mathscr{A}$ will get $x = \widetilde{E}_J^{-1}(t, y)$. For both kinds of queries, the challenger will store an entry $(\mathsf{Prim}, J, t, x, y, dir)$.

□ **Encryption Query:** For an encryption query of the form (i, N, A, M), $\mathscr{A}$ will get $C\|\mathsf{T}$ such that $C\|\mathsf{T} \leftarrow \mathcal{E}(i, N, A, M)$. $\mathscr{A}$ will also get the leakage corresponding to all internal primitive and other computations. This query will be stored in the form $(\mathcal{E}, i, N, A, M, C\|\mathsf{T}, \mathfrak{L}, \mathbf{hk})$, where **hk** is the set of all input-key pairs used during encryption in internal primitive calls and $\mathfrak{L}$ is the set of all leakage results during this computation. We divide all encryption queries into two types: $\mathcal{E}_1$, where the adversary can repeat the nonce for the same user and $\mathcal{E}_2$, where the adversary has to make queries for the same user under a fresh nonce. The $\mathcal{E}_2$ queries act as the *challenge query* for muCCAmL1 security.

□ **Decryption Query:** For a decryption query of the form $(i, N, A, C\|\mathsf{T})$, $\mathscr{A}$ will get M such that $M \leftarrow \mathcal{D}(i, N, A, C\|\mathsf{T})$. Note that M can be a message or $\perp$ depending on whether the authentication has been successful or not, respectively. $\mathscr{A}$ will not receive any leakage during the decryption process.
Note that muCCAmL1 security is defined as the distinguishing advantage between a real and an ideal world. Without loss of generality, we can assume that the muCIML2 security of Tweplex as the leakage assumptions are weaker in the muCCAmL1 model. Then, there will be no valid decryption query. So, it remains to prove the muCPAmL1 security of Tweplex. For that, we will first upper bound the probability in nonce-misuse scenario by defining Bad events. Assuming $\overline{\mathsf{Bad}}$, we show then that the output of Tweplex is indistinguishable from a randomly chosen message in the black-box setting. From this, we will argue muCCAmL1 security in the presence of (bounded) leakage. Let us define two games as follows:

- Game $\mathscr{G}_1$: This is the real-world encryption of a given message M, $\mathsf{Real}(M)$.
- Game $\mathscr{G}_4$: This is equivalent to the real-world $\mathsf{Real}(\$)$, that encrypts not the given M but a randomly chosen message M^* of the same length as M with the real encryption algorithm.

It holds that

$$\mathbf{Adv}^{mu\mathsf{CCAmL1}}_{\mathcal{E}_k, \mathcal{D}_k, \mathcal{L}_\mathcal{E}}(\mathscr{A}) = \Delta\mathscr{G}_{14}\,.$$

6.2 Confidentiality under Nonce Misuse and Bounded Leakage

Here, we define Bad events similar to those defined in our CIML2 proof.

- Bad1: There are two different users sharing the same secret key and the same public constant, i.e., $\exists\ i \neq j,\ i,j \in \{1,2,\ldots,u\} :\ K_i = K_j,\ P_i = P_j$.
- Bad2 : Two distinct primitive queries lead to a collision after their corresponding feed-forward operations, i.e $\exists$ $(\texttt{Prim}, J, t, x, y, \star)$ and $(\texttt{Prim},\ J',\ t',\ x',\ y',\ \star)$ such that $x \oplus y = x' \oplus y'$.
- Bad3: There is a collision between one of the KDF calls and a primitive query, i.e., $\exists\ (\texttt{Prim}, J, t, x, y, \star) :\ J = K_i,\ t = P_i \| 0^{(dn/2-n)}$, for some user i.
- Bad4: There is a collision in the output of a protected block of a $\mathcal{E}_2$ queries and another encryption query, along with same public constatnt and same nonce, i.e., $\exists\ (\mathcal{E}_2, i, N_i^a, A_i^a, M_i^a, C_i^a \| \mathsf{T}_i^a, \mathfrak{L}, \mathbf{hk}_i^a)$ the a-th query to a user i and another query $(\star, j, N_j^b, A_j^b, M_j^b, C_j^b \| \mathsf{T}_j^b, \mathfrak{L}, \mathbf{hk}_j^b)$ such that $k_{i,0}^a = k_{j,0}^b,\ N_i^a \| P_i = N_j^b \| P_j$.
- Bad5 : There is double block collision between an internal (h,k) pair, i.e., $\exists\ (\mathcal{E}_2, i,\ N_i^a, A_i^a, M_i^a, C_i^a \| \mathsf{T}_i^a, \mathfrak{L}, \mathbf{hk}_i^a)$ the a-th query to user i and another query $((\star, j,\ N_j^b, A_j^b, M_j^b, C_j^b \| Tag_j^b, \mathbf{hk}_j^b))$ such that $(h', k') = (h^\star, k^\star)$, where $(h', k') \in \mathbf{hk}_i^a$ and $(h^*, k^*) \in \mathbf{hk}_j^b$.
- Bad6: There is a key-tweak pair collision among a KDF and TGF query, i.e., $\exists\ (\mathcal{E}_2, i, N_i^a, A_i^a, M_i^a, C_i^a \| \mathsf{T}_i^a, \mathfrak{L}, \mathbf{hk}_i^a)$ and another query $\exists\ (\star, j, N_j^b, A_j^b, M_j^b, C_j^b \| \mathsf{T}_j^b, \mathfrak{L}, \mathbf{hk}_j^b)$ such that $(K_i = K_j) \wedge (P_i \| 0^n = h_{l_a+v_a+1} \| k_{l_b+v_b+1})$, for some $i,j \in \{1,2,\ldots,u\}$.
- Bad7 : There is collision between TGF call and primitive call, i.e., $\exists\ (\mathcal{E}_2,\ i,\ N_i^a,\ A_i^a,\ M_i^a, C_i^a \| \mathsf{T}_i^a, \mathfrak{L}, \mathbf{hk}_i^a)$ and $(\texttt{Prim}, J, t, x, y, \star)$ such that $t = h_{l_a+v_a+1} \|\ k_{l_a+v_a+1} \|\ 0^{(dn/2-2n)} \wedge J = K_i$.

We define the event Bad that is true if and only if any one of the conditions above is satisfied. One can see that these Bad conditions are similar to those of CIML2. Moreover, note that the adversary can not repeat a nonce for $\mathcal{E}_2$ queries which restrains it from obtaining information compared to a CIML2 adversary. The probability of these Bad conditions is upper bounded by the probability for a CIML2 adversary to forge successfully. We obtain

$$\Pr[\texttt{Bad}] \leq \frac{u^2}{2^{2n+1}} + \frac{4q^2}{2^n} + \frac{2q}{2^n} + \frac{q^2}{2^{2n}}\,. \tag{1}$$

If none of these events happen, all outputs from the KDF will be fresh for each $\mathcal{E}_2$ query. Moreover, from the absence of $\texttt{Bad}_2$, it follows that all internal primitive calls are pairwise unique. The tweaks used in the TGF module for every $\mathcal{E}_2$ query is also fresh. Then, there is no difference between $Real(M)$ and $Real(\$)$ in the black-box setting under the assumption of nonce-misuse resilience.

6.3 Proof Idea of muCCAmL1 Security

A detailed treatment and the game definitions are in Supporting Material E, where we show that

$$\Delta\mathscr{G}_{14} \leq \mathbf{Adv}^{\mathsf{conf}}_{\Pi[\tilde{E}]}(q,\sigma) + 2\sigma \cdot \mathbf{Adv}^{\mathsf{LUP\text{-}d}}_{\mathcal{F}[\tilde{E}],\mathcal{L}^{in},\mathcal{L}^{out}}(p,q) + \sigma \cdot \mathbf{Adv}^{\mathsf{XOR\$}}_{\mathcal{F}[\tilde{E}],\mathcal{L}^{out},\mathcal{L}^{\oplus}}(q) + \mathbf{Adv}^{\mathsf{muCIML2}}_{\Pi[\tilde{E}]}(q,\sigma)\,.$$

Though, the reductions are standard, and for the sake of space limitations, here, we provide only the core ideas.

