IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 2, 34 pages.

https://doi.org/10.62056/akgyl86bm

FEDT: Forkcipher-based Leakage-resilient Beyond-birthday-secure AE

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

[1] Institute for Advancing Intelligence, TCG CREST, Kolkata, India
[2] Independent Researcher, Singapore, Singapore
[3] Ramakrishna Mission Vivekananda Educational and Research Institute, Kolkata, India

Abstract. There has been a notable surge of research on leakage-resilient authenticated encryption (AE) schemes, in the bounded as well as the unbounded leakage model. The latter has garnered significant attention due to its detailed and practical orientation. Designers have commonly utilized (tweakable) block ciphers, exemplified by the TEDT scheme, achieving $\mathcal{O}(n - \log(n^2))$-bit integrity under leakage and comparable AE security in the black-box setting. However, the privacy of TEDT was limited by $n/2$-bits under leakage; TEDT2 sought to overcome these limitations by achieving improved security with $\mathcal{O}(n - \log n)$-bit integrity and privacy under leakage.

This work introduces FEDT, an efficient leakage-resilient authenticated encryption (AE) scheme based on fork-cipher. Compared to the state-of-the-art schemes TEDT and TEDT2, which process messages with a rate of 1/2 block per primitive call for encryption and one for authentication, FEDT doubles their rates at the price of a different primitive. FEDT employs a more parallelizable tree-based encryption compared to its predecessors while maintaining $\mathcal{O}(n - \log n)$-bit security for both privacy and integrity under leakage. FEDT prioritizes high throughput at the cost of increased latency. For settings where latency is important, we propose FEDT*, which combines the authentication part of FEDT with a CTR-based encryption. FEDT* offers security equivalent to FEDT while increasing the encryption rate of 4/3 and reducing the latency.

1 Introduction

The study of authenticated encryption (AE) schemes, designed to safeguard both message confidentiality and ciphertext integrity, has been a dynamically evolving research domain since its conceptualization as a cryptographic primitive [BN00, Rog02]. Over the years, various variants such as online schemes, nonce-based, and deterministic authenticated encryption [BBKN01, HRRV15, RS06] have emerged. This proliferation of designs aims to strike a balance between efficiency and security, with recent contributions from competitions like CAESAR [Ber14] and the NIST Lightweight Competition [TMC+23] further enriching the landscape. In addition to these general considerations, a significant body of research has focused on enhancing the robustness of AE schemes. This includes addressing challenges like security under nonce-misuse [RS06], mitigating the impact of accidental nonce repetitions [DNT19], and addressing scenarios where unverified plaintexts might be released [ABL+14, CDD+19]. Another very important practical challenge in this direction is to have designs with protection against side channel attacks.

E-mail: nilanjan.datta@tcgcrest.org (Nilanjan Datta), avijit.dutta@tcgcrest.org (Avijit Dutta), elist@posteo.de (Eik List), sougata.mandal@tcgcrest.com (Sougata Mandal)

Received: 2024-04-09 Accepted: 2024-06-03

The conventional theoretical security considerations for Authenticated Encryption (AE) treat cryptographic primitives as black boxes, not accounting for real-world threats where adversaries exploit additional information from side channels. Such leaks, including timing, power consumption, or electromagnetic radiation, can reveal internal states and keys. Power-consumption attacks, such as Differential Power Analysis (DPA) [KJJ99], expose vulnerabilities in widespread AEAD schemes such as OCB [RBBK01, KR16], GCM [MV04, Dwo07], or CCM [Dwo04], which invoke a block cipher multiple times with a single key. Typically, the responsibility of protecting the underlying block cipher against leakage falls on implementors. Hardware-level measures involve introducing noise or specialized circuits, while at the implementation level, techniques like masking [CJRR99, GP99] or shuffling [HOM06, VMKS12] are employed. However, robust protection often incurs significant area, power, or efficiency penalties, substantially impacting performance in both software and hardware implementations (e.g. [GSF13]).

To address this, research has explored more efficient schemes with trade-offs between security and performance. Instead of uniformly applying strong protection to all block-cipher invocations in an AEAD scheme, leakage-resistant modes of operation have emerged [BMOS17, BKP+18, BPPS17, DEM+17, BPS19]. These modes support dedicated, leveled implementations, where certain calls to the cryptographic primitive receive strong protection against DPA attacks, while others, which constitute the majority of computations, are allowed to leak information. In essence, leakage-resilient AE schemes prioritize security despite potential leaks, striking a balance between efficiency and protection in practical scenarios.

1.1 Leakage-resilient Authenticated Ciphers

A portfolio of leakage-resilient schemes for leveled implementations has been developed in the past few years. We consider the notions for leakage-resilient authenticated encryption by Guo et al. [GPPS19, BKP+18, BPPS17]. In [GPPS19], 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 [BKP+18, BPPS17]; (2) chosen-ciphertext security with misuse-resilience and Leakage in encryption and decryption oracle, called CCAmL2 [GPPS19], along with a multi-user multi-challenge variant [GPPS18]. Bellizia et al. [BBC+20] referred to leveled designs achieving both CIML2 and CCAmL2 security as Grade-3 protected. Grade-3 protected authenticated ciphers include tweakable block cipher based constructions such as TEDT [BGP+20], TEDT2 [Lis21] and permutation-based designs like ISAP [DEM+17, DEM+20]. However, all these constructions are two-pass modes of operation. As a result, it seems interesting to investigate for single-pass Grade-3 designs. 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 only, which was formulated as CCAmL1 [GPPS19]. Following [GPPS19], the designs achieving CIML2 and CCAmL1 security are called Grade-2 protected. Grade-2-protected designs include tweakable block cipher based designs AEDT [BPPS17], Triplex [SPS+22], Multiplex [SPS24], and Tweplex [DDLM23].

Grade-3-protected Leakage-resilient Authenticated Ciphers. In this paper we focus on efficient Grade-3 protected designs. Most of these constructions are (1) based on tweakable block cipher and (2) share a common structure. First, they use a *key-derivation function* (KDF) that employs a highly-protected implementation of the primitive to derive a session state from the nonce and the long-term secret key (master key, hereafter). Then, the plaintext is encrypted using the session state using less protected implementations of the primitive. The encryption function adopts the idea of continuous rekeying. It ensures that the security does not degrade too much even if the underlying less protected primitive calls leak continuously. After encryption, a hash function is applied on the ciphertext, the nonce, and associated data again using a less protected implementation of the primitive to generate a forward chaining value. Finally, this value is input to a *tag-generation function* (TGF) that employs a highly protected implementation and the master key to generate the authentication tag.

In [BPPS17], Berti et al. introduced the design structure of Encrypt, Digest and Tag, dubbed EDT. To address the limitations of EDT, in [BGP+20], Berti et al. proposed a tweakable block cipher based EDT construction, dubbed TEDT. TEDT uses two primitive calls per message block: one to derive a new subkey to process the subsequent block and one to generate the keystream block which is added to the current message block to produce the current ciphertext block. The resulting rate-$1/2$[1] encryption provides $n/2$-bit security under leakage. TEDT employs Hash-then-TBC framework with Hirose's double-block hash function [Hir06] to obtain beyond-birthday-secure authentication even in the presence of unbounded leakage. TEDT2 [Lis21] uses a tweakable block cipher with a $3n$-bit tweakey efficiently to strengthen the security of the encryption to obtain beyond-birthday-bound security also under leakage. The basic structural difference between TEDT and TEDT2 lies in the fact that TEDT2 replaced TEDT's previous internal Hirose's double-block hash function with Naito's MDPH [Nai19], and it moved the nonce from the hash to the tag-generation function both for better efficiency. TEDT2 processes a $2n$-bit message block with each iteration of two primitive calls. Thus, the hash-function rate increases from $1/2$ to 1. Using a TBC with $3n$-bit tweakey, they could spare to process the nonce during hashing and use it in the tag-generation function (TGF) instead. Using a TBC based on the TWEAKEY framework, TEDT2 obtain a longer tweakey for higher security, whose encryption needs two primitive calls for the tweakey update per message block. To compensate for the additional call, it uses the tweakey for processing two message blocks. They obtain a more secure rate-$1/2$ construction that provides $O(n - \log(n))$-bit security in both the black-box setting and under leakage, where they adopted from TEDT the assumption that the distinguishing advantage for the XORs of the plain/ciphertexts with the PRG keystream does not endanger the security.

1.2 Our Contribution

In this paper, we introduce a leakage-resilient authenticated encryption scheme, namely FEDT, based on fork-ciphers [ALP+19]. FEDT employs a balanced tree-structured encryption and a fork-cipher-based hash function. For processing a message of mn bits along with associated data of an bits, the encryption function of FEDT needs m and the authentication component requires $(a + m)/2 + 1$ primitive calls. In terms of security, FEDT achieves nearly optimal security bounds of $O(n - \log(n))$ bits, both in the black-box setting and under leakage. FEDT is well-suited for applications with a considerable - but limited - message length that benefit from parallelizability, such as disk encryption or memory encryption, where frame lengths are typically of 512 bytes or 4 kB. There, leakage resistance is relevant while the security goals demand offline ciphers. On the downside, our proposal induces a non-trivial latency of roughly $\log_2(d)$ fork-cipher calls for a dn-bit message. As part of a remedy, we propose a variant of FEDT with lower latency, dubbed FEDT*, which employs an unbalanced tree-like encryption where each level generates two keys and processes four message blocks. The encryption function of FEDT* employs $3m/4$ primitive calls to process a mn-bit message at the same security level as FEDT. Table 1 compares its parameters with that of similar Grade-3 designs.

[1]The rate of a construction is the ratio of number of message blocks to the number of primitive calls.

Table 1: Comparison of leakage-resilient Grade-3 constructions. TBC/TFC = tweakable block cipher/tweakable fork-cipher.

	Security (bits)		#Primitive calls				
Construction	CIML2	CCAmL2	Prim.	Enc	Hash	KDF	TGF
Romulus-LR-TEDT [GKP20]	$n - \log(n^2)$	$n/2$	TBC	$2m$	$a + m + 3$	1	1
TEDT [BGP+20]	$n - \log(n^2)$	$n/2$	TBC	$2m$	$2(a + m + 2)$	1	1
TEDT2 [Lis21]	$n - \log(n)$	$n - \log(n)$	TBC	$2m$	$a + m + 2$	2	1
FEDT [This work]	$n - \log(n)$	$n - \log(n)$	TFC	m	$(a + m)/2 + 1$	1	1
FEDT* [This work]	$n - \log(n)$	$n - \log(n)$	TFC	$3m/4$	$(a + m)/2 + 1$	1	1

2 Preliminaries

An adversary $\mathcal{A}$ is an algorithm and the notation $y \leftarrow \mathcal{A}(x_1, x_2, \ldots, x_i; r)$ denotes running the algorithm $\mathcal{A}$ with randomness r on inputs $x_1, \ldots, x_i$ and assigning the output to y. Equivalently, we can express the notation above as follows: let $y \overset{\$}{\leftarrow} \mathcal{A}(x_1, \ldots; r)$ be the result of picking r uniformly at random, computing $\mathcal{A}(x_1, \ldots; r)$, and assigning the result to y. For a set $\mathcal{X}$, the notation $\mathcal{X} \overset{\cup}{\leftarrow} x$ denotes $\mathcal{X} = \mathcal{X} \cup \{x\}$. For bit strings x and y, $x\|y$ denotes the concatenation of x and y. We denote by $[X]_x$ the encoding of a non-negative integer $X < 2^x$ as its x-bit representation. For an algorithm $\mathcal{A}$ and an oracle $\mathcal{O}$, we write $\mathcal{A}^{\mathcal{O}}$ to denote the output of $\mathcal{A}$ at the end of its interaction with $\mathcal{O}$. A distinguisher $\mathcal{A}$ is an adversary 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 interaction, it returns a bit $b \in \{0, 1\}$. We write $\mathcal{A}^{\mathcal{O}} = b$ to denote the output of $\mathcal{A}$ at the end of its interaction with $\mathcal{O}$. The distinguishing advantage of $\mathcal{A}$ against $\mathcal{O}_0$ and $\mathcal{O}_1$ is defined as

$$\Delta_{\mathcal{A}}(\mathcal{O}_1; \mathcal{O}_0) \overset{\Delta}{=} \left|\Pr[\mathcal{A}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_0} = 1]\right| ,$$

where the probabilities depend on the random coins of $\mathcal{O}_0$ and $\mathcal{O}_1$ and the random coins of the distinguisher $\mathcal{A}$. The time complexity of the adversary is defined over the usual RAM (random-access machine, see e.g. [Ost90]) model of computations.

2.1 Fork-ciphers

A fork-cipher (FC) [ALP+19] $\widetilde{\mathsf{F}} : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \to \{0, 1\}^{2n}$ is a family of tweakable keyed functions, with an associated key space $\mathcal{K}$ and tweak space $\mathcal{T}$, comprised of a pair of deterministic algorithms $(\widetilde{\mathsf{F}}^+, \widetilde{\mathsf{F}}^-)$, where the encryption algorithm

$$\widetilde{\mathsf{F}}^+ : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \times \{0, 1, 2\} \longrightarrow \{0, 1\}^n \cup (\{0, 1\}^n \times \{0, 1\}^n)$$

takes as inputs a key $k \in \mathcal{K}$, a tweak $J \in \mathcal{T}$, a message $m \in \{0, 1\}^n$, and a selector bit $s \in \{0, 1, 2\}$, and produces an output as

$$\widetilde{\mathsf{F}}^+(k, J, m, s) = \begin{cases} c_0, & \text{if } s = 0 \\ c_1, & \text{if } s = 1 \\ (c_0, c_1), & \text{if } s = 2 \end{cases}$$

where the ciphertext is $c = (c_0, c_1)$. We call c_0 the left and c_1 the right ciphertext block, respectively. The decryption algorithm

$$\widetilde{\mathsf{F}}^- : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \times \{0, 1\} \times \{0, 1, 2\} \longrightarrow \{0, 1\}^n \cup (\{0, 1\}^n \times \{0, 1\}^n)$$

takes as input a key $k \in \mathcal{K}$, a tweak $J \in \mathcal{T}$, a ciphertext block c_b, a bit $b \in \{0, 1\}$ specifying whether c_b is the left or the right ciphertext block, and a selector bit $s \in \{0, 1, 2\}$ and

Algorithm 1 STFP Game.

```
Real world Real World
    function Initialize
        $K \xleftarrow{\$} \mathcal{K}$
    function Finalize
        $b' \in \{0,1\} \leftarrow \mathcal{A}^{\widetilde{\mathsf{F}}^+_K, \widetilde{\mathsf{F}}^-_K}$
    function Oracle $\widetilde{\mathsf{F}}^+_K(J, x, s)$
        return $\widetilde{\mathsf{F}}^+(K, J, x, s)$
    function Oracle $\widetilde{\mathsf{F}}^-_K(J, x, b, s))$
        return $\widetilde{\mathsf{F}}^-(K, J, x, b, s)$

Ideal world
    function Initialize
        $\widetilde{P}_0, \widetilde{P}_1 \xleftarrow{\$} \mathsf{TP}(\mathcal{T}, n)$
    function Finalize
        $b' \in \{0,1\} \leftarrow \mathcal{A}^{\$^+, \$^-}$
    function Oracle $\$^+(J, x, s)$
        return $\begin{cases} \widetilde{P}_0(J,x) & \text{if } s = 0 \\ \widetilde{P}_1(J,x) & \text{if } s = 1 \\ (\widetilde{P}_0(J,x), \widetilde{P}_1(J,x)) & \text{if } s = 2 \end{cases}$
    function Oracle $\$^-(J, y, b, s)$
        return $\begin{cases} \widetilde{P}_b^{-1}(J,y) & \text{if } s = 0 \\ \widetilde{P}_{1\oplus b}(J, \widetilde{P}_b^{-1}(J,y)) & \text{if } s = 1 \\ ((\widetilde{P}_b^{-1}(J,y), \widetilde{P}_{1\oplus b}(J, \widetilde{P}_b^{-1}(J,y)) & \text{if } s = 2 \end{cases}$
```

outputs

$$\widetilde{\mathsf{F}}^-(k, J, c_b, b, s) = \begin{cases} m & \text{if } s = 0 \\ c_{1-b}, & \text{if } s = 1 \\ (m, c_{1-b}), & \text{if } s = 2 \end{cases}$$

where m is the plaintext block. The correctness condition of a fork-cipher states that for every key $k \in \mathcal{K}$, tweak $J \in \mathcal{T}$, plaintext $m \in \{0,1\}^n$, and $b \in \{0,1\}$, the following conditions should hold:

1. $\widetilde{\mathsf{F}}^-(k, J, \widetilde{\mathsf{F}}^+(k, J, m, b), b, 0) = m$,
2. $\widetilde{\mathsf{F}}^-(k, J, \widetilde{\mathsf{F}}^+(k, J, m, b), b, 1) = \widetilde{\mathsf{F}}^+(k, J, m, 1-b)$,
3. $\widetilde{\mathsf{F}}^-(k, J, x, b, 2) = (\widetilde{\mathsf{F}}^-(k, J, x, b, 0), \widetilde{\mathsf{F}}^-(k, J, x, b, 1))$, and
4. $\widetilde{\mathsf{F}}^+(k, J, x, 2) = (\widetilde{\mathsf{F}}^+(k, J, x, 0), \widetilde{\mathsf{F}}^+(k, J, x, 1))$.