Without leakage and in the absence of Bad, we can ensure indistinguishability between $\mathscr{G}_1$ and $\mathscr{G}_4$. In the presence of leakage, we show that the security of the scheme reduces to the SPA security of a single block-cipher call under the security assumption in [6]. Following the definition of multi-user CCAmL1 security, this is equivalent to the difference between $\mathscr{G}_1$ and $\mathscr{G}_4$. Now, we will argue the muCCAmL1 security as follows:

From the definition of muCCAmL1, we consider leakage only in encryption direction. The weaker (bounded-)leakage assumption in muCCAmL1 compared to CIML2 adversary allows us to reduce security to a CIML2 adversary with equal resources and assume CIML2 security in the following. We assume the bound on the leakage function using the hard-to-invert property introduced in [43] and also used in TEDT [8], TEDT2 [31].

For challenge queries (to $\mathcal{E}_2$), the adversary has to submit a fresh nonce. Then, k_0 will be fresh up to the birthday bound as it is the output of a block-cipher call with the secret key. This k_0 will be used as key only in two block-cipher calls with different inputs for computing (h_1, k_1). For processing the associated data, some key k_i can be used twice with input h_i and $h_i \oplus \theta_1$. Following a similar argument, two such invocations pass randomness about h_i, k_i to h_{i+1}, k_{i+1}. Similarly, while processing messages, the key and chaining values are used only a few times and are updated in each iteration. Each key k_s is used in $(d+1)$ block-cipher calls with input $h_s, h_s \oplus \theta_1, h_s \oplus \theta_2, \ldots, h_s \oplus \theta_d$. Therefore, the secrecy and randomness will be maintained in the next iteration. Moreover, at the end in the TGF, there is a protected block-cipher call with the long-term secret key that is assumed to not leak any significant information. The only remaining leakage is due to the XORs for creating the ciphertext as $C_i^j \leftarrow M_i^j \oplus e_i^1$ where $e_i^j \leftarrow E(k_{a+i-1}, t, h_{a+i}^j \oplus \theta_j) \oplus h_{a+i}^j \oplus \theta_j$ for $j = 2, 3, \ldots, d$ and $e_i^1 = h_{a+i}$, where subscript i denoted number of block and superscript j denote the number of sub-block of block i. For these, we use the similar LOR2 leakage assumption as defined for TEDT [8] and the XOR$ assumption in TEDT2 [31]. Moreover, those computations involve the hidden internal value h_{a+i-1}. Thus, from the low probability of the Bad events and the mentioned leakage assumptions above, we achieve muCCAmL1 security up to the birthday bound.

7 Conclusion

We have shown that Multiplex-type constructions that use TBCs with smaller tweaks than proposed cannot achieve beyond-birthday-bound CIML2 security but can increase the throughput. We presented Tweplex, a birthday-bound-secure scheme that uses the minimal tweak length under a higher throughput. It remains an interesting research problem to find another rekeying or message processing function that will give a Grade-2 AE scheme with a higher rate and $> n/2$-bit security.

References

1. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.
2. Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting Authenticated Encryption Robustness with Minimal Modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO III*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.
3. Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.
4. Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2001.
5. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. *IACR Trans. Symmetric Cryptol.*, 2020(S1):295–349, 2020.
6. Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography - A Practical Guide Through the Leakage-Resistance Jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.
7. Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. https://competitions.cr.yp.to/caesar.html, last update 20 Feb 2019, last accessed 18 July 2023, 2014.
8. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a Leakage-Resistant AEAD Mode for High Physical Security Applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.
9. Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext Integrity with Misuse and Leakage: Definition and

Efficient Constructions with Symmetric Primitives. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *AsiaCCS*, pages 37–50. ACM, 2018.

10. Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.
11. Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR Trans. Symmetric Cryptol.*, 2019(4):119–146, 2019.
12. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
13. Benoît Cogliati, Jérémy Jean, Thomas Peyrin, and Yannick Seurin. A Long Tweak Goes a Long Way: High Multi-user Security Authenticated Encryption from Tweakable Block Ciphers. *IACR Cryptol. ePrint Arch.*, page 846, 2022.
14. Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges Resist Leakage: The Case of Authenticated Encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT II*, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240. Springer, 2019.
15. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.
16. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - Towards Side-Channel Secure Authenticated Encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.
17. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2 Submission to the CAESAR Competition. https://competitions.cr.yp.to/round3/asconv12.pdf, September 15 2016. Submission to the CAESAR competition.
18. Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Beyond Birthday Bound Secure MAC in Faulty Nonce Model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT I*, volume 11476 of *Lecture Notes in Computer Science*, pages 437–466. Springer, 2019.
19. Morris Dworkin. NIST Special Publication 800-38C – Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality [including updates through 7/20/2007]. Technical report, U.S. National Institute of Standards and Technology, 2004.
20. Morris Dworkin. NIST Special Publication 800-38D – Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Technical report, U.S. National Institute of Standards and Technology, 2007.
21. Louis Goubin and Jacques Patarin. DES and Differential Power Analysis (The "Duplication" Method). In Çetin Kaya Koç and Christof Paar, editors, *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
22. Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. Multiparty Computation: How Large Is the Gap for AES? In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2013.

23. Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated Encryption with Nonce Misuse and Physical Leakage: Definitions, Separation Results and First Construction - (Extended Abstract). In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.
24. Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the Impossible: Automated Search for Full Impossible-Differential, Zero-Correlation, and Integral Attacks. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 128–157. Springer, 2023.
25. Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.
26. Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.
27. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust Authenticated-Encryption AEZ and the Problem That It Solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT I*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44. Springer, 2015.
28. Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO I*, volume 9215 of *Lecture Notes in Computer Science*, pages 493–517. Springer, 2015.
29. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
30. Ted Krovetz and Phillip Rogaway. Ocb (v1.1). https://competitions.cr.yp.to/round3/ocbv11.pdf, 2016.
31. Eik List. TEDT2 - Highly Secure Leakage-Resilient TBC-Based Authenticated Encryption. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT*, volume 12912 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2021.
32. David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.
33. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight Authenticated Encryption Mode Suitable for Threshold Implementation. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT II*, volume 12106 of *Lecture Notes in Computer Science*, pages 705–735. Springer, 2020.
34. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Secret Can Be Public: Low-Memory AEAD Mode for High-Order Masking. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO III*, volume 13509 of *Lecture Notes in Computer Science*, pages 315–345. Springer, 2022.
35. Thomas Peters, Yaobin Shen, and François-Xavier Standaert. Multiplex: TBC-based Authenticated Encryption with Sponge-Like Rate. `https://dial.uclouvain.be/pr/boreal/object/boreal%3A273131/datastream-/PDF_01/view`, 2023.
36. Lingyue Qin, Xiaoyang Dong, Anyu Wang, Jialiang Hua, and Xiaoyun Wang. Mind the TWEAKEY Schedule: Cryptanalysis on SKINNYe-64-256. In Shweta Agrawal

and Dongdai Lin, editors, *ASIACRYPT I*, volume 13791 of *Lecture Notes in Computer Science*, pages 287–317. Springer, 2022.
37. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS*, pages 98–107. ACM, 2002.
38. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *CCS*, pages 196–205. ACM, 2001.
39. Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.
40. Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Cassiers, and Corentin Verhamme. Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):135–162, 2022.
41. Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, , Lawrence E. Bassham, Jinkeon Kang, Noah D. Wallerand John M. Kelsey, and Deukjo Hong. NIST IR 8454 – Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process. Technical report, US National Institute of Standards and Technology, June 2023.
42. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.
43. Yu Yu and François-Xavier Standaert. Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.