For nonempty sets $\mathcal{K}$, $\mathcal{T}$, $\mathcal{B}$, we define $\mathsf{TFC}(\mathcal{K}, \mathcal{T}, \mathcal{B} \cup \mathcal{B}^2)$ as the set of all tweakable fork-ciphers with key space $\mathcal{K}$, tweak space $\mathcal{T}$, and input space $\mathcal{B}$.

Tweakable Fork-permutation Notion. The security of a tweakable fork-cipher is evaluated against an ideal forked permutation in the Tweakable Fork-permutation (TFP) notion with encryption queries only or the Strong Tweakable Fork-Permutation (STFP) notion. Algorithm 1 shows the strong STFP notion. The ideal world provides access to the same interfaces as a real fork-cipher, consisting of encryption and decryption oracles $\$^+$ and $\$^-$. Both use two random tweakable permutations $\widetilde{P}_0$ and $\widetilde{P}_1$ internally. The *Strong Tweakable Fork-permutation* (STFP) advantage of a distinguisher $\mathcal{A}$ in distinguishing the real world $(\widetilde{\mathsf{F}}^+_k, \widetilde{\mathsf{F}}^-_k)$ from the ideal world $(\$^+, \$^-)$ as

$$\mathbf{Adv}^{\text{stfp}}_{\widetilde{\mathsf{F}}_k}(\mathcal{A}) = \Delta_{\mathcal{A}}\left((\widetilde{\mathsf{F}}^+_k, \widetilde{\mathsf{F}}^-_k); (\$^+, \$^-) \right).$$

We say that $\widetilde{\mathsf{F}} = (\widetilde{\mathsf{F}}^+, \widetilde{\mathsf{F}}^-)$ is $(q, \mathtt{t}, \epsilon)$-$\mathsf{STFP}$-secure if the maximal STFP advantage on F is at most ϵ where the maximum is taken over all distinguishers that make a total of at most q queries to both the encryprtion and decryption oracle altogether and run in time at most $\mathtt{t}$ steps.

Here, we state a fact on secure fork-ciphers which we will require later.

Lemma 1. Let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be an ideal fork-cipher. Given $k,\ x \in \{0,1\}^n$ and $y\|z \in \{0,1\}^{2n}$, the probability of finding a tweak J such that $\widetilde{\mathsf{F}}[k, J, x, 2] = y\|z$ is bounded by 2^{-2n}. Moreover, for any given $y\|z \in \{0,1\}^{2n}$, the probability of finding a key k, a tweak J, and an input x such that $\widetilde{\mathsf{F}}[k, J, x, 2] = y\|z$ can be bounded by 2^{-n}.

Proof. The proof is straight-forward and follows directly from the definition of a secure fork-cipher. For the first part, the event is identical to finding J such that $(\widetilde{\mathsf{F}}[k, J, x, 0] = y)$ and $(\widetilde{\mathsf{F}}[k, J, x, 1] = z)$, the probability of which can be bounded by 2^{-2n} from the randomness of $\widetilde{\mathsf{F}}$. For the second part, the adversary can guess a key k and set $x = \widetilde{\mathsf{F}}^{-1}[k, J, z, 1, 0]$. This essentially ensures that the above equality holds if and only one can find J such that $\widetilde{\mathsf{F}}[k, J, \widetilde{\mathsf{F}}^{-1}[k, J, z, 1, 0], 0] = y$, the probability of which can be bounded by 2^{-n}. $\square$

2.2 Nonce-based Authenticated Encryption and Its Security in The Presence of Leakage

Let $\mathcal{K}, \mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{C}, \mathcal{T}$ be non-empty sets for keys, nonce, associated data, messages, ciphertext, and tags, respectively. A nonce-based authenticated encryption scheme with associated data (nAEAD) 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 $\mathcal{D} : \mathcal{K}\times\mathcal{N}\times\mathcal{A}\times\mathcal{C}\times\mathcal{T} \to \mathcal{M}\cup\{\perp\}$, where $\perp$ indicates an invalid ciphertext-tag pair. We will often write $\mathcal{E}_K^{N,A}(M)$ and $\mathcal{D}_K^{N,A}(C,T)$ for $\mathcal{E}(K, N, A, M)$ and $\mathcal{D}(K, N, C, T)$, respectively.

It holds that $\mathcal{D}_K^{N,A}(C,T) = \perp$ if $\nexists\ M \in \mathcal{M}$ satisfying $\mathcal{E}_K^{N,A}(M) = (C,T)$. In this case, we call (N, A, C, T) an invalid decryption query. The correctness condition of an nAE scheme states that for every $K \in \mathcal{K}, N \in \mathcal{N}, A \in \mathcal{A}$, and $M \in \mathcal{M}$, it holds that $\mathcal{D}_K^{N,A}(\mathcal{E}_K^{N,A}(M)) = M$; the tidiness condition 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 $\mathcal{D}_K^{N,A}(C,T) \neq \perp$, it holds that $\mathcal{E}_K^{N,A}(\mathcal{D}_K^{N,A}((C,T))) = (C,T)$. Let $\Pi = (\mathcal{E}, \mathcal{D})$ be an nAEAD scheme. We define the nAEAD advantage of $\mathcal{A}$ on Π as

$$\mathbf{Adv}_{\Pi_K}^{\mathsf{nAEAD}}(\mathcal{A}) \overset{\Delta}{=} \Delta_{\mathcal{A}}\left((\mathcal{E}_K, \mathcal{D}_K); (\$, \perp)\right) ,$$

where \$ is a random oracle that returns a tuple of uniform random strings (C,T) of expected lengths on input (N, A, M) and $\perp$ is a reject oracle that always returns $\perp$ on any input (N, A, C, T). We call $\mathcal{A}$ *nonce-respecting* if and only if it refrains from reusing nonces in queries directed to the first oracle (referred to the encryption oracle), and the corresponding advantage is termed *nonce-respecting* advantage. In contrast, should $\mathcal{A}$ reuse nonces in queries to the first oracle, it is classified as a *nonce-misuse* adversary, with the associated advantage termed as *nonce-misuse* advantage. Note that in both scenarios we presume that $\mathcal{A}$ can repeats nonces in queries to the second oracle (refered to as the decryption oracle). Throughout this work, we will consider non-trivial adversaries i.e. it will not forward a response from its encryption oracle to the decryption oracle as a forgery.

Security under Nonce Misuse and Leakage. We consider two different classes of adversaries, called *nonce-misuse-resilient* adversary and *leakage adversary.* In the setting with nonce-misuse resilience, an adversary $\mathcal{A}$ is allowed to make two types of queries: non-challenge queries and challenge queries. Non challenger queries are answered through the real oracles and the challenge queries are answered either through the real oracles or the ideal oracles. A nonce-misuse resilient adversary can repeat nonces during non-challenge queries to both the encryption and decryption oracles, but must employ a fresh nonce for every challenge query. On the other hand, in the leakage setting, a leakage adversary $\mathcal{A}$ is allowed to make queries to the encryption and to the decryption oracle in either of the two worlds and is given not only the output but also the leakages from the constructions.

Algorithm 2 The CIML2 experiment.

```
11: procedure INITIALIZE
12:     $K \xleftarrow{\$} \mathcal{K}$
13:     $b \leftarrow 0$
14:     $S \leftarrow \emptyset$

21: function $\mathrm{L}\mathcal{E}(K, N, A, M, \mathfrak{L}_{\mathcal{E}})$
22:     $(C, T) \leftarrow \mathcal{E}(K, N, A, M)$
23:     $S \xleftarrow{\cup} \{(N, A, C, T)\}$
24:     $R \xleftarrow{\$} \mathcal{R}$
25:     $L \leftarrow \mathfrak{L}_{\mathcal{E}}(K, N, A, M, R)$
26:     return $(C, T, L)$

31: function $\mathrm{L}\mathcal{D}(K, P, N, A, C, T, \mathfrak{L}_{\mathcal{D}})$
32:     $M \leftarrow \mathcal{D}(K, N, A, C, T)$
33:     $R \xleftarrow{\$} \mathcal{R}$
34:     $L \leftarrow \mathfrak{L}_{\mathcal{D}}(K, N, A, C, T, R)$
35:     return $(M, L)$

41: function FINALIZE
42:     return $b$

51: function $\mathrm{L}\mathcal{D}^{ch}(K, N, A, C, T, \mathfrak{L}_{\mathcal{D}})$
52:     $R \xleftarrow{\$} \mathcal{R}$
53:     $L \leftarrow \mathfrak{L}_{\mathcal{D}}(K, N, A, C, T, R)$
54:     if $(N, A, C, T) \in S$ then
55:         return $(\perp, L)$
56:     if $\mathcal{D}(K, N, A, C, T) = \perp$ then
57:         return $(\perp, L)$
58:     $M \leftarrow \mathcal{D}(K, N, A, C, T)$
59:     $b \leftarrow 1$
60:     return $(M, L)$
```

2.2.1 Authenticity of nAEAD under Leakage

Our attention will be directed towards nonce-based AEAD under leakage. We specifically consider cases where the AEAD scheme exhibits nonce-misuse-resistant integrity and nonce-misuse-resilient confidentiality. Achieving complete nonce-misuse-resistant confidentiality under leakage assumptions appears challenging, as shown in [BGP$^+$20]. Leakage in the AEAD implementation is categorized into leakage during encryption and leakage during decryption. Berti et al. initially introduced a nonce-misuse-resistant leakage integrity concept with only encryption leakage (CIML) in [BKP$^+$18]. Subsequently, they extended this notion to encompass both encryption and decryption leakage (CIML2) in [BPPS17]. In [BGP$^+$20], Berti et al. further defined a multi-user distinguishing version of the CIML2 notion called muCIML2, emphasizing the equivalency between forgery and distinguishing attacks. The follow-up work [Lis21] introduced a single-user, multi-challenge variant of the muCIML2 notion, called qCIML2 to prove the integrity security of the TEDT2 construction. However, qCIML2 has been introduced as a distinguishing game advantage, but we use the notion in our work in terms of the forging game to prove the integrity security of our construction for single user. Therefore, we will use the CIML2 notion in terms of single user forging, described in Algorithm. 2.

Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme and $\mathfrak{L}_{\mathcal{E}}, \mathfrak{L}_{\mathcal{D}}$ be two leakage functions associated with the encryption and decryption function, respectively. We define two functions called *encryption with leakage* and *decryption with leakage*, denoted as $\mathrm{L}\mathcal{E}$ and $\mathrm{L}\mathcal{D}$ respectively corresponding to the encryption and decryption function $\mathcal{E}, \mathcal{D}$ as follows: on input (K, N, A, M), $\mathrm{L}\mathcal{E}$ returns $(\mathcal{E}_K^{N,A}(M), \mathfrak{L}_{\mathcal{E}}(K, N, A, M, R))$ for some randomness $R \xleftarrow{\$} \mathcal{R}$ sampled uniformly at random from a non-empty set of randomness $\mathcal{R}$, which models randomness (e.g. implementation and measuring noise) in practice. Similarly, on input (K, N, A, C, T), $\mathrm{L}\mathcal{D}$ returns $(\mathcal{D}_K^{N,A}(C, T), \mathfrak{L}_{\mathcal{D}}(K, N, A, C, T, R)$ for some $R \xleftarrow{\$} \mathcal{R}$. In the following, we recall the CIML2 definition for $\Pi[\mathsf{IC}]$, i.e. an AEAD scheme based on an ideal i.e. random cipher IC. We write $\mathsf{IC}^{\pm}$ as short form to indicate that oracles to IC and IC^{-1} are available.

Definition 1 (CIML2)**.** Let $K \xleftarrow{\$} \mathcal{K}$ and let $\mathcal{A}$ be an adversary for a nonce-based AEAD scheme $\Pi[\mathsf{IC}]_K = (\mathcal{E}[\mathsf{IC}]_K, \mathcal{D}[\mathsf{IC}]_K)$ based on an ideal cipher IC. Let $\mathfrak{L}_{\mathcal{E}}$ and $\mathfrak{L}_{\mathcal{D}}$ be two leakage functions associated with encryption and decryption, respectively. Then, we define

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\Pi[\mathsf{IC}]_K, \mathfrak{L}_{\mathcal{E}}, \mathfrak{L}_{\mathcal{D}}}(\mathcal{A}) \overset{\Delta}{=} \Pr[\mathcal{A}^{\mathrm{L}\mathcal{E}[\mathsf{IC}]_K, \mathrm{L}\mathcal{D}[\mathsf{IC}]_K, \mathsf{IC}^{\pm}} \text{ forges}] .$$

Algorithm 3 The qCCAmL2 experiment.

```
11: procedure INITIALIZE
12:     $K \xleftarrow{\$} \mathcal{K}$
13:     $b \xleftarrow{\$} \{0,1\}$
14:     $S_N \leftarrow \emptyset$
15:     $S_{ch} \leftarrow \emptyset$

21: function $\mathrm{L}\mathcal{E}^1(K, N, A, M, \mathfrak{L}_\mathcal{E})$
22:     $(C, T) \leftarrow \mathcal{E}(K, N, A, M)$
23:     $S_N \xleftarrow{\cup} N$
24:     $R \xleftarrow{\$} \mathcal{R}$
25:     $L \leftarrow \mathfrak{L}_\mathcal{E}(K, N, A, M, R)$
26:     return $(C, T, L)$

31: function $\mathrm{L}\mathcal{D}^1(K, N, A, C, T, \mathfrak{L}_\mathcal{D})$
32:     if $(N, A, C, T) \in S_{ch}$ then
33:         return $\perp$
34:     $R \xleftarrow{\$} \mathcal{R}$
35:     $M \leftarrow \mathcal{D}(K, N, A, C, T)$
36:     $L \leftarrow \mathfrak{L}_\mathcal{D}(K, N, A, C, T, R)$
37:     return $(M, L)$

41: function FINALIZE($b'$)
42:     return $b = b'$

51: function $\mathrm{L}\mathcal{E}^2(K, N, A, M, \mathfrak{L}_\mathcal{E})$
52:     if $N \in S_N$ then
53:         return $\perp$
54:     if $b = 0$ then
55:         $(C, T) \leftarrow \mathcal{E}(K, N, A, M)$
56:     else
57:         $M^* \xleftarrow{\$} \{0,1\}^{|M|}$
58:         $M \leftarrow M^*$
59:         $(C, T) \leftarrow \mathcal{E}(K, N, A, M)$
60:     $S_N \xleftarrow{\cup} N$
61:     $S_{ch} \xleftarrow{\cup} (N, A, C, T)$
62:     $R \xleftarrow{\$} \mathcal{R}$
63:     $L \leftarrow \mathfrak{L}_\mathcal{E}(K, N, A, M, R)$
64:     return $(C, T, L)$

61: function $\mathrm{L}\mathcal{D}^2(K, N, A, C, T, \mathfrak{L}_\mathcal{D})$
62:     if $(N, A, C, T) \notin S_{ch}$ then
63:         return $\perp$
64:     $R \xleftarrow{\$} \mathcal{R}$
65:     $L \leftarrow \mathfrak{L}_\mathcal{D}(K, N, A, C, T, R)$
66:     return $L$
```

We define $\mathbf{Adv}^{\mathsf{CIML2}}_{\Pi[\mathsf{IC}]_K}(p, q, \sigma)$ as the maximum advantage of all adversary making at most q queries of at most σ blocks to its oracles and every leakage will be computed at most p times.