Supporting Material

A Algorithmic Description of muCCAmL1 and muCIML2

Algorithm 2 The muCIML2 experiment G_{Π}^{CIML2}.

procedure INITIALIZE
  $K_1, K_2, \ldots, K_u \xleftarrow{\$} \mathcal{K}$
  $b \leftarrow 0$
  $S \leftarrow \emptyset$

function $\mathrm{L}\mathcal{E}(i, K_i, P_i, N, A, M, \mathfrak{L})$
  $R \xleftarrow{\$} \mathcal{R}$
  $(C, T) \leftarrow \mathcal{E}(K_i, P_i, N, A, M)$
  $S \xleftarrow{\cup} \{(i, N, A, C, T)\}$
  $L \leftarrow \mathfrak{L}_{K_i}^{\mathcal{E}}(P_i, N, A, M, R)$
  return (C, T, L)

function $\mathrm{L}\mathcal{D}(i, K_i, P_i, N, A, C, T, \mathfrak{L})$
  $R \xleftarrow{\$} \mathcal{R}$
  $M \leftarrow \mathcal{D}(K_i, P_i, N, A, C, T)$
  $L \leftarrow \mathfrak{L}_{K_i}^{\mathcal{D}}(P_i, N, A, C, T, R)$
  return (M, L)

function FINALIZE
  return b

function $L\mathcal{D}_{ch}(i, K_i, P_i, N, A, C, T, \mathfrak{L})$
  $R \xleftarrow{\$} \mathcal{R}$
  $L \leftarrow \mathfrak{L}_{K_i}^{\mathcal{D}}(P_i, N, A, C, T, R)$
  if $(i, N, A, C, T) \in S$ **then**
    return $(\perp, L)$
  if $\mathcal{D}(K_i, P_i, N, A, C, T) \stackrel{?}{=} \perp$ **then**
    return $(\perp, L)$
  $M \leftarrow \mathcal{D}(K_i, P_i, N, A, C, T)$
  $b \leftarrow 1$
  return (M, L)

function $\widetilde{E}(K, T, X)$
  $Y \leftarrow \widetilde{E}(K, T, X)$
  return Y

function $\widetilde{E}^{-1}(K, T, Y)$
  $X \leftarrow \widetilde{E}^{-1}(K, T, Y)$
  return X

B Description of The Modules in Tweplex

B.1 The MBL Compression Function

Multi-block length compression function based on tweakable block cipher, proposed by Peters et al. [35], is used in building a multi-block length hash function in the ideal cipher model based on a tweakable block cipher with n-bit key and dn bits tweak. MBL compression function can be regarded as a generalization of Hirose's double-block length compression function [26] based on the ideal cipher model to output arbitrary blocks without increasing the size of key. Let $d \geq 1$ be some public parameter and $\widetilde{E} : \{0,1\}^n \times \{0,1\}^{dn/2} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher. Then, $F : \{0,1\}^{(d+1)n} \times \{0,1\}^n \to \{0,1\}^{(d+1)n}$ is a multi-block compression function such that

$$(h_i, k_i, e_i^1, \ldots, e_i^{d-1}) = F(h_{i-1}, k_{i-1}, e_{i-1}^1, \ldots, e_{i-1}^{d-1}, t_i),$$

Algorithm 3 The muCCAmL1 experiment.

11: **procedure** INITIALIZE
12: $K_1, K_2, \ldots, K_u \xleftarrow{\$} \mathcal{K}$
13: $b \xleftarrow{\$} \{0,1\}$
14: $S_1, S_2, \ldots, S_u \leftarrow \emptyset$

21: **function** $\widetilde{E}(K, T, X)$
22: $Y \leftarrow \widetilde{E}(K, T, X)$
23: **return** Y

26: **function** $\widetilde{E}^{-1}(K, T, Y)$
27: $X \leftarrow \widetilde{E}^{-1}(K, T, Y)$
28: **return** X

31: **function** FINALIZE(b')
32: **return** $b = b'$

41: **function** $\mathrm{L}\mathcal{E}_1(i, K_i, P_i, N, A, M, \mathfrak{L})$
42: $R \xleftarrow{\$} \mathcal{R}$
43: $(C, T) \leftarrow \mathcal{E}(K_i, P_i, N, A, M)$
44: $S_i \xleftarrow{\cup} N$
45: $L \leftarrow \mathfrak{L}^{\mathcal{E}}_{K_i}(P_i, N, A, M, R)$
46: **return** (C, T, L)

51: **function** $\mathrm{L}\mathcal{E}_2(i, K_i, P_i, N, A, M, \mathfrak{L})$
52: $R \xleftarrow{\$} \mathcal{R}$
53: **if** $N \in S_i$ **then**
54: **return** $\perp$
55: **if** $b = 0$ **then**
56: $(C, T) \leftarrow \mathcal{E}(K_i, P_i N, A, M)$
57: **else**
58: $M^* \xleftarrow{\$} \{0,1\}^{|M|}$
59: $M \leftarrow M^*$
60: $(C, T) \leftarrow \mathcal{E}(K_i, P_i, N, A, M)$
61: $S_i \xleftarrow{\cup} N$
62: $L \leftarrow \mathfrak{L}^{\mathcal{E}}_{K_i}(P_i, N, A, M, R)$
63: **return** (C, T, L)

71: **function** $\mathcal{D}(i, K_i, P_i, N, A, C, T, \mathfrak{L})$
72: $M \leftarrow \mathcal{D}(K_i, P_i, N, A, C, T)$
73: **return** M

where $h_i, k_i, e_i^1, \ldots, e_i^{d-1}, t_i \in \{0,1\}^n$ and $h_{i-1}, k_{i-1}, e_{i-1}^1, \ldots, e_{i-1}^{d-1} \in \{0,1\}^n$ are computed as follows:

$$\begin{cases} h_i = \widetilde{E}_{k_{i-1}}(e_{i-1}^1 \| \ldots \| e_{i-1}^{d-1} \| t_i, h_{i-1}) \oplus h_{i-1} \\ k_i = \widetilde{E}_{k_{i-1}}(e_{i-1}^1 \| \ldots \| e_{i-1}^{d-1} \| t_i, h_{i-1} \oplus \theta_1) \oplus (h_{i-1} \oplus \theta_1) \\ e_i^1 = \widetilde{E}_{k_{i-1}}(e_{i-1}^1 \| \ldots \| e_{i-1}^{d-1} \| t_i, h_{i-1} \oplus \theta_2) \oplus (h_{i-1} \oplus \theta_2) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ e_i^{d-1} = \widetilde{E}_{k_{i-1}}(e_{i-1}^1 \| \ldots \| e_{i-1}^{d-1} \| t_i, h_{i-1} \oplus \theta_d) \oplus (h_{i-1} \oplus \theta_d), \end{cases}$$

where $\theta_1, \theta_2, \ldots, \theta_d$ are non-zero distinct constants.

B.2 Key-derivation Function

For a given parameter d, the KDF module of Tweplex first produces an initial key k_0 using a heavily protected TBC that takes a secret key K_i for user i, dn bits tweak $P_i \| 0^{(d-1)n}$, and nonce N as input. Then, it uses the initial key k_0 in the next two mildly protected tweakable block cipher as a key with input 0^n and $0^n \oplus \theta_1$, for some publicly chosen constant θ_1 and tweak $N \| P_i \| 0^{(d-2)n}$ for user i to both TBC calls. Output of these two TBCs are masked with their corresponding input to produce the initial key, chaining value pair (h_1, k_1). Note that, the use of a heavily protected TBC in the first part of the KDF module is used to protect the master secret key against leakages.

B.3 Message-processing Function

The MPF module of Tweplex consists of two parts: (a) one that processes the associated data and (b) another one that encrypts message blocks. The initial input to the module is the key, chaining value pair (h_1, k_1), which is then feeded into the Hirose's compression function to process the associated data blocks such that the tweak size to each of the tweakable block cipher is $dn/2$ bits. Followed by, it computes a multi-block hash function based on the MBL compression function that is build upon $(d+1)$ tweakable block cipher calls such that each of the TBC takes n bit input, n bit key and $dn/2$ bits tweak. This compression function is iterated as long as there are message blocks need to be processed. Note that, at each round of the multi-block hash function d message blocks are processed to generate d ciphertext blocks. At the end, we use two TBC to absorb last d block ciphertext in exact similar process. Output of these two TBC (h', k') concatinated with $0^{(d-2)n}$ becomes the tweak of TGF. We would like to note here that this multi-block hash funtion is very similar to that of used in the Multiplex construction except the use of latest cipher text blocks. It is easy to see that Tweplex achieves a rate of $d/(d+1)$ using TBC with tweak length $dn/2$-bits instead of dn bits as Triplex or Multiplex.

B.4 Tag-generation Function

This module of the construction consists of a single invocation of strongly protected tweakable block cipher with the master secret key as key, fixed input 0^n and the output of the multihash $h'\|k'\|0^{(d-2)n}$ as tweak. As before, strong protection to the TBC ensures to protect the master secret key from leaking. Finally, output of this tweakable block cipher is given as the tag.