2.2.2 Privacy of nAEAD under Leakage

Next, we consider privacy under leakage. Let Π, $\mathfrak{L}_\mathcal{E}$, and $\mathfrak{L}_\mathcal{D}$ be defined as before. We define the following two pair of oracles corresponding to the encryption and decryption functions: $(\mathrm{L}\mathcal{E}^1, \mathrm{L}\mathcal{D}^1)$ and $(\mathrm{L}\mathcal{E}^2, \mathrm{L}\mathcal{D}^2)$, where on input (K, N, A, M), $\mathrm{L}\mathcal{E}^1$ returns $(\mathcal{E}_K^{N,A}(M), \mathfrak{L}_\mathcal{E}(K, N, A, M, R))$ for a randomly sampled R. Similarly, on input (K, N, A, C, T), the oracle $\mathrm{L}\mathcal{D}^1$ returns $(\mathcal{D}_K^{N,A}(C, T), \mathfrak{L}_\mathcal{D}(K, N, A, C, T, R))$ for a randomly sampled R. Another pair of oracles $(\mathrm{L}\mathcal{E}^2, \mathrm{L}\mathcal{D}^2)$ are functionally almost identical to that of $(\mathrm{L}\mathcal{E}^1, \mathrm{L}\mathcal{D}^1)$, with the only difference that $\mathrm{L}\mathcal{E}^2$ and $\mathrm{L}\mathcal{D}^2$ additionally checks for nonce repetitions. We call the first pair of oracles $(\mathrm{L}\mathcal{E}^1, \mathrm{L}\mathcal{D}^1)$ *non-challenge oracles* and the other pair of oracles $(\mathrm{L}\mathcal{E}^2, \mathrm{L}\mathcal{D}^2)$ *challenge oracles*. Informally, qCCAmL2 is defined as a distinguishing game to distinguish the output of the challenge oracle in both the real and ideal worlds with additional access to the non-challenge oracle in both the worlds.

Definition 2 (qCCAmL2)**.** Let $K \xleftarrow{\$} \mathcal{K}$ and let $\mathcal{A}$ be an adversary for a nonce-based AEAD scheme $\Pi[\mathsf{IC}]_K = (\mathcal{E}[\mathsf{IC}]_K, \mathcal{D}[\mathsf{IC}]_K)$ based on an ideal cipher IC. Let $\mathfrak{L}_\mathcal{E}$ and $\mathfrak{L}_\mathcal{D}$ be two leakage functions associated with the encryption and decryption function, respectively. Then, we define

$$\mathbf{Adv}^{\mathsf{qCCAmL2}}_{\Pi[\mathsf{IC}]_K, \mathfrak{L}_\mathcal{E}, \mathfrak{L}_\mathcal{D}}(\mathcal{A}) \triangleq \Delta_\mathcal{A}(\mathrm{L}\mathcal{E}^1_K[\mathsf{IC}]_K, \mathrm{L}\mathcal{D}^1_K[\mathsf{IC}]_K, \mathrm{L}\mathcal{E}^2_K[\mathsf{IC}]_K, \mathrm{L}\mathcal{D}^2_K[\mathsf{IC}]_K, \mathsf{IC}^\pm\ ;$$
$$\mathrm{L}\mathcal{E}^1_K[\mathsf{IC}]_K, \mathrm{L}\mathcal{D}^1_K[\mathsf{IC}]_K, \$_\mathcal{E}, \$_\mathcal{D}, \mathsf{IC}^\pm)\,.$$

Here, $\mathcal{E}^1_K, \mathcal{D}^1_K$ are non-challenge oracles. These two oracles are the same for both the real and ideal world. The adversary can repeat nonces to these two oracles. The security is defined as distinguishing event between $\mathcal{E}^2_K, \mathcal{D}^2_K$ and $\$_\mathcal{E}, \$_\mathcal{D}$. As we are considering a nonce-misuse-resilient adversary, $\mathcal{A}$ cannot repeat nonce for the queries to these challenge oracles. In the ideal world, $\$_{\mathcal{E}}$ encrypts a random string of the same length as the original message. $\mathcal{D}_k^2$ accepts only the previous reply of $\mathcal{E}_k^2$ and outputs only the leakage, but not the decryption result to avoid trivial wins. Similarly, $\$_{\mathcal{D}}$ accepts only outputs from previous queries to $\$_{\mathcal{E}}$ and outputs the decryption leakages but not the decryption results to avoid trivial attacks.

In this work, we prove the confidentiality of the construction in the *nonce-misuse-resilient* setting, where adversaries must use a fresh nonce for each challenge query to $\mathrm{L}\mathcal{E}_K^2$ and $\$_{\mathcal{E}}$. By assuming there is no non-challenge valid decryption query, we define the $\mathsf{qCPAmL2}$ notion as follows:

Definition 3 (qCPAmL2)**.** Let $K \xleftarrow{\$} \mathcal{K}$ and let $\mathcal{A}$ be an adversary for a nonce-based AEAD scheme $\Pi[\mathsf{IC}]_K = (\mathcal{E}[\mathsf{IC}]_K, \mathcal{D}[\mathsf{IC}]_K)$ based on an ideal cipher IC. Let $\mathfrak{L}_{\mathcal{E}}$ and $\mathfrak{L}_{\mathcal{D}}$ be two leakage functions associated with the encryption and decryption function, respectively. Then, we define

$$\mathbf{Adv}^{\mathsf{qCPAmL2}}_{\Pi[\mathsf{IC}]_K,\mathfrak{L}_{\mathcal{E}},\mathfrak{L}_{\mathcal{D}}}(\mathcal{A}) \overset{\Delta}{=} \Delta_{\mathcal{A}}(\mathrm{L}\mathcal{E}_K^1[\mathsf{IC}]_K, \mathrm{L}\mathcal{E}_K^2[\mathsf{IC}]_K, \mathrm{L}\mathcal{D}_K^2[\mathsf{IC}]_K, \mathsf{IC}^{\pm}\ ; \mathrm{L}\mathcal{E}_K^1[\mathsf{IC}]_K, \$_{\mathcal{E}}, \$_{\mathcal{D}}, \mathsf{IC}^{\pm})\,.$$

We define $\mathbf{Adv}^{\mathsf{qCCAmL2}}_{\Pi[\mathsf{IC}]_K,\mathfrak{L}_{\mathcal{E}},\mathfrak{L}_{\mathcal{D}}}(p,q,\sigma)$ as the maximum advantage of all adversaries making at most q queries of at most σ blocks in total and every leakage will be computed at most p times and define $\mathbf{Adv}^{\mathrm{qCPAmL2}}_{\Pi[\mathsf{IC}]_K,\mathfrak{L}_{\mathcal{E}},\mathfrak{L}_{\mathcal{D}}}(p,q,\sigma)$ similarly in the natural manner.

3 Design and Specification of FEDT

In this section, we formally specify the FEDT authenticated encryption scheme and give a brief design rationale.

3.1 Specification

FEDT uses a fork-cipher $\widetilde{\mathsf{F}}$ as its underlying primitive that takes an n-bit key, a $2n$-bit tweak, and an n-bit input and produce a $2n$-bit output: $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$. The mode follows an Encrypt-then-MAC composition.

Encryption Module. The encryption function takes as inputs a nonce N, a message M, and a master key K and generates the ciphertext C (with $|C| = |M|$). The encryption is essentially a one-time pad, where the key-stream is generated from the nonce using the tweakable fork cipher in a tree-like structure as shown in Figure 1a.

Authentication Module. The authentication module takes as inputs the nonce N, associated data A, the ciphertext C, and generates the tag T. First, we use an injective padding function pad similarly as in [BGP+20] on N, A, C to generate a bit string U whose length is a guaranteed multiple of $2n$ bits, where, $\mathsf{pad}(N, A, C) = (A\|C\|0^{2n-\delta})\ \|N\ \|\ ([|A|]_{n/2}\|[|C|]_{n/2})$. The padding function ensures that for different $(N, A, C), (N', A', C')$, we have $U \neq U'$. Next, the authentication module generates a $2n$-bit string $V\|W$ from U using $2n$-bit parts as tweak inputs to $\widetilde{\mathsf{F}}$. The i-th fork-cipher invocation takes the most significant n-bits of the previous fork-cipher output as the key, the least significant n-bits as the input and U_i as tweak and generates a $2n$-bit output. The initial input and key is taken as 0^n, and the last fork-cipher output is considered as $V\|W$. Finally, $V\|W$ is used as the tweak with input 0^n, and key K into $\widetilde{\mathsf{F}}$ to generate the tag. A pictorial description of the authentication module is shown in Fig. 1b.

Algorithm 4 Specification of FEDT.

```
function $\mathcal{E}(K, N, A, M)$
    $k_1 \| k_2 \leftarrow \mathsf{KDF}(K, N \| 0^n, N)$
    $C \leftarrow \mathsf{Enc}(k_1 \| k_2, N, M)$
    $V \| W \leftarrow \mathsf{Hash}(A, N, C)$
    $T \leftarrow \mathsf{TGF}(K, V \| W, 0^n)$

function $\mathcal{D}(K, N, A, C, T)$
    $V \| W \leftarrow \mathsf{Hash}(A, N, C)$
    $ip \leftarrow \mathsf{TGF}^{-1}(K, V \| W, T)$
    if $ip = 0^n$ then
        return $\mathsf{Enc}(K, N, C)$
    return $\perp$

function $\mathsf{pad}(A, N, C)$
    $l \leftarrow |A \| C|$
    $\delta \leftarrow l \mod 2n$
    $Y \leftarrow (A \| C \| 0^{2n-\delta}) \| N \| ([|A|]_{n/2} \| [|C|]_{n/2})$
    return $Y$

function $\mathsf{KDF}(K, J, X)$
    return $\widetilde{\mathsf{F}}^{+}(K, J, X, 2)$

function $\mathsf{TGF}(K, J, X)$
    return $\widetilde{\mathsf{F}}^{+}(K, J, X, 0)$

function $\mathsf{Enc}(k_1 \| k_2, N, M)$
    $l \leftarrow \lceil |M|/n \rceil$
    $M_1 \| M_2 \| \cdots \| M_l \leftarrow M$
    for $i \leftarrow 3, 5, 7, \ldots, (2l-3)$ do
        $a \leftarrow (i-1)/2$
        $J_a \leftarrow N \| [a]_n$
        $k_i \| k_{i+1} \leftarrow \widetilde{\mathsf{F}}^{+}(k_a, J_a, N, 2)$
    for $i \leftarrow 1, 2, 3, \ldots, l$ do
        $C_i \leftarrow M_i \oplus k_{l-2+i}$
    $C \leftarrow C_1 \| C_2 \| \cdots \| C_l$
    return $C$

function $\mathsf{Hash}(A, N, C)$
    $U \leftarrow \mathsf{pad}(A, N, C)$
    $U_1 \| U_2 \| \cdots \| U_l \xleftarrow{}_{2n} U$
    $u_0 \leftarrow 0^n$
    $v_0 \leftarrow 0^n$
    for $i \leftarrow 1, 2, 3, \ldots, l$ do
        $u_i \| v_i \leftarrow \widetilde{\mathsf{F}}^{+}(v_{i-1}, U_i, u_{i-1}, 2)$
    return $u_l \| v_l$

function $\mathsf{TGF}^{-1}(K, J, Y)$
    return $\widetilde{\mathsf{F}}^{-}(K, J, Y, 0, 0)$
```

Decryption Algorithm. The corresponding decryption algorithm takes as inputs a nonce N, associated data A, and a ciphertext-tag pair (C, T) and outputs the message M if the verification was successful and $\perp$ otherwise. The algorithm is a verify-then-decrypt type algorithm. First, it invokes the authentication module on (N, A, C), and the inverse of the fork-cipher to verify the authenticity of the submitted ciphertext-tag pair. If the input is proven authentic, then it decrypts the ciphertext to get the plaintext. The usage of the inverse fork-cipher prevents leaking the correct tag for an invalid ciphertext. The complete specification of FEDT is provided in Algorithm 4. It is pictorically depicted in Fig. 1a and Fig. 1b for messages of $8n$ bits.

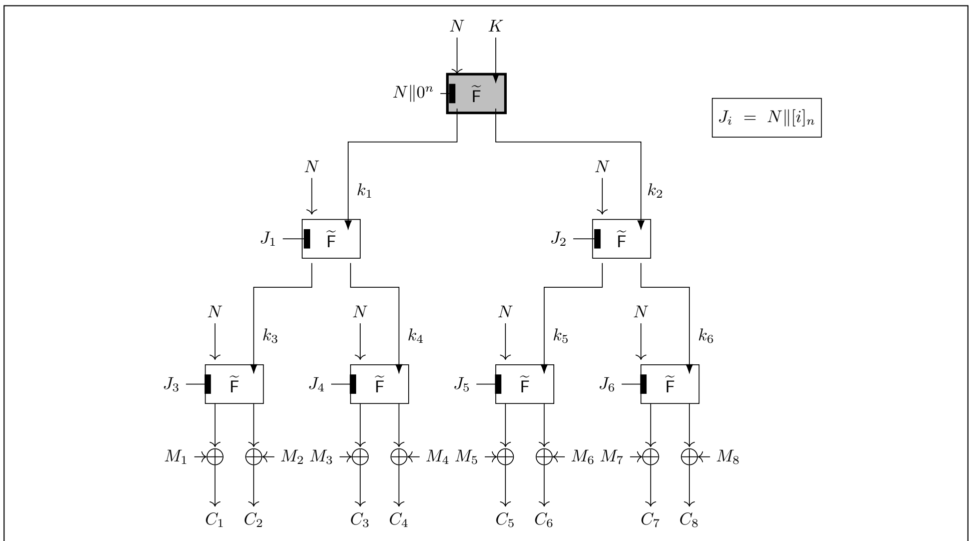
3.2 Design Rationale

In this section, we describe our design rationale of FEDT. Prior, we will briefly describe two existing Grade-3 leakage-resilient nAEAD constructions: (a) TEDT and (b) TEDT2, and a overview of our construction FEDT.