C Authenticity Proof of Tweplex

C.1 Proof of Lemma 1

Let $q_c = q_e + q_d$ and q_p be the number of construction and primitive queries, respectively. We will bound the probability of Bad as follows:

□ **Bounding** Bad1*:* The key and the public values corresponding to an user is chosen uniformly at random over $\{0,1\}^n \times \{0,1\}^n$. Thus, for any two user i and j, we have $\Pr[K_i = K_j \wedge P_i = P_j] \leq 2^{-2n}$. As there is a total $\binom{u}{2}$ possible pairs of users, we have

$$\Pr[\texttt{Bad1}] \leq \binom{u}{2} \cdot \frac{1}{2^{2n}} \leq \frac{u^2}{2^{2n+1}}. \tag{2}$$

□ **Bounding** $\texttt{Bad2}$*:* From the definition of a tweakable block cipher, for any two $(\texttt{Prim}, J, t, x, y) \neq (\texttt{Prim}, J', t', x', y') \in \tau_p$ we have

$$\Pr[x \oplus y = x' \oplus y'] \leq \frac{1}{2^n}.$$

Moreover there are q_p many elements in τ_p. So

$$\Pr[\texttt{Bad2}] \leq \frac{\binom{q_p}{2}}{2^n} \leq \frac{q_p^2}{2^{n+1}}$$

□ **Bounding** $\texttt{Bad3}$: It is easy to see that for any primitive query $(\texttt{Prim}, J, t, x, y)$, $\mathscr{A}$ can target such user's secret key whose public parameter satisfy $t = P_i \| 0^{(\frac{dn}{2} - n)}$. As K_i's are distributed uniformly, i.e., for any user i, $\Pr[J = K_i] = \frac{1}{2^n}$, we have

$$\Pr[\texttt{Bad3}] \leq \frac{uq_p}{2^n} \tag{3}$$

□ **Bounding** $\texttt{Bad4} \wedge \overline{\texttt{Bad1}}$*:* Let i, j $(\neq i)$ be two users such that $N_i^a \| P_i = N_j^b \| P_j$. Assuming $\overline{\texttt{Bad1}}$, $P_i = P_j$ ensures that $K_i \neq K_j$. Thus, $k_{i,0}^a$ and $k_{j,0}^b$ are outputs of block-cipher calls under different keys, and the probability that the outputs match is $\leq \frac{1}{2^n}$. $\mathscr{A}$ can target at most u users' secret keys. Hence,

$$\Pr[\texttt{Bad4} \wedge \overline{\texttt{Bad1}}] \leq \frac{uq_d}{2^n}. \tag{4}$$

□ **Bounding** $\texttt{Bad5} \wedge \overline{\texttt{Bad3}}$*:* As E is considered as an indistinguishable from a random tweakable permutation, the probability that $E(J_i, T_i, x_i) \oplus x_i = E(J_j, T_j, x_j) \oplus x_j$ for some $i, j \in \{1, 2, \ldots, q_p\}$ is $\leq \frac{1}{2^n}$. Moreover collision in tweak of two TGF calls is equivalent to collision in hash H. Observe that tweak used for creation of h and tweak used for creation of k are independent. So clearly Collision in Hash $H \Leftrightarrow$ Collision in any one of h, k. Moreover Collision in one of h, k is equivalent to $\texttt{Bad2}$. Hence, we have

$$\Pr[\texttt{Bad5} \wedge \overline{\texttt{Bad3}}] = 0\,.$$

□ **Bounding** $\texttt{Bad6} \wedge \overline{\texttt{Bad3}} \wedge \overline{\texttt{Bad5}}$*:* Assuming that $\texttt{Bad3}$ and $\texttt{Bad6}$ did not occur, at least one of $h_{l_a+v_a+1}$, $k_{l_a+v_a+1}$ will differ from all previous outcomes, i.e., chosen uniformly at random from a set of size at least $(2^n - q_p)$. Thus, either $\Pr[h_{l_a+v_a+1} = P_j] \leq \frac{1}{2^n - q_p} \leq \frac{1}{2^{n-1}}$, or $\Pr[k_{l_a+v_a+1} = 0^n] \leq \frac{1}{2^n - q_p} \leq \frac{1}{2^{n-1}}$ (assuming $q_p \leq 2^{n-1}$). We obtain two cases:

- $i = j$: In this case, $K_i = K_j$. Therefore, the target value of $\mathscr{A}$ is a unique value $P_i \| 0^n$. Hence

$$\Pr[\texttt{Bad6} \wedge \overline{\texttt{Bad3}} \wedge \overline{\texttt{Bad5}} | i = j] \leq \frac{2q_c}{2^n}\,.$$

- $\underline{i \neq j}$: In this case, we have $\Pr[K_i = K_j] \leq \frac{1}{2^n}$, as secret keys are distributed uniformly among users. Thus, in this case a collision between KDF and TGF calls will happen iff $K_i = K_j \wedge h_{l_a+v_a+1} = P_j \wedge k_{l_a+v_a+1} = 0^n$. Using a similar approach as in the previous case for freshness of $h_{l_a+v_a+1}$ or $k_{l_a+v_a+1}$

$$\Pr[\mathtt{Bad6} \wedge \overline{\mathtt{Bad3}} \wedge \overline{\mathtt{Bad5}} | i \neq j] \leq \frac{\binom{q_c}{2}}{2^{2n-1}} \leq \frac{q_c^2}{2^{2n}}\,.$$

Combining the two cases above, we have

$$\Pr[\mathtt{Bad6} \wedge \overline{\mathtt{Bad3}} \wedge \overline{\mathtt{Bad5}}] \leq \frac{2q_c}{2^n} + \frac{q_c^2}{2^{2n}}\,. \tag{5}$$

□ **Bounding** $\mathtt{Bad7}$*:* Due to the uniform distribution of secret keys, we have $\Pr[J = K_i] = \frac{1}{2^n}$. Now, considering the total of q_c construction queries(TGF calls) and q_p primitive queries, we have

$$\Pr[\mathtt{Bad8}] \leq \frac{q_c \cdot q_p}{2^n}. \tag{6}$$

Let $\mathtt{Bad} = \bigcup_{i=1}^{7} \mathtt{Bad}_i$. Then

$$\begin{aligned}\Pr[\mathtt{Bad}] &\leq \frac{u^2}{2^{2n+1}} + \frac{q_p^2}{2^{n+1}} + \frac{uq_p}{2^n} + \frac{uq_d}{2^n} + \frac{2q_c}{2^n} + \frac{q_c^2}{2^{2n}} + \frac{q_c \cdot q_p}{2^n} \\ &\leq \frac{u^2}{2^{2n+1}} + \frac{4q^2}{2^n} + \frac{2q}{2^n} + \frac{q^2}{2^{2n}}\,.\end{aligned} \tag{7}$$

C.2 Bounding the Forging Probability when Bad Does Not Occur

Now, we will bound the forging probability when $\mathtt{Bad}$ does not occur. Let $(\mathcal{D}, i, N_i^a, A_i^a, M_i^a, C_i^a \| \mathsf{T}_i^a, \mathbf{h}_i^a, \mathbf{k}_i^a)$ be a forgery attempt. We calculate the success probability of this forgery to be valid for each of the following cases:

- $\underline{\text{Case A:}}$ $(\mathcal{E}, i, N_i^a, A_i^a, \star, C_i^a \| \mathsf{T}_i^b, \mathbf{h}_i^b, \mathbf{k}_i^b)$ is a previous query for the same user i. In this case, the tweaks used in the TGF for both queries are equal, and hence, by non-triviality of the forgery attempt $\mathsf{T}_i^a \neq \mathsf{T}_i^b$. Hence,

$$\widetilde{E}^{-1}(K_i, twk, \mathsf{T}_i^a) \neq \widetilde{E}^{-1}(K_i, twk, \mathsf{T}_i^b)$$

  Thus, in this case $\mathscr{A}$ will be successful with probability 0.
- $\underline{\text{Case B:}}$ $(\mathcal{D}, i, N_i^b, A_i^b, M_i^b, C_i^a \| \mathsf{T}_i^b, \mathbf{h}_i^b, \mathbf{k}_i^b)$ is a previous query for the same user i. In this case, at least one of T_i^a and T_i^b is invalid. Assuming T_i^b is invalid (otherwise, the success probability is 0), $\widetilde{E}^{-1}(K_i, twk, \mathsf{T}_i^a)$ should be fresh, and $\Pr[\widetilde{E}^{-1}(K_i, twk, \mathsf{T}_i^a) = 0^n] \leq \frac{1}{2^n - q_c} \leq \frac{2}{2^n}$ considering $q_c \leq 2^{n-1}$. So, in this case $\mathscr{A}$ will succeed with probability at most $2/2^n$.