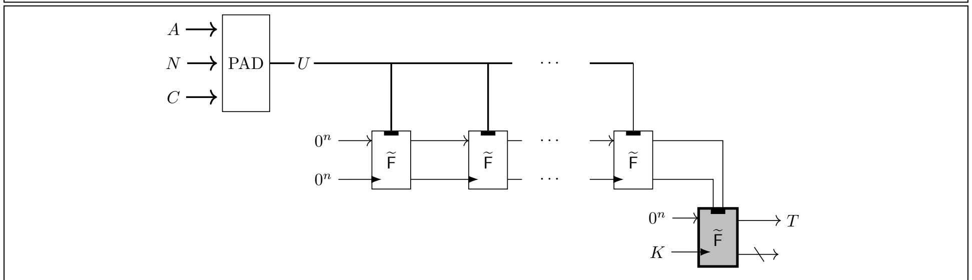
TEDT. The structure of TEDT is as follows:

$$\begin{cases} IV \leftarrow \widetilde{\mathsf{E}}(K, PK \| 0, N \| 0^{n/4}), \\ C \leftarrow \mathsf{G}[\widetilde{\mathsf{E}}](IV, N) \oplus M, \\ V \| W \leftarrow \mathsf{H}[\widetilde{\mathsf{E}}](N, A, C), \\ T \leftarrow \widetilde{\mathsf{E}}(K, [W]_{n-1} \| 1, V). \end{cases}$$

A public user-specific value PK aids TEDT in maintaining security integrity within a multi-user framework. The underlying PRG, denoted as G, is derived from Bellare-Yee [BY03] re-keying principles, following guidelines proposed for leakage resilience in [PSV15]. Notably, each TBC call in G shares the same tweak within a user and utilizes an identical key in two TBCs at each level. Consequently, only n bits differ between two $2n$-bit tweakeys. The hash function H employed in TEDT is a variation of Hirose's double-block hash function [Hir06].



(a) Encryption for a message of eight blocks.



(b) Authentication, composed of hashing and the TGF.

Figure 1: Schematic view of FEDT for eight message blocks.

The tag generation involves a call to a strongly protected implementation of the TBC utilizing W as a tweak, V as input, and the master key K to produce the tag T.

TEDT2. TEDT2 adheres to a structure akin to TEDT with modifications in the KDF and TGF. A synopsis of TEDT2 is as follows:

$$\begin{cases} a\|b \leftarrow \mathsf{KDF}(K, N)\,, \\ C \leftarrow \mathsf{G}[\widetilde{\mathsf{E}}](a, b, N) \oplus M\,, \\ U\|V \leftarrow \mathsf{H}[\widetilde{\mathsf{E}}](A, C)\,, \\ T \leftarrow \widetilde{\mathsf{E}}(K, 8\|N\|V, U)\,. \end{cases}$$

The KDF employs two TBC calls sharing the same master key K, fixed input 0, and distinct tweaks using domain separation. TEDT2 utilizes the encryption function G, which is a CTR-based mode, where each TBC employs a $3n$-bit tweakey with domain-separated $2n$-bit tweaks. The hash function H is a variant of Naito's hash function [Nai19] based on MDPH [HPY07]. Ultimately, TEDT2 employs a call to a strongly protected TBC implementation with the master key K, input U, a nonce N, input V, and a domain separation factor in tweak to generate the tag T.

FEDT. The outline of FEDT is as follows:

$$\begin{cases} K_1 \| K_2 \leftarrow \widetilde{\mathsf{F}}(K, N \| 0^n, N)\,, \\ C \leftarrow \mathsf{G}[\widetilde{\mathsf{E}}](K_1, K_2, N) \oplus M\,, \\ U \| V \leftarrow \mathsf{H}[\widetilde{\mathsf{E}}](N, A, C)\,, \\ T \leftarrow \widetilde{\mathsf{F}}(K, U \| V, 0^n)\,. \end{cases}$$

FEDT uses a call to a strongly protected implementation of a fork-cipher in the KDF using master key K and the nonce in both input and tweak. The encryption function Enc uses then a less protected implementation of the fork-cipher with a tweakey of $3n$ bit. The tweaks of each fork-cipher call depend on the nonce and the index of the call in the encryption tree which yields a distinct tweak for every primitive call. FEDT employs a fork-cipher-based hash function that is heavily inspired from TBC-based double-block-length hashing. FEDT uses the fixed input 0^n in the TGF in a similar way as [BGPS21]. Finally, FEDT uses a call to a strongly protected implementation of the fork-cipher with the master key K, the $2n$-bit hash output as the tweak, and the fixed input 0^n as inputs. FEDT output the most significant n-bit of the output as the tag. Note that FEDT does not use the nonce in the TGF.

Comparison. The privacy of TEDT is limited to the birthday bound primarily since it uses the same tweak in each TBC call in the encryption for the same user. TEDT processes the fresh n-bit key at every step in the encryption. To address this bottleneck, TEDT2 uses a fresh $2n$-bit tweakey for every step of the encryption. To preserve the rate of TEDT, TEDT2 processes two message blocks per step compared to TEDT's single block. Instead of refreshing the tweak in each step to achieve beyond-birthday-bound privacy, FEDT uses distinct tweaks of the form $N \| [i]_n$. This approach eliminates the need for an extra primitive call to process fresh tweaks, yielding an efficient solution for beyond-birthday-bound privacy. The tree structure also restricts the length of the longest TBC sequence to $\log(d)$ for a dn-bit message, surpassing the efficiency of the d-length TBC sequence in both TEDT and TEDT2. Thus, it can enhance parallelization, and have a primitive rate of one compared to 0.5 for TEDT and TEDT2. However, the different primitives demand a more fine-grained comparison.

The authentication in FEDT follows a sequence similar to the pad-Hash-TGF employed in TEDT and TEDT2. We introduce a hash function based on a fork-cipher, which follows from the TBC-based double-block-length hash function. In this hash, every call to the primitive takes a $2n$-bit block of the padded nonce, associated data, and ciphertext string as a tweak. Subsequently, in the TGF, the $2n$-bit hash serves as the tweak in a strongly protected call to the fork-cipher with the master key K. FEDT employs a fixed input 0^n, distinguishing it from the variable input in TEDT and TEDT2. The primitive rate in this hash function is twice that of TEDT and TEDT2, however, under a different primitive.

3.3 FEDT*: A Low-latency Variant of FEDT

While FEDT may benefit from parallelization because of its tree-structured encryption, this induces also a non-trivial latency of roughly $\log_2(d)$ fork-cipher calls before the first ciphertext can be output for a message of d n-bit blocks. As part of a remedy, we propose FEDT* as a second approach, that replaces the encryption function of FEDT with a fork-cipher-based variant of the RCTR encryption in TEDT2. A schematic diagram of FEDT* is depicted in Figure 2 along with a description in Algorithm 5. FEDT* inherits the KDF, Hash function, and TGF from FEDT. However, it processes four message blocks in each level of encryption using three calls to the primitive, achieving an increased rate (of 4/3), where each level generates two keys for the next level and four n-bit ciphertext blocks. Note that these three fork-cipher calls can be computed in parallel. The rate at each level

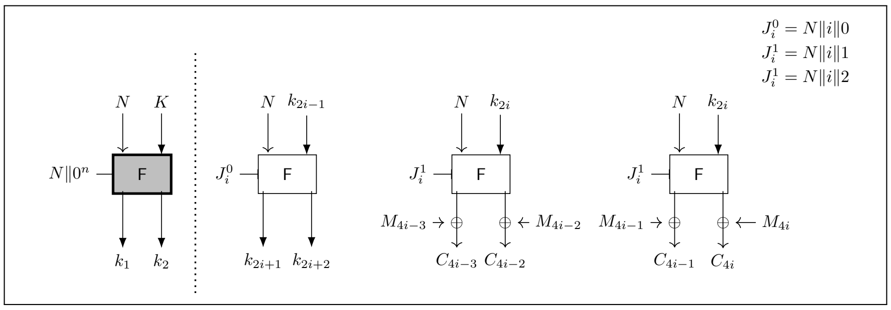


Figure 2: Encryption function of FEDT*.

Algorithm 5 Specification of FEDT*.

```
function $\mathcal{E}(K, N, A, M)$
    $k_1\|k_2 \leftarrow \mathsf{KDF}(K, N\|0^n, N)$
    $C \leftarrow \mathsf{Enc}(k_1\|k_2, N, M)$
    $V\|W \leftarrow \mathsf{H}(A, N, C)$
    $T \leftarrow \mathsf{TGF}(K, V\|W, 0^n)$

function $\mathcal{D}(K, N, A, C, T)$
    $V\|W \leftarrow \mathsf{H}(A, N, C)$
    $ip \leftarrow \mathsf{TGF}^{-1}(K, V\|W, T)$
    if $ip = 0^n$ then
        return $\mathsf{Enc}(K, N, C)$
    return $\perp$

function $\mathsf{pad}(A, N, C)$
    $l \leftarrow |A\|C|$
    $\delta \leftarrow l \mod 2n$
    $Y \leftarrow (A\|C\|0^{2n-\delta})\|N\|([|A|]_{n/2}\|[|C|]_{n/2})$
    return $Y$

function $\mathsf{KDF}(K, J, X)$
    return $\widetilde{\mathsf{F}}^{+}(K, J, X, 2)$

function $\mathsf{TGF}(K, J, X)$
    return $\widetilde{\mathsf{F}}^{+}(K, J, X, 0)$

function $\mathsf{Enc}(k_1\|k_2, N, M)$
    $l \leftarrow \lceil |M|/n \rceil$
    $M_1\|M_2\| \cdots \|M_l \leftarrow M$
    for $i = 1, 2, 3, \ldots, l/4$ do
        $k_{2i+1}\|k_{2i+2} \leftarrow \widetilde{\mathsf{F}}(k_{2i-1}, N\|[i]_{n-8}\|[0]_8, N, 2)$
        $Y_{4i-3}\|Y_{4i-2} \leftarrow \widetilde{\mathsf{F}}(k_{2i}, N\|[i]_{n-8}\|[1]_8, N, 2)$
        $Y_{4i-1}\|Y_{4i} \leftarrow \widetilde{\mathsf{F}}(k_{2i}, N\|[i]_{n-8}\|[2]_8, N, 2)$
        for $s \leftarrow 0, \ldots, 3$ do
            $C_{4i-s} = M_{4i-s} \oplus Y_{4i-s}$
    return $C \leftarrow C_1\|C_2\| \cdots \|C_l$

function $\mathsf{Hash}(A, N, C)$
    $U \leftarrow \mathsf{pad}(A, N, C)$
    $U_1\|U_2\| \cdots \|U_l \leftarrow_{2n} U$
    $u_0 \leftarrow 0^n$
    $v_0 \leftarrow 0^n$
    for $i \leftarrow 1, 2, 3, \ldots, l$ do
        $u_i\|v_i \leftarrow \widetilde{\mathsf{F}}^{+}(v_{i-1}, U_i, u_{i-1}, 2)$
    return $u_l\|v_l$

function $\mathsf{TGF}^{-1}(K, J, Y)$
    return $\widetilde{\mathsf{F}}^{-}(K, J, Y, 0, 0)$
```

can be further increased by employing more parallel fork-ciphers with domain-separating tweaks. We note that a complete comparison must depend on the cost of the primitives for achieving a similar security level. Since the optimal security and efficiency of fork-ciphers are less studied than tweakable block ciphers, we leave the further understanding of the security of concrete instances as an open problem.

4 Security Analysis of FEDT

In this section, we analyze the security of FEDT. First, we study the collision security of the underlying hash function, before we consider its CIML2 security and qCCAmL2 security bounds.

4.1 Collision Security of the Hash Function

Here, we bound the collision security of the underlying hash function of FEDT $\mathsf{H}[\widetilde{\mathsf{F}}]$.

Lemma 2. Let $\mathcal{A}$ be an adversary aiming to discover a collision in the hash function $\mathsf{H}[\widetilde{F}]$ while making a maximum of q_p primitive queries, including those made during hash queries. Let Coll be the event that $\mathcal{A}$ finds a collision in H i.e., $\mathcal{A}$ output U and U' such that $\mathsf{H}[\widetilde{\mathsf{F}}](U) = \mathsf{H}[\widetilde{\mathsf{F}}](U')$. Then,

$$\Pr[\mathsf{Coll}] \leq \frac{q_p}{2^n} + \frac{q_p^2}{2^{2n}} + \frac{q_p^3}{2^{3n}}\,.$$

Proof. $\mathcal{A}$ can make two kinds of queries to oracles: construction queries and ideal-cipher queries. In the former, $\mathcal{A}$ submits an input U of dn bits for some integer d and obtains a $2n$-bit output $\mathsf{H}[\widetilde{\mathsf{F}}](U) = V\|W$. For ideal-cipher queries, $\mathcal{A}$ chooses a key k, a tweak J, and evaluate both the forward and backward queries to the ideal-cipher $\widetilde{\mathsf{F}}$ as follows: for the forward query, $\mathcal{A}$ submits a key k, a tweak J, and an n-bit input X with a selector bit s and $\mathcal{A}$ gets $\widetilde{\mathsf{F}}^{+}(k, J, X, s)$ as the response. On the other hand, for a backward query, $\mathcal{A}$ submits a key k, a tweak J, and an n-bit input Y with a bit b and a selector bit s and gets $\widetilde{\mathsf{F}}^{-}(k, J, Y, b, s)$ as the response. The collection of all construction queries and ideal-cipher queries constitute the attack transcript as follows: $\tau = \{\tau_h, \tau_p\}$, where τ_h consists of queries of the form $(U, V\|W)$, where U is queried to the hash function $\mathsf{H}[\widetilde{\mathsf{F}}]$ during construction queries and $V\|W$ is the corresponding response, i.e $\mathsf{H}[\widetilde{\mathsf{F}}](U) = V\|W$.

On the other hand, τ_p consists of ideal-cipher queries of the form $(k, J, X, Y\|Z)$, where $Y\|Z \leftarrow \widetilde{\mathsf{F}}^{+}(k, J, X, 2)$. Note that, for a fixed key k and a tweak J, any of X, Y, Z will fix the value of other two.

Let $\mathtt{bad}$ be a Boolean flag, which is true if and only if there exists an ideal-cipher query that hits the initial value 0^{2n}. Formally,

$$\boxed{\mathtt{bad} \leftarrow 1 \text{ if } \exists (k, J, x, y\|z) \in \tau_p \text{ such that } y\|z = 0^{2n}.}$$

To set the boolean flag $\mathtt{bad}$, the adversary must discover a key, tweak, input tuple (k, J, x) such that $\widetilde{\mathsf{F}}(k, J, x, 2) = 0^{2n}$ holds. Following Lemma 1, for a fixed tuple of (k, J, x), the probability that $\widetilde{\mathsf{F}}(k, J, x, 2) = 0^{2n}$ is at most 2^{-n}. Moreover, the number of choices of such tuple is at most q_p. Hence we have

$$\Pr[\mathtt{bad}] \leq \frac{q_p}{2^n}\,. \tag{1}$$

We will now establish an upper bound on the collision probability of the hash function given that $\mathtt{bad}$ condition does not occur. Following Eqn. (1), we have

$$\Pr[\mathsf{Coll}] \leq \Pr[\mathsf{Coll}|\neg\mathtt{bad}]] + \Pr[\mathtt{bad}] \leq \underbrace{\Pr[\mathsf{Coll}|\neg\mathtt{bad}]}_{(1)} + \frac{q_p}{2^n}\,. \tag{2}$$

It remains to bound (1). Note that a collision in the hash function implies

$$(U, V\|W), (U', V'\|W') \in \tau_h : V\|W = V'\|W'.$$

Let U and U' be represented as follows: $U \leftarrow U_1\|U_2\|\cdots\|U_l$ and $U' \leftarrow U'_1\|U'_2\|\cdots\|U'_{l'}$, where each U_i and U'_j is a $2n$-bit block. Note that, $(U, V\|W) \in \tau_h$, implies the existence of a sequence $\Big((u_0, v_0), (u_1, v_1), \ldots, (u_l, v_l)\Big)$, where each $u_i, v_i \in \{0,1\}^n$ with the initial values $(u_0, v_0) = (0^n, 0^n)$ and $(u_l, v_l) = (V, W)$. Similarly, for $(U', V\|W) \in \tau_h$, implies the existence of a sequence $\Big((u'_0, v'_0), (u'_1, v'_1), \ldots, (u'_{l'}, v'_{l'})\Big)$, where each $u'_i, v'_i \in \{0,1\}^n$ with the initial values $(u'_0, v'_0) = (0^n, 0^n)$ and $(u'_{l'}, v'_{l'}) = (V, W)$. Furthermore, it must hold that

$$\begin{cases} \widetilde{\mathsf{F}}^{+}(v_{i-1}, U_i, u_{i-1}, 2) = u_i\|v_i, \;\; \forall i = 1, 2, \ldots, l \\ \widetilde{\mathsf{F}}^{+}(v'_{i-1}, U'_i, u'_{i-1}, 2) = u'_i\|v'_i \;\; \forall i = 1, 2, \ldots, l' \end{cases}$$

During the hash evaluation for input U and U', we add all the intermediate state variables $(v_{i-1}, U_i, u_{i-1}, u_i \| v_i)$ and $(v'_{i-1}, U'_i, u'_{i-1}, u'_i \| v'_i)$ to the ideal-cipher query transcript τ_p. Without loss of generality, we assume that no (u_i, v_i) is equal to any (u'_j, v'_j) except for the initial and final pairs.[2] Furthermore, we also assume that $(v_{l-1}, U_l, u_{l-1}, u_l \| v_l)$ is added to τ_p after $(v'_{l'-1}, U_{l'}, u'_{l'-1}, u'_{l'} \| v'_{l'})$. Now, we have the following cases:

Case (1): $(v_{l-1}, U_l, u_{l-1}, u_l \| v_l)$ **is added to** τ_p **after** $(v_{l-2}, U_{l-1}, u_{l-2}, u_{l-1} \| v_{l-1})$ **:** Let Coll_1 represent the event where $\mathcal{A}$ finds a collision in H. In this scenario, with a given key v_{l-1} and a given input u_{l-1}, the adversary must discover a suitable tweak $J = U_l$, such that $\widetilde{\mathsf{F}}^+(v_{l-1}, J, u_{l-1}, 2) = u_l \| v_l$ for a given output $u_l \| v_l$. For a fixed value of (u_{l-1}, v_{l-1}) and for a fixed value of (u_l, v_l), we apply Lemma 1, and it follows that the probability of finding such a tweak J is upper bounded by 2^{-2n} due to the randomness of the fork-cipher $\widetilde{\mathsf{F}}$. Furthermore, there are at most q_p choices for (u_{l-1}, v_{l-1}) and at most q_p choices for (u_l, v_l). Therefore, by varying over all possible choices of indices, we have

$$\Pr[\mathsf{Coll}_1 | \neg \mathsf{bad}] \leq \frac{q_p^2}{2^{2n}} . \tag{3}$$

Case (2): $(v_{l-1}, U_l, u_{l-1}, u_l \| v_l)$ **is added to** τ_p **before** $(v_{l-2}, U_{l-1}, u_{l-2}, u_{l-1} \| v_{l-1})$ **:** In this scenario, with a given output $u_l \| v_l$, the adversary $\mathcal{A}$ must obtain a key $k = v_{l-1}$, a tweak $J = U_l$, and an input $x = u_{l-1}$, such that $\widetilde{\mathsf{F}}^+(k, J, x, 2) = u_l \| v_l$. For a fixed choice of (u_{l-1}, v_{l-1}) and for a fixed choice of (u_l, v_l), we apply Lemma 1, and it follows that the probability of finding such (key, tweak, input)-tuple is upper bounded by 2^{-n} due to the randomness of $\widetilde{\mathsf{F}}$. However, we have to analyze the probability of getting the primitive query $(v_{l-2}, U_{l-1}, u_{l-2}, u_{l-1} \| v_{l-1})$, which we do in the following two subcases:

- Subcase (2a): $(v_{l-2}, U_{l-1}, u_{l-2}, u_{l-1} \| v_{l-1})$ is added to τ_p after $(*, *, *, u_{l-2} \| v_{l-2})$: Let Coll_{2a} represent the event where $\mathcal{A}$ finds a collision in H. In this scenario, similar to Case (1), the collision follows from the right choice of the tweak U_{l-1}, as the adversary has to find a suitable tweak for a given output $u_{l-1} \| v_{l-1}$ with a given key v_{l-2} and input u_{l-2}. By virtue of Lemma 1, the probability of finding such a suitable tweak is upper bounded by 2^{-2n} due to the randomness of $\widetilde{\mathsf{F}}$. Furthermore, there are at most q_p choices for each of $u_{l-2} \| v_{l-2}, u_{l-1} \| v_{l-1}$, and $u_l \| v_l$. Over all possible choices of indices, we have

$$\Pr[\mathsf{Coll}_{2a} | \neg \mathsf{bad}] \leq \frac{q_p^3}{2^{3n}} . \tag{4}$$

- Subcase (2b): $(v_{l-2}, U_{l-1}, u_{l-2}, u_{l-1} \| v_{l-1})$ is added to τ_p before $(*, *, *, u_{l-2} \| v_{l-2})$: Let Coll_{2b} represent the event in which $\mathcal{A}$ finds a collision in H. This scenario is analogous to Case (2). Note that $u_0 \| v_0$ is set to 0^{2n}. By continuing with a similar argument recursively, we reach a point where there exists some $a \in [0, l]$ such that either $u_a \| v_a = 0^{2n}$ or $(*, *, *, u_a \| v_a)$ is added to τ_p before $(v_a, U_{a+1}, u_a, u_{a+1} \| v_{a+1})$. In both cases, the adversary must find a suitable tweak U_{a+1} such that $u_{a+1} \| v_{a+1} \leftarrow \widetilde{\mathsf{F}}^+(v_a, U_{a+1}, u_a, 2)$, where $u_a \| v_a$ and $u_{a+1} \| v_{a+1}$ are given. Then, for a fixed choice of indices, by applying Lemma 1, the probability for finding such a tweak is upper bounded by 2^{-2n} due to the randomness of $\widetilde{\mathsf{F}}$. Furthermore, there are at most q_p choices for each of $u_a \| v_a, u_{a+1} \| v_{a+1}$, and $u_l \| v_l$. Over all possible choices of indices, we have

$$\Pr[\mathsf{Coll}_{2b} | \neg \mathsf{bad}] \leq \frac{q_p^3}{2^{3n}} . \tag{5}$$

The result follows by combining Equations (2) through (5). □

[2] If the collision occurs at some earlier point α, we would have considered the sequence up to α.

4.2 CIML2 Security of FEDT

Theorem 1. Let $K \xleftarrow{\$} \mathcal{K}$ and let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a fork-cipher. Let $\mathcal{A}$ be a $\mathsf{CIML2}$-adversary on $\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K$ that makes at most q_c construction queries and at most q_p ideal-cipher queries. Then, we have

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p) \leq \frac{4q}{2^n} + \frac{q^2}{2^{2n}} + \frac{q^3}{2^{3n}} ,$$

where $q = \max\{q_c, q_p\}$.

Proof. We define the transcript containing all queries and responses of the adversary $\mathcal{A}$ as $\tau = (K, \tau_p, \tau_c, \tau_h, \tau_{\mathsf{KDF}}, \tau_{\mathsf{TGF}})$, where τ_p contains all queries and responses to the ideal fork-cipher $\widetilde{\mathsf{F}}$ including both the ideal-cipher query and the internal state variables generated during the computation of the construction queries in the following way: $(k, J, X, Y\|Z)$, where $Y\|Z \leftarrow \widetilde{\mathsf{F}}^+(k, J, X, 2)$. τ_c contains all construction queries and responses as (K, N, A, M, C, T), where either $(C,T) \leftarrow \mathcal{E}(K, N, A, M)$ or $M \leftarrow \mathcal{D}(K, N, A, C, T)$. Moreover, τ_h consists all queries to the hash function and the corresponding responses in the form $(U, V\|W)$, where $V\|W \leftarrow \mathsf{H}(U)$. The transcript corresponding to the key-derivation function τ_{KDF} consists of all queries and responses to the assumed leak-free key-derivation function as $(K, J, X, Y\|Z)$, where K is the secret key and $Y\|Z \leftarrow \widetilde{\mathsf{F}}^+(K, J, X, 2)$. Finally, τ_{TGF} consists of all queries and responses to the tag generation function as $(K, J, X, Y\|Z)$, where $X \leftarrow \widetilde{\mathsf{F}}^-(K, J, Y, 0, 0)$ or $Y\|Z \leftarrow \widetilde{\mathsf{F}}^+(K, J, X, 2)$. We assume that KDF and TGF do not leak the master secret key K and all other primitives leak all key, input and output. We reveal the master key K to the adversary and add it to the transcript at the end of the interaction. We call a transcript $\tau = (K, \tau_p, \tau_c, \tau_h, \tau_{\mathsf{KDF}}, \tau_{\mathsf{TGF}})$ attainable if the probability of realizing it in the ideal world is non-zero.

Bad Transcripts and Their Probability. We call an attainable transcript $\mathtt{Bad}$ if at least one of the following events occurs:

- $\mathtt{Bad}_1$: $\exists (U_i, V_i\|W_i), (U_j, V_j\|W_j) \in \tau_h$ such that $U_i \neq U_j$ but $(V_i, W_i) = (V_j, W_j)$.
- $\mathtt{Bad}_2$: $\exists (K, J_i, X_i, *) \in \tau_{\mathsf{KDF}}$ and $\exists (K, J_j, 0^n, *) \in \tau_{\mathsf{TGF}}$ such that $X_i = 0^n \wedge J_i = J_j$.
- $\mathtt{Bad}_3$: $\exists (K, J_i, X_i, Y_i\|Z_i) \in \tau_{\mathsf{KDF}}$ and $(k_p, J_p, X_p, Y_p\|Z_p) \in \tau_p$ such that $(K, J_i, X_i) = (k_p, J_p, X_p)$, or $(K, J_i, Y_i\|Z_i) = (k_p, J_p, Y_p\|Z_p)$.
- $\mathtt{Bad}_4$: $\exists (K, J_i, 0^n, Y_i\|Z_i) \in \tau_{\mathsf{TGF}}$ and $(k_p, J_p, X_p, Y_p\|Z_p) \in \tau_p$ such that $(K, J_i, X_i) = (k_p, J_p, X_p)$, or $(K, J_i, Y_i\|Z_i) = (k_p, J_p, Y_p\|Z_p)$.

Note that the TGF uses a strongly protected fork-cipher implementation with a fixed key K and the input 0^n. A collision in the tweak of the TGF for two queries would result in a forgery. The output of the hash function H is employed as the tweak for the TGF. Therefore, $\mathtt{Bad}_1$ is designed to prevent such forgery attempts by removing collisions in the tweak values. Furthermore, guessing the master key K could also lead to a trivial forgery. Since K is utilized exclusively in the KDF and the TGF. To mitigate this risk, $\mathtt{Bad}_3$ and $\mathtt{Bad}_4$ are introduced to address the chance of a forgery from guessing the master key.

Bounding $\mathtt{Bad}_1 \vee \mathtt{Bad}_2$. By definition, $\mathtt{Bad}_1$ essentially says that there is a collision in the hash function as defined in Lemma 2. Moreover, $\mathtt{Bad}_2$ happens if $X_i = N_i = 0^n$ and the adversary can find a hash-function pre-image of $J_i = N_i\|0^n = 0^{2n}$. This condition is equivalent to $\mathtt{bad}$ condition defined on Lemma 2. It follows that

$$\Pr[\mathtt{Bad}_1 \vee \mathtt{Bad}_2] = \Pr[\mathsf{Coll}] \leq \frac{q_p}{2^n} + \frac{q_p^2}{2^{2n}} + \frac{q_p^3}{2^{3n}} . \tag{6}$$

Bounding $\mathtt{Bad}_3$. Among all primitive queries, this bad event will occur if the adversary successfully guesses the master key K since all KDF queries rely on the same master key. Furthermore, there are at most q_p primitive queries. From the randomness of K, we have

$$\Pr[\mathtt{Bad}_3] \leq \frac{q_p}{2^n}\,. \tag{7}$$

Bounding $\mathtt{Bad}_4$. Similar as $\mathtt{Bad}_3$, this bad event will occur if the adversary successfully guesses the master key K since all TGF queries rely on the same master key. Furthermore, there are at most q_p primitive queryies. From the randomness of K, we have

$$\Pr[\mathtt{Bad}_4] \leq \frac{q_p}{2^n}\,. \tag{8}$$

Hence, by applying the union bound on the three cases above, we obtain

$$\Pr[\mathtt{Bad}] \leq \frac{q_p}{2^n} + \frac{q_p^2}{2^{2n}} + \frac{q_p^3}{2^{3n}} + \frac{q_p}{2^n} + \frac{q_p}{2^n} = \frac{3q_p}{2^n} + \frac{q_p^2}{2^{2n}} + \frac{q_p^3}{2^{3n}}\,. \tag{9}$$

Bounding The Forging Advantage Conditioned on $\neg\mathtt{Bad}$. We upper bound the probability of a forgery conditioned on the case that no $\mathtt{Bad}$ events have occurred. A successful forgery implies the existence of a tuple $(K, N, A, M, C\|T) \in \tau_c$, which yields a valid decryption query i.e., $M \neq \perp$. Following the processing of (N, A, C, T), we can find an element $(\mathsf{pad}(A, N, C), V\|W) \in \tau_h$, a $(k_p, J_p, x_p, V\|W) \in \tau_p$, and $(K, V\|W, 0^n, T\|*) \in \tau_{\mathsf{TGF}}$. To bound the probability of this event, we proceed as follows: First, we argue that the tuple $(K, V\|W, 0^n, T\|\star)$ cannot represent a forward query. If it were a forward query, there would exist an encryption query (N', A', M') such that $\mathcal{E}(K, N, A, M) = C', T$. Furthermore, since (N, A, C, T) is a valid decryption query, we have $(N, A, C) \neq (N', A', C')$, which implies that $\mathsf{pad}(A, N, C) \neq \mathsf{pad}(A', N', C')$. This would lead to a collision in the hash function H, which contradicts the assumption of $\neg\mathtt{Bad}_1$. Hence, for the successful forgery, $\mathcal{A}$ must find values for J and T such that $\widetilde{\mathsf{F}}^{-}(K, J, T, 0, 0) = 0^n$, where K is secret, as assumed by $\neg\mathtt{Bad}_3$ and $\neg\mathtt{Bad}_4$, and J must be fresh and distinct from all the tweak values used in the TGF during encryption queries. From the security properties of the primitive fork-cipher, we know that for any (J, T), $\Pr[\widetilde{\mathsf{F}}^{-}(K, J, T, 0, 0) = 0^n] \leq 2^{-n}$. Therefore, we have:

$$\Pr[\mathtt{Forge}|\neg\mathtt{Bad}] \leq \frac{q_c}{2^n}\,. \tag{10}$$

By combining Equations (9) and (10) and using $q = \max\{q_c, q_p\}$, we have

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p) \leq \Pr[\mathtt{Bad}] + \Pr[\mathtt{Forge}|\neg\mathtt{Bad}] \leq \frac{4q}{2^n} + \frac{q^2}{2^{2n}} + \frac{q^3}{2^{3n}}\,,$$

which proves the result. □

4.3 CCAmL2 Security of FEDT

In this section, we establish the $\mathsf{CCAmL2}$ security of FEDT. Before delving into our main theorem, we will explore two security concepts following [BBC+20, BGP+19, Lis21], that will play a crucial role in our analysis. To ensure confidentiality, we will incorporate two additional security notions that address non-invertibility under leakage, as described in Algorithm 6, and indistinguishability under leakage for the XOR function, as outlined in Algorithm 7.