- Case C: If neither of the above two cases happen, then, since Bad does not occur (and hence, no collision in the hash), the key-tweak pair for the TGF i.e., $(K_j, h_{l_a+v_a+1}\|k_{l_a+v_a+1}\|0^{(\frac{dn}{2}-2n)})$ differs from all previous tuples (h,k). Hence, we obtain

$$\Pr[\widetilde{E}^{-1}(K_i, h_{l_a+v_a+1}\|k_{l_a+v_a+1}\|0^{(\frac{dn}{2}-2n)}, Tag_i^a) = 0^n] \leq \frac{1}{2^n - q_c} \leq \frac{2}{2^n}.$$

Combining all cases above, we obtain

$$\Pr[\mathscr{A} \text{ forges} \wedge \overline{\texttt{Bad}}] \leq \sum_{1}^{q_d} \frac{2}{2^n} \leq \frac{2q_d}{2^n} \leq \frac{2q}{2^n}. \tag{8}$$

D Forgery Algorithms for Multiplex

Case A: $\exists\ x, y \in \mathsf{Ker}(f) : g(x) = g(y)$	Case B: $\exists\ x, y \in \mathsf{Ker}(g) : f(x) = f(y)$
1 :	`Choose` $z \in \{0,1\}^{dn} : f(z) \neq 0^l$; `Choose` $z \in \{0,1\}^{dn} : g(z) \neq 0^l$;
2 :	`Make an Encryption Query` $(N, A, M = (M_1\| \cdots \|M_d))$;
3 :	`Let the response be` $(C = C_1\| \cdots \|C_d,\ T)$;
4 :	`Compute` $e_1 = C_1 \oplus M_1, \ldots, e_d = C_d \oplus M_d$;
5 :	`Compute` $M_1'\| \cdots \|M_d' = (x \oplus z) \oplus (e_1\| \cdots \|e_d)$.
6 :	`Make an Encryption Query` $(N, A, M' = (M_1'\| \cdots \|M_d'))$;
7 :	`Let the response be` $(C',\ T')$;
8 :	`Forge` $(N, A, C'' = (y \oplus z),\ T')$;

Fig. 6: Forgery algorithm in Cases A (without) and B (with the boxed statements) for Multiplex.

Case C: Cases A and B do not hold

1 : `Make an Encryption Query` $(N, A, M = (M_1 \| \cdots \| M_d))$;
2 : `Let the resonse be` $(C = (C_1 \| \cdots \| C_d), T)$;
3 : `Note internal round pair` (h_α, k_α), `used to create` $C_1 \| \cdots \| C_d$;
4 : `Choose` $z \in \{0,1\}^{dn} : \ g(z) \neq 0$;
5 : `Let` $a = \min\{2^{2n-l}, 2^{\frac{n}{2}}\}$;
6 : `Choose distinct` $x_1, x_2, \ldots, x_a \leftarrow \mathsf{Ker}(g)$;
7 : `Compute` $k_i \leftarrow \widetilde{E}(k_\alpha, t_1^i, h_\alpha)$, `where` $t_1^i \leftarrow f(x_i \oplus z)$, `for` $i = 1, \ldots, a$;
8 : `Find` (i, j) `such that` $k_i = k_j$;
9 : `Compute` $e_1 := C_1 \oplus M_1, \cdots, \ e_d := C_d \oplus M_d$;
10 : `Compute` $M_1' \| \cdots \| M_d' = \ (x_i \oplus z) \oplus (e_1 \| \cdots \| e_d)$;
11 : `Make an Encryption Query` $(N, A, M' = (M_1' \| \cdots \| M_d'))$;
12 : `Let the response be` (C', T');
13 : `Forge with` $(N, A, C'' = (x_j \oplus z), T')$;

Fig. 7: Forgery algorithm in Case C for Multiplex.

E Details of the muCCAmL1 Security of Tweplex

Before proceeding to the main result, we first define two security notions that will be useful to state the result.

Definition 2. *Let $\widetilde{E} \in \mathsf{TBC}(\{0,1\}^n, \mathcal{T}, \{0,1\}^n)$. Let $\mathcal{L}^{in}$ and $\mathcal{L}^{out}$ be two sets of in- and output leakage, respectively. Let $\mathscr{A}$ be an adversary that provides $k_0, h_0 \in \{0,1\}^n$ and $t_1, t_2, t_1', t_2', t_3' \in \mathcal{T}$ and plays LUP-d game, as given in Algorithm 4, against $\mathcal{F}[\widetilde{E}]$, and outputs a set $\mathbf{K}$. The LUP-d advantage of $\mathscr{A}$ is defined as*

$$\mathbf{Adv}^{\mathsf{LUP\text{-}d}}_{\mathcal{F}[\widetilde{E}]_{k_0,h_0},\mathcal{L}^{in},\mathcal{L}^{out}}(\mathscr{A}) \triangleq \Pr[|\mathbf{K}| \leq q \wedge (h_1, k_1) \in \mathbf{K}] \,.$$

We define $\mathbf{Adv}^{\mathsf{LUP\text{-}d}}_{\mathcal{F}[\widetilde{E}],\mathcal{L}^{in},\mathcal{L}^{out}}(p, q)$ as the maximum of all LUP-d adversaries $\mathscr{A}$ on $\mathcal{F}[\widetilde{E}]$ that ask at most p queries and output a set of at most q elements each.

Definition 3. *Let $\widetilde{E} \in \mathsf{TBC}(\{0,1\}^n, \mathcal{T}, \{0,1\}^n)$. Let $\mathcal{L}^{out}$ and $\mathcal{L}^{\oplus}$ be two sets of leakage. Let $\mathscr{A}$ be an adversary that provides $K, M \in \{0,1\}^n$ and $T \in \mathcal{T}$ and plays XOR$ game as given in Algorithm 5 against $\mathcal{F}[\widetilde{E}]$, and outputs a bit b'. The XOR$ advantage of $\mathscr{A}$ is defined as*

$$\mathbf{Adv}^{\mathsf{XOR\$}}_{\mathcal{F}[\widetilde{E}]_{K,T},\mathcal{L}^{out},\mathcal{L}^{\oplus}}(\mathscr{A}) \triangleq \left|\Pr[\mathscr{A}^1 = 1] - \Pr[\mathscr{A}^0 = 1]\right| \,.$$

Algorithm 4 LUP-d Game.

procedure INITIALIZE($k_0, h_0, t_1, t_2, t'_1, t'_2, t'_3$)
  $k_1 \xleftarrow{\$} \{0,1\}^n$; $h_1 \xleftarrow{\$} \{0,1\}^n$
  $ip_1 \leftarrow \widetilde{E}^{-1}(k_0, t_1, k_1)$
  $ip_2 \leftarrow \widetilde{E}^{-1}(k_0, t_2, h_1)$

function LEAK[$\widetilde{E}$]($\Lambda^{in}, \Lambda^{out}, K, T, X, Y$)
  $R^{in}, R^{out} \xleftarrow{\$} \mathcal{R}$
  $L^{in} \leftarrow \Lambda^{in}(K, T, X : R^{in})$
  $L^{out} \leftarrow \Lambda^{out}(K, T, Y : R^{out})$
  return (L^{in}, L^{out})

function FINALIZE(**K**)
  if $|\mathbf{K}| \leq q \wedge (h_1, k_1) \in \mathbf{K}$ **then**
    return 1
  return 0

function $\mathcal{F}[\widetilde{E}](\Lambda^{in}, \Lambda^{out})$
  for $i \leftarrow 1 \ldots p$ **do**
    $R_1^{in}, R_1^{out} \xleftarrow{\$} \mathcal{R}$
    $k_1 \leftarrow \widetilde{E}[k_0, t_1, ip_1]$
    $\mathfrak{L} \xleftarrow{\cup} \Lambda^{out}(k_0, t_1, k_1 : R_1^{in})$
    $h_1 \leftarrow \widetilde{E}[k_0, t_2, ip_2]$
    $\mathfrak{L} \xleftarrow{\cup} \Lambda^{out}(k_0, t_2, h_1 : R_1^{out})$
    $k_2 \leftarrow \widetilde{E}(k_1, t'_1, h_1)$
    $\mathfrak{L} \xleftarrow{\cup}$ LEAK[$\widetilde{E}$]($\Lambda^{in}, \Lambda^{out}, k_1, t'_1, h_1, k_2$)
    $h_2 \leftarrow \widetilde{E}(k_1, t'_2, h_1 \oplus \theta_1)$
    $\mathfrak{L} \xleftarrow{\cup}$ LEAK[$\widetilde{E}$]($\Lambda^{in}, \Lambda^{out}, k_1, t'_2, h_1 \oplus \theta_1, h_2$)
    for $j \leftarrow 2 \ldots d$ **do**
      $e_j \leftarrow \widetilde{E}(k_1, t'_3, h_1 \oplus \theta_j)$
      $\mathfrak{L} \xleftarrow{\cup}$ LEAK[$\widetilde{E}$]($\Lambda^{in}, \Lambda^{out}, k_1, t'_3, h_1 \oplus \theta_j, e_j$)
  return $(k_2, h_2, e_2, \ldots, e_d, \mathfrak{L})$