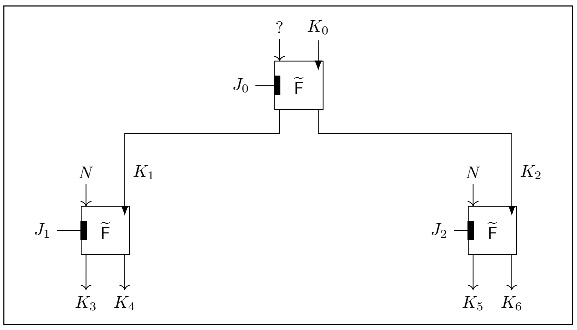


Figure 3: F-LUP setting in $\mathsf{FEDT}[\widetilde{\mathsf{F}}]$.

4.3.1 F-LUP and F-XOR Security Notions

For TEDT [BGP+19], the LUP-2 game was introduced to capture the notion of non-invertibility or unpredictability under leakage for a single iteration of the underlying PRG. Following the result, LUP-4 game was designed in TEDT2 [Lis21] to encompass unpredictability in a more generalized scenario related to the RCTR encryption. In this work, we take a comparable approach and following the notion of LUP-2 and LUP-4, we introduce a similar security game, called F-LUP, which is designed to capture the notion of unpredictability in the context of our tree-based encryption function of FEDT. To ensure that the repetition of same query yields different leakage outcomes, we choose random coins R from a non-empty finite set $\mathcal{R}$ and use it during the computation of the leakage function.

F-LUP. In this game, the adversary $\mathcal{A}$ provides a key K_0 and tweaks J_0, J_1, and J_2 to the challenger. The challenger will choose two random keys K_1 and K_2 and accordingly computes X_1 and X_2. The challenger will compute K_3 and K_4 using K_1, J_1, and N as in the tree-based encryption of FEDT. Moreover, it will also compute K_5 and K_6 using K_2, J_2, and N in a similar way. The adversary will get output leakage only for the computation involving X_1 or X_2. We assume that $\mathcal{A}$ repeats a query at most p times and for each query, it will get K_3, K_4, K_5, K_6, and the corresponding leakages. For the p-time evaluation of a leakage function $\mathfrak{L}$ on inputs X, Y, we write $\{\mathfrak{L}(X,Y)\}_{1..p}$. At the end of the interaction, $\mathcal{A}$ will output a set $\mathcal{K}'$ of cardinality of at most q. We say that $\mathcal{A}$ wins if either $K_1 \in \mathcal{K}'$ or $K_2 \in \mathcal{K}'$. A formal description of the F-LUP game is shown in Algorithm 6 and the computations are depicted in Fig. 3.

Definition 4. Let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a fork-cipher and $\mathfrak{L}^{\text{in}}, \mathfrak{L}^{\text{out}}$ be two leakage functions. Let $\mathcal{A}$ be an adversary playing the F-LUP game in Algorithm 6 against the encryption module of the $\mathsf{FEDT}[\widetilde{\mathsf{F}}]$ construction that provides $K_0 \in \mathcal{K}$ and $J_0, J_1, J_2 \in \{0,1\}^{2n}$ and outputs a set $\mathcal{K}'$. Then, the F-LUP advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}}}(\mathcal{A}) = \Pr\left[\mathcal{A} \text{ wins Game } \mathsf{F\text{-}LUP}\right] .$$

We define $\mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}]}(p,q)$ for the maximum advantage over all F-LUP adversaries on the encryption module of the FEDT construction where $\mathcal{A}$ makes at most p queries and $|\mathcal{K}'| \leq q$.

F-XOR. We also have to study the security of all constituent operations, given that even the slightest information leakage can imperil confidentiality. Like other operations, the XOR operation also warrants careful scrutiny. There exist two predominant approaches

Algorithm 6 Game F-LUP.

```
11: function QUERY($K_0, J_0, J_1, J_2$)
12:     $\mathcal{L} \leftarrow \emptyset$
13:     $K_1 \xleftarrow{\$} \{0,1\}^n$, $K_2 \xleftarrow{\$} \{0,1\}^n$
14:     $X_1 \leftarrow \widetilde{\mathsf{F}}^{-}(K_0, J_0, K_1, 0, 0)$
15:     $X_2 \leftarrow \widetilde{\mathsf{F}}^{-}(K_0, J_0, K_2, 1, 0)$
16:     for $i \leftarrow 1, \ldots, p$ do
17:         $R_1, R_2, R_3, R_4, R_5, R_6 \xleftarrow{\$} \mathcal{R}$
18:         $K_1 \leftarrow \widetilde{\mathsf{F}}^{+}(K_0, J_0, X_1, 0)$
19:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J_0, K_1, 0; R_1)$
20:         $K_2 \leftarrow \widetilde{\mathsf{F}}^{+}(K_0, J_0, X_2, 1)$
21:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J_0, K_2, 1; R_2)$
22:         $K_3 \leftarrow \widetilde{\mathsf{F}}^{+}(K_1, J_1, N, 0)$
23:         $K_4 \leftarrow \widetilde{\mathsf{F}}^{+}(K_1, J_1, N, 1)$
24:         $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\text{in}}(K_1, J_1, N, 0; R_3), \mathfrak{L}^{\text{in}}(K_1, J_1, N, 1, R_4)\}$
25:         $K_5 \leftarrow \widetilde{\mathsf{F}}^{+}(K_2, J_2, N, 0)$
26:         $K_6 \leftarrow \widetilde{\mathsf{F}}^{+}(K_2, J_2, N, 1)$
27:         $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\text{in}}(K_2, J_2, N, 0, ; R_5), \mathfrak{L}^{\text{in}}(K_2, J_2, N, 1; R_6)\}$
28:     return $(K_3, K_4, K_5, K_6, \mathcal{L})$

31: function FINALIZE($\mathcal{K}'$)
32:     return $((K_1 \in \mathcal{K}' \vee K_2 \in \mathcal{K}')$
33:             $\wedge |\mathcal{K}'| \leq q)$
```

for characterizing the security of XOR operations in the presence of leakage. In prior works [BBC+20, BGP+19, GPPS20], a left-or-right approach has been employed, concentrating on individual components in isolation for security analysis. Conversely, a variant known as the real-or-random approach, as introduced by [Lis21], is adopted in this work to maintain an alignment with the FEDT framework. In the real-or-random approach, the evaluation takes place in both the real and the ideal world. In the real world, the challenger processes the original message M, while in the ideal world, the challenger operates on a random message M^* of the same length as M. The adversary wins if it can correctly identify the world it interacts with. Note that in real-or-random notion, we are not computing random function in ideal case. Instead, we encrypt a randomly sampled message using the original construction, eliminating dependencies on the adversary's choice of an alternative message. However, it is easy to see that our proof holds for both the left-or-right and real-or-random notions. A formal definition of the F-XOR game is defined in Algorithm 7.

Definition 5. Let $K \xrightarrow{\$} \mathcal{K}$ and let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a fork-cipher and $\mathfrak{L}^{\oplus}, \mathfrak{L}^{\text{out}}$ be two leakage functions. Let $\mathcal{A}$ be an adversary playing F-XOR game as shown in Algorithm 7 against the encryption module of the FEDT[$\widetilde{\mathsf{F}}$] construction that provides $K_0 \in \mathcal{K},\ J \in \{0,1\}^{2n},\ M_1, M_2 \in \{0,1\}^n$ and output a bit β'. Then, the F-XOR advantage of $\mathcal{A}$ as follows:

$$\mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\text{out}}}(\mathcal{A}) = \left| \Pr\left[\beta = \beta'\right] - \frac{1}{2} \right| .$$

We define $\mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}]}(p, q)$ as the maximum of all F-XOR adversaries $\mathcal{A}$ on FEDT.Enc[$\widetilde{\mathsf{F}}$] that make at most q queries and repeat every query at most p times.

4.3.2 Main Security Theorem

Theorem 2. Let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a fork-cipher. Let $\mathcal{A}$ be a qCCAmL2-adversary on FEDT[$\widetilde{\mathsf{F}}$]$_K$ that makes at most q_c construction queries such that each message consists of at most σ_m blocks. Let q_p be the total number of ideal-cipher queries including the internal state variables generated during the computation of the

Algorithm 7 Game F-XOR.

```
11: procedure INITIALIZE
12:     $\mathcal{L} \leftarrow \emptyset$
13:     $Y \xleftarrow{\$} \{0,1\}^n,\ Z \xleftarrow{\$} \{0,1\}^n$
14:     $\beta \xleftarrow{\$} \{0,1\}$

21: function FINALIZE($\beta'$)
22:     return $\beta = \beta'$

31: function QUERY($K_0, J, M_1, M_2$)
32:     $M_1^*, M_2^* \leftarrow M_1, M_2$
33:     if $\beta = 0$ then
34:         $M_1^*, M_2^* \xleftarrow{\$} \{0,1\}^n \times \{0,1\}^n$
35:     $X_1 \leftarrow \mathsf{F}^{-}(K_0, J, Y, 0, 0)$
36:     $X_2 \leftarrow \mathsf{F}^{-}(K_0, J, Z, 1, 0)$
37:     for $i \leftarrow 1, \ldots, p$ do
38:         $Y \leftarrow \mathsf{F}^{+}(K_0, J, X_1, 0)$
39:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J, Y, 0)$
40:         $Z \leftarrow \mathsf{F}^{+}(K_0, J, X_2, 1)$
41:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J, Z, 1)$
42:         $C \leftarrow (M_1^* \oplus Y) \| (M_2^* \oplus Z)$
43:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\oplus}(Y, M^*)$
44:     return $(C, \mathcal{L})$
```

construction queries. Let $\mathfrak{L}^{\text{in}}$, $\mathfrak{L}^{\text{out}}$, and $\mathfrak{L}^{\oplus}$ be leakage functions as defined in the F-LUP and F-XOR game. Then, for $q = \max\{q_p, q_c\}$, we have

$$\mathbf{Adv}^{\mathsf{qCCAmL2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K, \mathfrak{L}^{\text{in}}, \mathfrak{L}^{\text{out}}, \mathfrak{L}^{\oplus}}(\mathcal{A}) \leq \frac{q}{2^n} + \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\text{in}}, \mathfrak{L}^{\text{out}}}(q_c, q_p) + \sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\text{out}}}(p, q_c) + \mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p)\,.$$

Proof. qCCAmL2 security is defined as a distinguishing notion between a real and an ideal world. Without loss of generality, let us assume the CIML2 security of FEDT as the leakage assumptions are weaker in the qCCAmL2 notion. Otherwise, we can define a CIML2 adversary $\mathcal{B}$ that simply simulates $\mathcal{A}$ and inherits the latter's advantage, which is upper bounded by

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p)\,. \tag{11}$$

In the remainder, we can safely assume that there will be no valid non-challenge decryption query. As any invalid decryption query does not compute the encryption function of FEDT, it does not leak any valid information about the encryption. In absence of valid decryption queries, qCCAmL2 security reduces to the qCPAmL2 setting. Thus, we have to prove the qCPAmL2 security of FEDT in the remainder. For this purpose, a qCPAmL2 adversary can make two kinds of encryption queries: one for information gathering, say $\mathcal{E}_1$, and another type in the challenge phase, say $\mathcal{E}_2$. The challenge decryption oracle $\mathcal{D}_2$ will accept only outputs from $\mathcal{E}_2$ queries as inputs and will output only the leakage corresponding to decryption queries. Non-challenge queries will be processed with the original construction and the submitted message. In the challenge phase, adversary queries will be processed with original construction with the given message or with the original construction but a message sampled independently and uniformly at random by the challenger from all messages of the same length as the input message. Moreover, for nonce-misuse resilience, each nonce of these challenge queries must be distinct from all nonces in other challenge and non-challenge queries.

To upper bound the distinguishing probability, we will introduce two more intermediate games called Ideal(M) and Ideal(\$) as defined in Algorithm 8 the challenge phase. Thus, we define four games $\mathcal{G}_1$ through $\mathcal{G}_4$ in Algorithm 8 as follows:

- $\mathcal{G}_1$: Real(M): Encrypt the given message M with the real encryption algorithm.
- $\mathcal{G}_2$: Ideal(M): Encrypt the given message M with the ideal encryption algorithm.

Algorithm 8 Games $\mathcal{G}_1$ through $\mathcal{G}_4$ in our qCPAmL2 proof. Left: $\mathcal{G}_1$ and $\mathcal{G}_4$, where the difference is that the boxed statements belong only to $\mathcal{G}_4$. Right: $\mathcal{G}_2$ and $\mathcal{G}_3$, where the difference is that the boxed statements belong only to $\mathcal{G}_3$.

```
function $\text{Real}[\widetilde{\mathsf{F}}](K, N, M)$
    $\mathcal{L} \leftarrow \emptyset$
    $\mathcal{Q} \xleftarrow{\cup} \{K\}$
    $K_1 \| K_2 \leftarrow \widetilde{\mathsf{F}}(K, J_0, N, 0)$
    $\mathcal{Q} \xleftarrow{\cup} \{K_1, K_2\}$
    $(M_1, M_2, \ldots, M_m) \xleftarrow{n} M$
    for $i \leftarrow 1, 2, \ldots, m-2$ do
        $J_i \leftarrow N \| [i]_n$
    for $i \leftarrow 4, 6, \ldots, (2m-2)$ do
        $a \leftarrow (i/2) - 1$
        $K_{i-1} \| K_i \leftarrow \widetilde{\mathsf{F}}(K_a, J_a, N, 2)$
        $\mathcal{Q} \xleftarrow{\cup} \{K_{i-1}, K_i\}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{in}(K_a, J_a, N, 2)\}_{1..p}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{out}(K_a, J_a, K_{i-1}, 0)\}_{1..p}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{out}(K_a, J_a, K_i, 1)\}_{1..p}$
    for $j \leftarrow 1, 2, \ldots, m$ do
        $\boxed{M_j \xleftarrow{\$} \{0,1\}^{|M_j|}}$
        $C_j = M_j \oplus K_{m-2+j}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_j, K_{m-2+j}, C_j)\}_{1..p}$
    $C \leftarrow C_1 \| C_2 \| \cdots \| C_m$
    return $(\mathcal{L}, C)$
```

```
function $\text{Ideal}[\widetilde{\mathsf{F}}](K, N, M)$
    $\mathcal{L} \leftarrow \emptyset$
    $\mathcal{Q} \xleftarrow{\cup} \{K\}$
    $(M_1, M_2, \ldots, M_m) \xleftarrow{n} M$
    for $i \leftarrow 1, 2, \ldots, m-2$ do
        $J_i \leftarrow N \| [i]_n$
    for $i \leftarrow 2, 4, 6, \ldots, (2m-2)$ do
        $a \leftarrow (i/2) - 1$
        $K_{i-1} \| K_i \xleftarrow{\$} \{0,1\}^{2n}$
        $\mathcal{Q} \xleftarrow{\cup} \{K_{i-1}, K_i\}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{in}(K_a, J_a, N, 2)\}_{1..p}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{out}(K_a, J_a, K_{i-1}, 0)\}_{1..p}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{out}(K_a, J_a, K_i, 1)\}_{1..p}$
    for $j = 1, 2, \ldots, m$ do
        $\boxed{M_j \xleftarrow{\$} \{0,1\}^{|M_j|}}$
        $C_j = M_j \oplus K_{m-2+j}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_j, K_{m-2+j}, C_j)\}_{1..p}$
    $C \leftarrow C_1 \| C_2 \| \cdots \| C_m$
    return $(\mathcal{L}, C)$
```

- $\mathcal{G}_3$: Ideal(\$): Encrypt a randomly chosen message M^* with the ideal encryption algorithm.
- $\mathcal{G}_4$: Real(\$): Encrypt a randomly chosen message M^* with the real encryption algorithm.

By definition and from the triangle inequality, we obtain that

$$\mathbf{Adv}^{\mathsf{qCPAmL2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K, \mathfrak{L}^{\mathrm{in}}, \mathfrak{L}^{\mathrm{out}}, \mathfrak{L}^{\oplus}}(\mathcal{A}) = \Delta\mathcal{G}_{14} \leq \Delta\mathcal{G}_{12} + \Delta\mathcal{G}_{23} + \Delta\mathcal{G}_{34} \,.$$

Now, we state the following result that bounds the difference between games.

Lemma 3. With the definition of games $\mathcal{G}_1$-$\mathcal{G}_4$, it holds that

$$\Delta\mathcal{G}_{12} + \Delta\mathcal{G}_{23} + \Delta\mathcal{G}_{34} \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\mathrm{in}}, \mathfrak{L}^{\mathrm{out}}}(q_c, q_p) + \frac{q_p}{2^n}$$
$$+ \sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\mathrm{out}}}(p, q_c) \,.$$

From Lemma 3, we obtain that

$$\mathbf{Adv}^{\mathsf{qCPAmL2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K, \mathfrak{L}^{\mathrm{in}}, \mathfrak{L}^{\mathrm{out}}, \mathfrak{L}^{\oplus}}(\mathcal{A}) \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\mathrm{in}}, \mathfrak{L}^{\mathrm{out}}}(q_c, q_p) + \frac{q_p}{2^n}$$
$$+ \sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\mathrm{out}}}(p, q_c) \,,$$

from which our result follows. □

4.3.3 Proof of Lemma 3

It remains to prove Lemma 3. First, we define some bad conditions:

$\texttt{Bad}_1$**:** This happens if the (key, tweak)-pair of an internal primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with a (key, tweak)-pair of another internal primitive query during the same encryption query. Let (N_i, A_i, M_i) be such an encryption query. Let a, b any two indices from $[1, l-1]$, where $|M_i| = ln$ bits. Then, the tweak for the a-th primitive call is $J_a^i = N_i \| a$ and the tweak for the b-th query is $J_b^i = N_i \| b$. From $J_a \neq J_b$, we obtain

$$\Pr[\texttt{Bad}_1] = 0\,.$$

$\texttt{Bad}_2$**:** This happens if the (key, tweak)-pair of a primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with a (key, tweak)-pair of another primitive call during another encryption query. Let (N_i, A_i, M_i) and (N_j, A_j, M_j) be two encryption queries such that the (key, tweak)-pair of the a-th primitive call of the i-th query collides with the (key, tweak)-pair of the b-th primitive call of the j-th query. We have $J_a^i = N_i \| a$ and $J_b^j = N_j \| b$. Moreover, we have $N_i \neq N_j$ due to the unique nonce in each challenge query. Hence

$$\Pr[\texttt{Bad}_2] = 0\,.$$

$\texttt{Bad}_3$**:** This happens if the (key, tweak)-pair of a primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with the (key, tweak)-pair of an ideal primitive query. Let (N_i, A_i, M_i) be the i-th query to the challenge oracle $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$. Let the (key, tweak)-pair of j-th primitive i.e $(k_j, N_i \| j)$ collide with an ideal primitive query. Note that for each primitive call in the challenge oracle, the tweaks are unique. Hence, for any ideal-primitive query, the adversary can target the key of at most one internal primitive call. Since there are at most q_p ideal-primitive queries, we obtain

$$\Pr[\texttt{Bad}_3] \leq \frac{q_p}{2^n}$$

We define an event $\texttt{Bad}$ if any of the above three conditions are satisfied. Hence

$$\Pr[\texttt{Bad}] = \mathsf{Pr}[\texttt{Bad}_1 \cup \texttt{Bad}_2 \cup \texttt{Bad}_3] \leq \frac{q_p}{2^n}\,. \tag{12}$$

The absence of $\texttt{Bad}$ ensures a fresh (key, tweak)-pair for each internal primitive call in the queries to $\mathcal{E}_K^2$. Thus, there is no difference between the real and the ideal game $\mathcal{G}_1$ and $\mathcal{G}_2$ in the black-box scenario. Under leakage, however, there may be. We will show that for an adversary $\mathcal{A}_{12}$ which will try to distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$, the probability of success of $\mathcal{A}_{12}$ is bounded by the maximal advantage of an $\mathsf{F\text{-}LUP}$ distinguisher on the isolated primitive call. In the following, let $\rightarrow$ and $\leftarrow$ indicate for- and backward direction, respectively. Let $\mathcal{A}_{\mathsf{LUP}}$ be an adversary for the $\mathsf{F\text{-}LUP}$ game defined in Algorithm 6 and $\mathcal{A}_{12}$ be the adversary that shall distinguish between Games Real(M) and Ideal(M). $\mathcal{A}_{\mathsf{LUP}}$ simulates $\mathcal{A}_{12}$'s challenger as follows:

- For a primitive query $(\rightarrow, k, J, X, b)$ or $(\leftarrow, k, J, Y_b, b, s)$, $\mathcal{A}_{\mathsf{LUP}}$ replies by quering its own primitive oracle $\tilde{\tilde{\mathsf{F}}}$.
- For a construction query (N_r, A_r, M_r) (in encryption direction) with message $M_r = M_r^1 \parallel M_r^2 \| \cdots \parallel M_r^m$, $\mathcal{A}_{\mathsf{LUP}}$ replies as follows:
  - Choose $K_1, K_2 \xleftarrow{\$} \{0,1\}^n$.
  - Choose $j \in \{1, 2, \ldots, \lceil \frac{m}{2} \rceil - 2\}$.
  - For $i \leftarrow 3, 4, \ldots, j$, $\mathcal{A}_{\mathsf{LUP}}$ will get K_i and the corresponding leakages by querying its primitive oracle with $(\rightarrow, K_a, N \| [a]_n, N, b)$, where $b = (i+1) \bmod 2$ and $a = \lfloor (i-1)/2 \rfloor$.

  - $\mathcal{A}_{\mathsf{LUP}}$ will get K_{2j+1} and K_{2i+2} and corresponding leakages by querying its own primitive oracle with $(\rightarrow, K_j, N_r\|[j]_n, N_r, 2)$.
  - $\mathcal{A}_{\mathsf{LUP}}$ will get $K_{4j+3}, K_{4j+4}, K_{4j+5}, K_{4j+6}$ and corresponding leakages by querying its own F-LUP challenger with $(K_j, N\|[j]_n, N\|[2j+1]_n, N\|[2j+2]_n)$.
  - For all other i's in $[2m-2]$, $\mathcal{A}_{\mathsf{LUP}}$ will get K_i and corresponding leakages by querying its primitive oracle with$(\rightarrow, K_a, N\|[a]_n, N, b)$, where $b = (i+1) \bmod 2$ and $a = \lfloor (i-1)/2 \rfloor$.
  - $\mathcal{A}_{\mathsf{LUP}}$ will compute $C_r^i = M_r^i \oplus K_{m-2+i}$.
  - Send $C_r = C_r^1\|C_r^2\|\cdots\|C_r^m$ and all leakages.

- $\mathcal{A}_{\mathsf{LUP}}$ stores all the primitive queries in the set τ_p.

At the end of an iteration, $\mathcal{A}_{\mathsf{LUP}}$ will output a set of keys used in the primitive queries by adversary $\mathcal{A}_{12}$ as $S = \{k : (k, J, *, *) \in \tau_p \wedge ((J = N\|[j]_n) \vee (J = N\|[2j+1]_n) \vee (J = N\|[2j+2]_n))\}$. The advantage of the adversary $\mathcal{A}_{12}$ will be inherited by $\mathcal{A}_{\mathsf{LUP}}$. Moreover, there are at most σ_m choices for j. Thus, in absence of `Bad`, we obtain

$$\Delta\mathcal{G}_{12} \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F\text{-}LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\mathrm{in}},\mathfrak{L}^{\mathrm{out}}}(q_c, q_p) + \frac{q_p}{2^n}\,. \tag{13}$$

Bounding the Difference between Ideal(M) and Ideal(\$). Let $\mathcal{A}_{\oplus}$ be an adversary for the F-XOR game defined in Algorithm 7 and $\mathcal{A}_{23}$ be an adversary that shall distinguish between the games Ideal(M) and Ideal(\$). $\mathcal{A}_{\oplus}$ simulates the challenger of $\mathcal{A}_{23}$ as follows:

- For a primitive query $(\rightarrow, k, t, X, s)$ or $(\leftarrow, k, t, Y_1\|Y_2, b, s)$, $\mathcal{A}_{\oplus}$ replies by quering its own primitive oracle $\widetilde{\widetilde{\mathsf{F}}}$.
- For a construction query (in encryption direction) with a message $M_r = M_r^1\|M_r^2\| \cdots \| M_r^m$, $\mathcal{A}_{\oplus}$ replies as follows:
  - First, choose $j \in \{m-1, m, \ldots, 2m-2\}$ and $K_0 \in \mathcal{K}$.
  - $\mathcal{A}_{\oplus}$ initializes an empty leakage list $\mathcal{L}$.
  - For $i = 1, 2, \ldots, 2m-2$, $\mathcal{A}_{\oplus}$ will choose $K_i \xleftarrow{\$} \mathcal{K}$ and compute leakages using $\mathfrak{L}^{\oplus}, \mathfrak{L}^{\mathrm{out}}$ as in the F-XOR game.
  - For $i = 1, 2, \ldots, j-1, j+1, \ldots, m$; $\mathcal{A}_{\oplus}$ will compute $C_r^i = M_r^i \oplus K_{m-2+i}$.
  - $\mathcal{A}_{\oplus}$ queries its challenger with M_r^j and get back C_r^j and corresponding leakages.
  - Send $C_r = C_r^1\|C_r^2\|\cdots\|C_r^m$ and all leakages.

At the end, $\mathcal{A}_{\oplus}$ relays the output of $\mathcal{A}_{23}$ to its challenger. As the only difference between the two games is in the j-th position, the advantage of $\mathcal{A}_{23}$ will be inherited by $\mathcal{A}_{\oplus}$. Since we have to consider it for all the message blocks to reduce to the full $\mathcal{A}_{23}$ security, we obtain

$$\Delta\mathcal{G}_{23} \leq \sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\oplus},\mathfrak{L}^{\mathrm{out}}}(p, q_c)\,. \tag{14}$$

Bounding the Difference between Ideal(\$) and Real(\$). It is easy to see that distinguishing between Ideal(\$) and Real(\$) – that is between games $\mathcal{G}_3$ and $\mathcal{G}_4$ – is the same setting as distinguishing between Ideal(M) and Real(M), i.e. between games $\mathcal{G}_1$ and $\mathcal{G}_2$. Thus,

$$\Delta\mathcal{G}_{34} \leq \Delta\mathcal{G}_{12}\,. \tag{15}$$

The result follows by combining Equations (13), (14), and (15).

5 Security Analysis of FEDT*

5.1 CIML2 security of FEDT*

Theorem 3. Let $K \xrightarrow{\$} \mathcal{K}$ and $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be a fork-cipher. Let $\mathcal{A}$ be a CIML2-adversary on FEDT*$[\widetilde{\mathsf{F}}]_K$ that makes at most q_c construction queries and at most q_p ideal-cipher queries. Then, for $q = \max\{q_c, q_p\}$, we have

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT*}[\widetilde{\mathsf{F}}]_K}(q_c, q_p) \leq \frac{4q}{2^n} + \frac{q^2}{2^{2n}} + \frac{q^3}{2^{3n}} .$$

Proof. Similarly as for FEDT, one can choose a suitable message to get any desired ciphertext for a given nonce. Therefore, the forgery security of FEDT* can be reduced to that of its authentication module, which is identical to that of FEDT. Consequently, the CIML proof of FEDT applies to FEDT* as well. □

5.2 CCAmL2 security of FEDT*

In this section, we establish the CCAmL2 security of FEDT*. Similar as FEDT, to ensure confidentiality, we need two auxiliary notions that address non-invertibility under leakage, as described in figure. 4, and indistinguishability under leakage for the XOR function, as outlined in Algorithm. 7. For the former, we will define a F*-LUP game, similarly as for FEDT. For leakage during XORing, we will use the same F-XOR game as FEDT.

5.2.1 The F*-LUP Security Notion

In this game, the adversary $\mathcal{A}$ provides a key K_0 and tweaks J_0, J_1, J_2, and J_3 to the challenger. The challenger will choose two random keys K_1, K_2 and accordingly computes X_1, X_2. The challenger will compute K_3, and K_4 using K_1, J_1, and N as in the encryption of FEDT*. Moreover, it will also compute Y_1 and Y_2 using K_2, J_2, and N in a similar way. It will also compute Y_3 and Y_4 using K_2, J_3, and N in a similar way. The adversary will get output leakage only for the computation involving X_1 or X_2. We assume that $\mathcal{A}$ repeats a query at most p times and for each query, it will get K_3, K_4, Y_1, Y_2, Y_3, Y_4 and the corresponding leakages. At the end of the interaction, $\mathcal{A}$ will output a set $\mathcal{K}'$ of cardinality of at most q. We say that $\mathcal{A}$ wins if either $K_1 \in \mathcal{K}'$ or $K_2 \in \mathcal{K}'$. A formal description of the F-LUP game is shown in Algorithm 9 and the game is pictorially depicted in Fig. 4.