Algorithm 5 XOR$ Game.

procedure INITIALIZE($K, T, M, \Lambda^{in}, \Lambda^{out}, \Lambda^{\oplus}$)
  $Y \xleftarrow{\$} \{0,1\}^n$; $b \xleftarrow{\$} \{0,1\}$
  $M^* \leftarrow M$
  if b=0 **then**
    $M^* \xleftarrow{\$} \{0,1\}^n$
  $X \leftarrow \widetilde{E}^{-1}(K, T, Y)$

function FINALIZE(b')
  return $b = b'$

function $\mathcal{F}[\widetilde{E}](\Lambda^{in}, \Lambda^{out}, \Lambda^{\oplus})$
  for $i \leftarrow 1 \ldots p$ **do**
    $R_1^{in}, R_1^{out}, R^{\oplus} \xleftarrow{\$} \mathcal{R}$
    $Y \leftarrow \widetilde{E}(K, T, X)$
    $\mathfrak{L} \xleftarrow{\cup} \Lambda^{out}(K, T, Y : R_1^{out})$
    $C \leftarrow Y \oplus M^*$
    $\mathfrak{L} \xleftarrow{\cup} \Lambda^{\oplus}(Y, C : R^{\oplus})$
  return $(C, \mathfrak{L})$

We define $\mathbf{Adv}^{\mathsf{XOR\$}}_{\mathcal{F}[\widetilde{E}], \mathcal{L}^{out}, \mathcal{L}^{\oplus}}(q)$ *as the maximum of all* XOR$ *adversaries* $\mathscr{A}$ *on* $\mathcal{F}[\widetilde{E}]$ *that ask at most* q *queries.*

Theorem 5. *Let* $\widetilde{E} \in \mathsf{TBC}(\{0,1\}^n, \mathcal{T}, \{0,1\}^n)$. *Let* $\mathscr{A}$ *be a* muCCAmL1 *adversary on* $\Pi[\widetilde{E}]_K = \mathsf{Tweplex}[\widetilde{E}]_K$ *that is allowed to ask at most* q_e *encryption queries of at most* σ *dn-bit blocks and* q *primitive queries in total. Let* $\mathcal{F}[\widetilde{E}]$ *denote an iteration of* Tweplex *for message encryption,* $\mathcal{L}^{in}$, $\mathcal{L}^{out}$, *and* $\mathcal{L}^{\oplus}$ *be leakage functions. Then*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{muCCAmL1}}_{\Pi[\widetilde{E}]}(\mathscr{A}) &\leq \mathbf{Adv}^{\mathsf{muCPAmL1}}_{\Pi[\widetilde{E}]} + \mathbf{Adv}^{\mathsf{muCIML2}}_{\Pi[\widetilde{E}]} \\ &\leq \Pr[\mathsf{Bad}] + 2\sigma \cdot \mathbf{Adv}^{\mathsf{LUP\text{-}d}}_{\mathcal{F}[\widetilde{E}], \mathcal{L}^{in}, \mathcal{L}^{out}}(p, q) + \\ &\quad \sigma \cdot \mathbf{Adv}^{\mathsf{XOR\$}}_{\mathcal{F}[\widetilde{E}], \mathcal{L}^{out}, \mathcal{L}^{\oplus}}(q) + \mathbf{Adv}^{\mathsf{muCIML2}}_{\Pi[\widetilde{E}]}.\end{aligned}$$

Proof. The proof follows similar steps as the qCPAmL2 proof by [31] on TEDT2 and the muCCAmL2 proof on TEDT [8]. However, we consider a multi-user version and assume that the adversary can query its decryption oracle only with

Algorithm 6 Real and Ideal worlds.

function REAL(h_1, k_1, A, M)
  $A_1 \| A_2 \| \cdots \| A_{da} \leftarrow PAD(A)$
  $M_1 \| M_2 \| \cdots \| M_{dm} \leftarrow PAD(M)$
  for $i \leftarrow 2 \cdots a$ **do**
    $t_{i,1} \leftarrow \mathsf{msb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
    $t_{i,2} \leftarrow \mathsf{lsb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
    $k_i \leftarrow \widetilde{E}(k_{i-1}, t_{i,1}, h_{i-1}) \oplus h_{i-1}$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}[\widetilde{E}](k_{i-1}, t_{i,1}, h_{i-1}, k_i)$
    $h_i \leftarrow \widetilde{E}(k_{i-1}, t_{i,2}, h_{i-1} \oplus \theta_1) \oplus h_{i-1} \oplus \theta_1$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{i-1}, t_{i,2}, h_{i-1} \oplus \theta_1, k_i)$
  $X_1 \leftarrow \mathsf{msb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
  $X_2 \leftarrow \mathsf{lsb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
  $X_3 \leftarrow X_1 \oplus X_2$
  for $i \leftarrow 1..m$ **do**
    $k_{a+i} \leftarrow \widetilde{E}(k_{a+i-1}, X_1, h_{a+i-1}) \oplus h_{a+i-1}$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{a+i-1}, X_1, h_{a+i-1}, k_{a+i})$
    $h_{a+i} \leftarrow \widetilde{E}(k_{a+i-1}, X_2, h_{a+i-1} \oplus \theta_1) \oplus h_{a+i-1} \oplus \theta_1$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{a+i-1}, X_2, h_{a+i-1} \oplus \theta_1, h_{a+i})$
    $C_{d(i-1)+1} \leftarrow M_{d(i-1)+1} + h_{a+i}$
    $\mathfrak{L} \leftarrow \mathsf{LEAK\text{-}XOR}(C_{d(i-1)+1}, M_{d(i-1)}, h_{a+i})$
    for $j \leftarrow 2 \cdots d$ **do**
      $e_j \leftarrow \widetilde{E}(k_{a+i-1}, X_3, h_{a+i-1} \oplus \theta_j) \oplus h_{a+i-1} \oplus \theta_j$
      $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+i-1}, X_3, h_{a+i-1} \oplus \theta_j, e_j)$
      $C_{d(i-1)+j} \leftarrow M_{d(i-1)+j} + e_j$
      $\mathfrak{L} \leftarrow \mathsf{LEAK\text{-}XOR}(C_{d(i-1)+j}, M_{d(i-1)+j}, e_j)$
    $X_1 \leftarrow \mathsf{msb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
    $X_2 \leftarrow \mathsf{lsb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
    $X_3 \leftarrow X_1 \oplus X_2$
  $twk_1 \leftarrow \widetilde{E}(k_{a+m}, X_1, h_{a+m}) \oplus h_{a+m}$
  $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+m}, X_1, h_{a+m}, twk_1)$
  $twk_2 \leftarrow \widetilde{E}(k_{a+m}, X_2, h_{a+m} \oplus \theta_1) \oplus h_{a+m+1} \oplus \theta_1$
  $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+m}, X_1, h_{a+m} \oplus \theta_1, twk_2)$
  $twk \leftarrow twk_1 \| twk_2$
  $C = C_1 \| C_2 \| C_3 \| \cdots \| C_{dm}$
  return $(C, twk, \mathfrak{L})$

function LEAK[$\widetilde{E}$](K, T, X, Y)
  $R^{in}, R^{out} \xleftarrow{\$} \mathcal{R}$
  $L^{in} \leftarrow \Lambda^{in}(K, T, X : R^{in})$
  $L^{out} \leftarrow \Lambda^{out}(K, T, Y : R^{out})$
  return (L^{in}, L^{out})

function IDEAL(h_1, k_1, A, M)
  $A_1 \| A_2 \| \cdots \| A_{da} \leftarrow PAD(A)$
  $M_1 \| M_2 \| \cdots \| M_{dm} \leftarrow PAD(M)$
  for $i \leftarrow 2 \cdots a$ **do**
    $t_{i,1} \leftarrow \mathsf{msb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
    $t_{i,2} \leftarrow \mathsf{lsb}_{d'}(A_{d(i-2)+1} \| A_{d(i-2)+2} \| \cdots \| A_{d(i-2)+d})$
    $k_i \xleftarrow{\$} \{0,1\}^n$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}[\widetilde{E}](k_{i-1}, t_{i,1}, h_{i-1}, k_i)$
    $h_i \xleftarrow{\$} \{0,1\}^n$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{i-1}, t_{i,2}, h_{i-1} \oplus \theta_1, k_i)$
  $X_1 \leftarrow \mathsf{msb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
  $X_2 \leftarrow \mathsf{lsb}_{d'}(A_{d(a-1)+1} \| A_{d(a-1)+2} \| \cdots \| A_{d(a-1)+d})$
  $X_3 \leftarrow X_1 \oplus X_2$
  for $i \leftarrow 1..m$ **do**
    $k_{a+i} \xleftarrow{\$} \{0,1\}^n$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{a+i-1}, X_1, h_{a+i-1}, k_{a+i})$
    $h_{a+i} \xleftarrow{\$} \{0,1\}^n$
    $\mathfrak{L} \xleftarrow{\cup} \mathsf{LEAK}(k_{a+i-1}, X_2, h_{a+i-1} \oplus \theta_1, h_{a+i})$
    $C_{d(i-1)+1} \leftarrow M_{d(i-1)+1} + h_{a+i}$
    $\mathfrak{L} \leftarrow \mathsf{LEAK\text{-}XOR}(C_{d(i-1)+1}, M_{d(i-1)}, h_{a+i})$
    for $j \leftarrow 2 \cdots d$ **do**
      $e_j \xleftarrow{\$} \{0,1\}^n$
      $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+i-1}, X_3, h_{a+i-1} \oplus \theta_j, e_j)$
      $C_{d(i-1)+j} \leftarrow M_{d(i-1)+j} + e_j$
      $\mathfrak{L} \leftarrow \mathsf{LEAK\text{-}XOR}(C_{d(i-1)+j}, M_{d(i-1)+j}, e_j)$
    $X_1 \leftarrow \mathsf{msb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
    $X_2 \leftarrow \mathsf{lsb}_{d'}(C_{d(i-1)+1} \| C_{d(i-1)+2} \| \cdots \| C_{d(i-1)+d})$
    $X_3 \leftarrow X_1 \oplus X_2$
  $twk_1 \xleftarrow{\$} \{0,1\}^n$
  $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+m}, X_1, h_{a+m}, twk_1)$
  $twk_2 \xleftarrow{\$} \{0,1\}^n$
  $\mathfrak{L} \leftarrow \mathsf{LEAK}(k_{a+m}, X_1, h_{a+m} \oplus \theta_1, twk_2)$
  $twk \leftarrow twk_1 \| twk_2$
  $C = C_1 \| C_2 \| C_3 \| \cdots \| C_{dm}$
  return $(C, twk, \mathfrak{L})$

function LEAK-XOR(C, M, e)
  for $i \leftarrow 1 \ldots p$ **do**
    $R^{\oplus} \xleftarrow{\$} \mathcal{R}$
    $C \leftarrow M \oplus e$
    $\mathfrak{L} \xleftarrow{\cup} \Lambda^{\oplus}(e, C : R^{\oplus})$
  return $\mathfrak{L}$)

some previous outputs from the encryption oracle due to the muCIML2 assumption. Moreover, we assume that the decryption oracle does not give any leakage. In the following, we will define and use a sequence of four games $\mathscr{G}_1$, $\mathscr{G}_2$, $\mathscr{G}_3$, and $\mathscr{G}_4$. $\mathscr{G}_1$ represents the left-hand side and $\mathscr{G}_4$ the right-hand side of Equation (9):

$$\triangle_{\mathscr{A}}(L\mathcal{E}_1^K, L\mathcal{E}_2^K, \mathcal{D}_K, \widetilde{E}, \widetilde{E}^{-1}, \mathcal{L}_{\mathcal{E}};\ L\mathcal{E}_1^K, L\hat{\$}_{\mathcal{E}}^K, \hat{\$}_{\mathcal{D}}^K, \widetilde{E}, \widetilde{E}^{-1}, \mathcal{L}_{\mathcal{E}})\,. \tag{9}$$

We will move stepwise from game $\mathscr{G}_1$ to $\mathscr{G}_4$ by defining adversaries $\mathscr{A}_{i,i+1}$, for $i = 1, 2, 3$ that aim at distinguishing between $\mathscr{G}_i$ and $\mathscr{G}_{i+1}$. We will write $\triangle\mathscr{G}_{i,j}$ for the maximal advantage over all adversaries $\mathscr{A}_{i,j}$ distinguishing between game $\mathscr{G}_i$ and $\mathscr{G}_j$.
We introduce $\mathsf{Real}[\widetilde{E}]$ and $\mathsf{Ideal}[\widetilde{E}]$ as defined in algorithm 6, where $\mathsf{Real}[\widetilde{E}]$ corresponds to message encryption using original Tweplex and $\mathsf{Ideal}[\widetilde{E}]$ corresponds to message encryption with ideal(random) oracle. We denote original message as M and random message as \$. four games are as follows:

- $\mathscr{G}_1 := \mathsf{Real}[\widetilde{E}](M)$,
- $\mathscr{G}_2 := \mathsf{Ideal}[\widetilde{E}](M)$,
- $\mathscr{G}_3 := \mathsf{Ideal}[\widetilde{E}](\$)$, and
- $\mathscr{G}_4 := \mathsf{Real}[\widetilde{E}](\$)$.

From the muCPAmL1 security definition and the triangle inequality, we have

$$\mathbf{Adv}_{\Pi}^{mu\mathsf{CPAmL1}}(\mathscr{A}) = \Delta\mathscr{G}_{1,4} \leq \sum_{i=1}^{3} \Delta\mathscr{G}_{i,i+1}\,. \tag{10}$$

Difference $\Delta\mathscr{G}_{1,2}$: We define the ideal world Ideal. It samples all the internal successive chaining values h, successive keys k, and values e's uniformly and independently from $\{0,1\}^n$. Then, it creates the ciphertext by XORing the message with the respective chosen e values. It also computes leakage outputs $\mathcal{L}^{in}$, $\mathcal{L}^{out}$, and $\mathcal{L}^{\oplus}$ with its primitive oracles $\widetilde{E}$ and $\widetilde{E}^{-1}$ and collects it as the would-be leakage in a vector $\mathfrak{L}$ and outputs the ciphertext C together with $\mathfrak{L}$.
From our definition of Bad events in Section 6 and our assumption $\overline{\mathsf{Bad}}$, it holds that each primitive query is fresh during the challenge query. This guarantees that there is no difference between the worlds Real and Ideal when leakage is absent. However, under leakage and in absence of Bad, there can be a distinguishing advantage. Though, the distinguishing advantage of $\mathscr{A}_{1,2}$ can be reduced to that of an LUP-d adversary $\mathscr{A}_{up}$ on an isolated iteration of Tweplex. We can define $\mathscr{A}_{up}$, that runs an instance of $\mathscr{A}_{1,2}$ and simulates its oracle as follows:

- For a primitive query of $\mathscr{A}_{1,2}$, $\mathscr{A}_{up}$ will query the same to it's own primitive oracle and return the result. Moreover it will store the query in transcript τ_p as (k, J, X, Y).
- For a construction query (i, N_i^a, A_i^a, M_i^a) to the user i. Where $M_1^a \leftarrow m_1 \| m_2 \| \cdots \| m_{l_m}$ and $A_1^a \leftarrow A_1 \| A_2 \| \cdots A_{l_a}$ where each A_j and m_j's are dn-bit block, $\mathscr{A}_{up}$ will reply as follows:

- $\mathscr{A}_{up}$ samples $s \xleftarrow{\$} [l_m]$, $h_1, k_1 \xleftarrow{\$} \{0,1\}^n$. Moreover, it initializes an empty list of leakages $\mathfrak{L}$.
- For each dn-bit block of associated data, $\mathscr{A}_{up}$ computes two tweaks $t_1 \leftarrow \mathsf{msb}_{dn/2}(A_j)$ and $t_2 \leftarrow \mathsf{lsb}_{dn/2}(A_j)$, $\forall j = 1, 2, \ldots, l_a - 1$. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.
- It computes k_{j+1} by querying its own primitive (k_j, t_1, h_j) and computes k_{j+1} by querying with $(k_j, t_2, j_j + \theta_1)$. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.
- For A_{l_a}, $\mathscr{A}_{up}$ computes $t_1 \leftarrow \mathsf{msb}_{dn/2}(A_j)$ and $t_2 \leftarrow \mathsf{lsb}_{dn/2}(A_j)$ and $t_3 \leftarrow t_1 \oplus t_2$. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.
- For $j = 1 \ldots s-1$, $\mathscr{A}_{up}$ queries to its primitive to get $k_{l_a+j}, h_{l_a+j}, e^2_{l_a+j} \| \cdots \| e^d_{l_a+j}$. $\mathscr{A}_{up}$ obtains $C^1_j \leftarrow m^1_j \oplus h_{l_a+j}, C^x_j \leftarrow m^x_j \oplus e^x_{l_a+j}, \forall x = 2 \ldots d$. Where m^x_j is the x-th d-bit block of m_j same for c. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.
- For $j = s$, $\mathscr{A}_{up}$ queries its primitive queries to get k_{l_a+s} and $\{h_{l_a+s}, e^2_{l_a+s}, \ldots, e^d_{l_a+s}\}$. It computes C^y_s, $\forall y = 1, 2, \ldots, d$ as before. Moreover, $\mathscr{A}_{up}$ queries its $\mathsf{LUP\text{-}d}$ challenger with $(k_{l_a+s-1}, h_{l_a+s-1}, t_1, t_2, t'_1, t'_2, t'_3)$ and receives $(k_{l_a+s+1}, h_{l_a+s+1}, e^2_{l_a+s+1}, e^3_{l_a+s+1}, \ldots, e^d_{l_a+s+1})$. Then, it computes C_{l_a+s+1} accordingly. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.
- For $j = s+2 \ldots l_m$, $\mathscr{A}_{up}$ queries its primitive oracle to compute k_{l_a+j}, h_{l_a+j}, and $e^2_{l_a+j}$, $\ldots$, $e^d_{l_a+j}$. It uses those to compute the respective ciphertexts. Moreover, $\mathscr{A}_{up}$ adds the leakage to $\mathfrak{L}$.

- It returns the ciphertext C^a_i along with the leakage $\mathfrak{L}$ to $\mathscr{A}_{1,2}$.

At the end of the interaction, $\mathscr{A}_{up}$ outputs a set of h, k values consists of all key input pair in the primitive querues of $\mathscr{A}_{1,2}$, $\mathbf{K} = \{(k,h) : (k, *, h, *)\tau_p\}$.
Clearly, the advantage of $\mathscr{A}_{1,2}$ is inherited by $\mathscr{A}_{up}$. Moreover, the difference also consists of the right guess of the index s, for which number of possible values are maximum σ. Thus, the advantage of $\mathscr{A}_{1,2}$ is upper bounded as follows:

$$\Delta\mathscr{G}_{1,2} \leq \Pr[\mathsf{Bad}] + \sigma \cdot \mathbf{Adv}^{\mathsf{LUP\text{-}d}}_{\mathcal{F}[\tilde{E}], \mathcal{L}^{in}, \mathcal{L}^{out}}(p, q)\,. \tag{11}$$

Difference $\Delta\mathscr{G}_{2,3}$: Since the ideal world Ideal outputs a random key string, in the black-box setting, its output will be indistinguishable from random. But under leakage, this may differ. However, we can reduce that distinguishing advantage to that of an $\mathsf{XOR\$}$ adversary. For this purpose, we will follow a similar approach as in the previous case. First, we define an adversary $\mathscr{A}_{\oplus}$ for the $\mathsf{XOR\$}$ game, which will then simulate the challenger of the distinguisher $\mathscr{A}_{2,3}$ as follows:

- For a primitive query by $\mathscr{A}_{2,3}$, $\mathscr{A}_{\oplus}$ queries its own primitive oracle and relays the response. Moreover, it stores the query in τ_p as (k, J, X, Y).
- For a construction query (i, N^a_i, A^a_i, M^a_i) to the user i, where $M^a_1 \leftarrow (m_1 \| m_2 \| \cdots \| m_{l_m})$ and $A^a_1 \leftarrow A_1 \| A_2 \| \cdots A_{l_a}$ where each A_j and m_j's are dn-bit block, $\mathscr{A}_{\oplus}$ will reply as follows:
  - $\mathscr{A}_{\oplus}$ samples $s \xleftarrow{\$} [l_m]$. Moreover it initializes an empty list of leakage $\mathfrak{L}$.
  - For $j = 1, \ldots, l_a$, it chooses $h_j, k_j \xleftarrow{\$} \{0,1\}^n$.

- For $j = 1 \dots s-1$, $\mathscr{A}_{\oplus}$, it chooses randomly k_{l_a+j}, h_{l_a+j}, $(e^2_{l_a+j} \| \cdots \| e^d_{l_a+j})$ from $\{0,1\}^n$. $\mathscr{A}_{\oplus}$ obtains $C^1_j \leftarrow m^1_j \oplus h_{l_a+j}, C^x_j \leftarrow m^x_j \oplus e^x_{l_a+j}, \forall x = 2 \dots d$, where m^x_j is the x-th d-bit block of m_j same for c. Moreover, $\mathscr{A}_{\oplus}$ adds the leakage obtained from its primitive oracle as defined in Algorithm 6 to the list $\mathfrak{L}$.
- For $j = s$, $\mathscr{A}_{\oplus}$, it chooses $k_{l_a+s} \xleftarrow{\$}$. It also chooses $\{h_{l_a+s} \xleftarrow{\$} \{0,1\}^n$. $\mathscr{A}_{\oplus}$ get C^1_s by quering the $\mathsf{XOR\$}$ challenger with $(k_{l_a+s-1}, \mathsf{lsb}_{dn/2}(C_{s-1}), m^1_s)$. $\mathscr{A}_{\oplus}$ also computes $t = \mathsf{lsb}_{dn/2}(C_{s-1}) \oplus \mathsf{msb}_{dn/2}(C_{s-1})$. $\mathscr{A}_{\oplus}$ will obtain C^x_s, $\forall x = 2, \dots, d$ by querying its $\mathsf{XOR\$}$ oracle by $(k_{l_a+s-1,}, t)$ and repeating it $d-1$ times. Moreover, $\mathscr{A}_{\oplus}$ adds the leakage obtained from its primitive oracle as defined in Algorithm 6 to the list $\mathfrak{L}$.
- For $j = s+2 \dots l_m$, $\mathscr{A}_{\oplus}$ will follow the same procedure as the case $1, \dots, s-1$. Moreover, $\mathscr{A}_{\oplus}$ adds the leakage obtained from its primitive oracle as defined in Algorithm 6 to the list $\mathfrak{L}$.

- It returns the ciphertext C^a_i along with the leakage $\mathfrak{L}$ to $\mathscr{A}_{2,3}$.

At the end of its interaction, $\mathscr{A}_{\oplus}$ will relay the response of $\mathscr{A}_{2,3}$ to its own challenger. Then, the advantage of $\mathscr{A}_{\oplus}$ is inherited by $\mathscr{A}_{2,3}$. It also consists of the right guess of the iteration s among the at most σ possible options. Thus,

$$\Delta\mathscr{G}_{2,3} \leq \sigma \cdot \mathbf{Adv}^{\mathsf{XOR\$}}_{\mathcal{F}[\widetilde{E}],\mathcal{L}^{out},\mathcal{L}^{\oplus}}(q)\,. \tag{12}$$

Difference $\Delta\mathscr{G}_{3,4}$: It is easy to see that this difference is equivalent to the difference $\Delta\mathscr{G}_{1,2}$.
Our claim in Theorem 5 follows from summing up the individual bounds from Equations (10), (11) and twice that of Equation (12).