Definition 6 (F*-LUP)**.** Let $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be an ideal fork-cipher and $\mathfrak{L}^{\text{in}}$ and $\mathfrak{L}^{\text{out}}$ be two leakage functions. Let $\mathcal{A}$ be an adversary playing the F*-LUP game as shown in Algorithm 9 against the encryption module of the FEDT*$[\widetilde{\mathsf{F}}]$ construction that provides $K_0 \in \mathcal{K}$ and $J_0, J_1, J_2, J_3 \in \{0,1\}^{2n}$ and output a set $\mathcal{K}'$. Then, the F*-LUP advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{F^*\text{-}LUP}}_{\mathsf{FEDT*.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}}}(\mathcal{A}) = \Pr\left[\mathcal{A} \text{ wins Game } \mathsf{F^*\text{-}LUP}\right] .$$

We define $\mathbf{Adv}^{\mathsf{F^*\text{-}LUP}}_{\mathsf{FEDT*.Enc}[\widetilde{\mathsf{F}}]}(p, q)$ for the maximum advantage over all F*-LUP adversaries on the encryption module of the FEDT* construction that makes at most p queries and $|\mathcal{K}'| \leq q$.

5.2.2 Main Security Result

Theorem 4. Let $K \xrightarrow{\$} \mathcal{K}$ and $\widetilde{\mathsf{F}} : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ be an ideal fork-cipher. Let $\mathcal{A}$ be a qCCAmL2-adversary on FEDT*$[\widetilde{\mathsf{F}}]_K$ that makes at most q_c construction

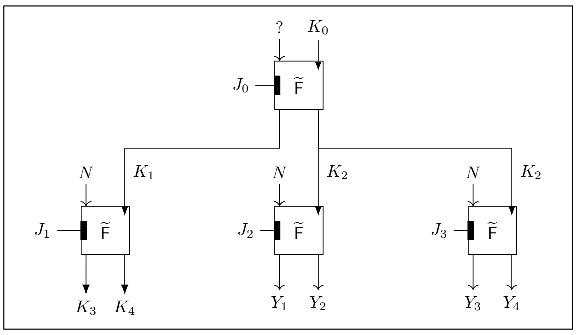


Figure 4: F^*-LUP setting in $\mathsf{FEDT^*}[\widetilde{\mathsf{F}}]$.

Algorithm 9 Game F^*-LUP for FEDT*.

```
11: function QUERY($K_0, J_0, J_1, J_2, J_3$)
12:     $\mathcal{L} \leftarrow \emptyset$
13:     $K_1 \xleftarrow{\$} \{0,1\}^n$, $K_2 \xleftarrow{\$} \{0,1\}^n$
14:     $X_1 \leftarrow \widetilde{\mathsf{F}}^-(K_0, J_0, K_1, 0, 0)$
15:     $X_2 \leftarrow \widetilde{\mathsf{F}}^-(K_0, J_0, K_2, 1, 0)$
16:     for $i \leftarrow 1 \dots p$ do
17:         $R_1, R_2, R_3, R_4, R_5, R_6 \xleftarrow{\$} \mathcal{R}$
18:         $K_1 \leftarrow \widetilde{\mathsf{F}}^+(K_0, J_0, X_1, 0)$
19:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J_0, K_1, 0; R_1)$
20:         $K_2 \leftarrow \widetilde{\mathsf{F}}^+(K_0, J_0, X_2, 1)$
21:         $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\text{out}}(K_0, J_0, K_2, 1; R_2)$
22:         $K_3 \leftarrow \widetilde{\mathsf{F}}^+(K_1, J_1, N, 0)$
23:         $K_4 \leftarrow \widetilde{\mathsf{F}}^+(K_1, J_1, N, 1)$
24:         $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\text{in}}(K_1, J_1, N, 0; R_3), \mathfrak{L}^{\text{in}}(K_1, J_1, N, 1; R_4)\}$
25:         $Y_1 \leftarrow \widetilde{\mathsf{F}}^+(K_2, J_2, N, 0)$
26:         $Y_2 \leftarrow \widetilde{\mathsf{F}}^+(K_2, J_2, N, 1)$
27:         $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\text{in}}(K_2, J_2, N, 0, ; R_5), \mathfrak{L}^{\text{in}}(K_2, J_2, N, 1; R_6)\}$
28:         $Y_3 \leftarrow \widetilde{\mathsf{F}}^+(K_2, J_3, N, 0)$
29:         $Y_4 \leftarrow \widetilde{\mathsf{F}}^+(K_2, J_3, N, 1)$
30:         $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\text{in}}(K_2, J_3, N, 0, ; R_7), \mathfrak{L}^{\text{in}}(K_2, J_3, N, 1; R_8)\}$
31:     return ($K_3, K_4, K_5, Y_1, Y_2, Y_3, Y_4, \mathcal{L}$)

41: function FINALIZE($\mathcal{K}'$)
42:     return (($K_1 \in \mathcal{K}' \vee K_2 \in \mathcal{K}'$)
43:             $\wedge |\mathcal{K}'| \leq q$)
```

queries such that each message consists of at most σ_m blocks. Let q_p be the total number of ideal-cipher queries including the internal state variables generated during the computation of the construction queries. Let $\mathfrak{L}^{\text{in}}$, $\mathfrak{L}^{\text{out}}$, and $\mathfrak{L}^{\oplus}$ be leakage functions as defined in the F^*-LUP and F-XOR games. Then, for $q = \max\{q_p, q_c\}$, we have

$$\mathbf{Adv}^{\mathsf{qCCAmL2}}_{\mathsf{FEDT^*}[\widetilde{\mathsf{F}}]_K, \mathfrak{L}^{\text{in}}, \mathfrak{L}^{\text{out}}, \mathfrak{L}^{\oplus}}(\mathcal{A}) \leq \frac{q}{2^n} + \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F}^*\text{-}\mathsf{LUP}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\text{in}}, \mathfrak{L}^{\text{out}}}(q_c, q_p) +$$
$$\sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\text{in}}}(p, q_c) + \mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p) .$$

Proof. qCCAmL2 security is defined as distinguishing security between real and random. Without loss of generality, assume the CIML2 security of FEDT* as the leakage assumptions are weaker in the qCCAmL2 notion. Otherwise, we can define a CIML2 adversary $\mathcal{B}$ that simply simulates $\mathcal{A}$ and inherits the latter's advantage, which is upper bounded by

$$\mathbf{Adv}^{\mathsf{CIML2}}_{\mathsf{FEDT}[\widetilde{\mathsf{F}}]_K}(q_c, q_p) . \tag{16}$$

In the remainder, we can safely assume that there will be no valid non-challenge decryption query. As any invalid decryption query does not compute the FEDT*.Enc function of the encryption module of FEDT*, it does not leak any valid information about encryption. In absence of valid decryption queries, qCCAmL2 security reduces to the qCPAmL2 setting. Thus, we have to prove the qCPAmL2 security of FEDT* in the remainder. For this purpose, a qCPAmL2 adversary can make two kinds of encryption queries: one for information gathering, say $\mathcal{E}_1$, and another type in the challenge phase, say $\mathcal{E}_2$. The challenge decryption oracle $\mathcal{D}_2$ will accept as inputs only outputs from $\mathcal{E}_2$ queries and will output only the leakage corresponding to the decryption. Non-challenge queries will be processed with the original construction and the submitted message. In the challenge phase, adversary queries will be processed with the original construction with the given message or with the original construction but with a message sampled independently and uniformly at random by the challenger from all messages of the same length as the input message. Moreover, for nonce-misuse resilience, each nonce of these challenge queries must be unique from other challenge and non-challenge queries. To upper bound the distinguishing probability, we will introduce two intermediate games called Ideal(M) and Ideal($) as defined in Algorithm 10 the challenge phase. Thus, we define four games $\mathcal{G}_1$ through $\mathcal{G}_4$ in Algorithm 10 as follows:

- $\mathcal{G}_1$: Real(M): Encrypt the given message M with the real encryption algorithm.
- $\mathcal{G}_2$: Ideal(M): Encrypt the given message M with the ideal encryption algorithm.
- $\mathcal{G}_3$: Ideal($): Encrypt a randomly chosen message M^* with the ideal encryption algorithm.
- $\mathcal{G}_4$: Real($): Encrypt a randomly chosen message M^* with the real encryption algorithm.

By definition and from the triangle inequality, we obtain that

$$\mathbf{Adv}^{\mathsf{qCPAmL2}}_{\mathsf{FEDT^*}[\widetilde{\mathsf{F}}]_K,\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}},\mathfrak{L}^{\oplus}}(\mathcal{A}) = \Delta\mathcal{G}_{14} \leq \Delta\mathcal{G}_{12} + \Delta\mathcal{G}_{23} + \Delta\mathcal{G}_{34}\,.$$

The following result bounds the difference between the games.

Lemma 4. With the definition of games $\mathcal{G}_1$-$\mathcal{G}_4$, it holds that

$$\begin{aligned}\Delta\mathcal{G}_{12} + \Delta\mathcal{G}_{23} + \Delta\mathcal{G}_{34} \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F^*\text{-}LUP}}_{\mathsf{FEDT^*.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}}}(q_c, q_p) + \frac{q_p}{2^n} \\ + \sigma_m \cdot \mathbf{Adv}^{\text{XOR}}_{\mathsf{FEDT^*.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\oplus},\mathfrak{L}^{\text{out}}}(p, q_c)\,.\end{aligned}$$

From Lemma 4, we obtain that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{qCPAmL2}}_{\mathsf{FEDT^*}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}},\mathfrak{L}^{\oplus}}(\mathcal{A}) \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F^*\text{-}LUP}}_{\mathsf{FEDT^*.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\text{in}},\mathfrak{L}^{\text{out}}}(q_c, q_p) + \frac{q_p}{2^n} \\ + \sigma_m \cdot \mathbf{Adv}^{\text{XOR}}_{\mathsf{FEDT^*.Enc}[\widetilde{\mathsf{F}}],\mathfrak{L}^{\oplus},\mathfrak{L}^{\text{out}}}(p, q_c)\,.\end{aligned}$$

and hence the result follows. □

5.2.3 Proof of Lemma 4

It only remains to prove Lemma 4. First, we define some bad conditions:

Algorithm 10 FEDT*: Games $\mathcal{G}_1$ through $\mathcal{G}_4$ in our qCPAmL2 proof. Left: $\mathcal{G}_1$ and $\mathcal{G}_4$, where the difference is that the boxed statements belong only to $\mathcal{G}_4$. Right: $\mathcal{G}_2$ and $\mathcal{G}_3$, where the difference is that the boxed statements belong only to $\mathcal{G}_3$.

```
function REAL[$\widetilde{\mathsf{F}}$]$(K, N, M)$
    $\mathcal{L} \leftarrow \emptyset$
    $\mathcal{Q} \xleftarrow{\cup} \{K\}$
    $k_1 \| k_2 \leftarrow \widetilde{\mathsf{F}}(K, N \| 0^n, N)$
    $\mathcal{Q} \xleftarrow{\cup} \{k_1, k_2\}$
    $l \leftarrow \lceil |M|/n \rceil$
    $M_1 \| M_2 \| \cdots \| M_l \leftarrow M$
    for $i \leftarrow 1, 2, 3, \ldots, l/4$ do
        $k_{2i+1} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i-1}, J_i^0, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i-1}, J_i^0, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i-1}, J_i^0, k_{2i+1}, 0)$
        $k_{2i+2} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i-1}, J_i^0, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i-1}, J_i^0, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i-1}, J_i^0, k_{2i+2}, 1)$
        $Y_{4i-3} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i}, J_i^1, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^1, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^1, Y_{4i-3}, 0)$
        $\boxed{M_{4i-3} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-3} = M_{4i-3} \oplus Y_{4i-3}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-3}, Y_{4i-3}, C_{4i-3})\}_{1..p}$
        $Y_{4i-2} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i}, J_i^1, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^1, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^1, Y_{4i-2}, 1)$
        $\boxed{M_{4i-2} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-2} = M_{4i-2} \oplus Y_{4i-2}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-2}, Y_{4i-2}, C_{4i-2})\}_{1..p}$
        $Y_{4i-1} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i}, J_i^2, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^2, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^2, Y_{4i-1}, 0)$
        $\boxed{M_{4i-1} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-1} = M_{4i-1} \oplus Y_{4i-1}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-1}, Y_{4i-1}, C_{4i-1})\}_{1..p}$
        $Y_{4i} \leftarrow \widetilde{\mathsf{F}}^+(k_{2i}, J_i^2, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^2, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^2, Y_{4i}, 1)$
        $\boxed{M_{4i} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i} = M_{4i} \oplus Y_{4i}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i}, Y_{4i}, C_{4i})\}_{1..p}$
    $C \leftarrow C_1 \| C_2 \| \cdots \| C_l$
    return $(\mathcal{L}, C)$
```

```
function IDEAL[$\widetilde{\mathsf{F}}$]$(K, N, M)$
    $\mathcal{L} \leftarrow \emptyset$
    $\mathcal{Q} \xleftarrow{\cup} \{K\}$
    $k_1 \| k_2 \xleftarrow{\$} \{0,1\}^{2n}$
    $\mathcal{Q} \xleftarrow{\cup} \{k_1, k_2\}$
    $l \leftarrow \lceil |M|/n \rceil$
    $M_1 \| M_2 \| \cdots \| M_l \leftarrow M$
    for $i = 1, 2, 3, \ldots, l/4$ do
        $k_{2i+1} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i-1}, J_i^0, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i-1}, J_i^0, k_{2i+1}, 0)$
        $k_{2i+2} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i-1}, J_i^0, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i-1}, J_i^0, k_{2i+2}, 1)$
        $Y_{4i-3} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^1, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^1, Y_{4i-3}, 0)$
        $\boxed{M_{4i-3} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-3} = M_{4i-3} \oplus Y_{4i-3}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-3}, Y_{4i-3}, C_{4i-3})\}_{1..p}$
        $Y_{4i-2} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^1, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^1, Y_{4i-2}, 1)$
        $\boxed{M_{4i-2} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-2} = M_{4i-2} \oplus Y_{4i-2}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-2}, Y_{4i-2}, C_{4i-2})\}_{1..p}$
        $Y_{4i-1} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^2, N, 0)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^2, Y_{4i-1}, 0)$
        $\boxed{M_{4i-1} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i-1} = M_{4i-1} \oplus Y_{4i-1}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i-1}, Y_{4i-1}, C_{4i-1})\}_{1..p}$
        $Y_{4i} \xleftarrow{\$} \{0,1\}^n$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{in}}(k_{2i}, J_i^2, N, 1)$
        $\mathcal{L} \xleftarrow{\cup} \mathfrak{L}^{\mathrm{out}}(k_{2i}, J_i^2, Y_{4i}, 1)$
        $\boxed{M_{4i} \xleftarrow{\$} \{0,1\}^n}$
        $C_{4i} = M_{4i} \oplus Y_{4i}$
        $\mathcal{L} \xleftarrow{\cup} \{\mathfrak{L}^{\oplus}(M_{4i}, Y_{4i}, C_{4i})\}_{1..p}$
    $C \leftarrow C_1 \| C_2 \| \cdots \| C_l$
    return $(\mathcal{L}, C)$
```

$\mathtt{Bad}_1$**:** This event happens if the (key, tweak)-pair of an internal primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with a (key, tweak)-pair of another internal primitive query during the same encryption query. Note that, tweak of any internal primitive is defined as $N \| [i]_{n-8} \| [d]_8$, where i represents the index of each level processing four message blocks and d is the domain separator for the primitive calls on the same level. Clearly, the tweaks of every two primitive queries during the same construction query differs either by the index i or by the domain separator (0, 1, or 2). Therefore

$$\Pr[\mathtt{Bad}_1] = 0\,.$$

$\mathtt{Bad}_2$**:** This event happens if the (key, tweak)-pair of a primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with a (key, tweak)-pair of another primitive call during another encryption query. Let (N_a, A_a, M_a) and (N_b, A_b, M_b) be two encryption queries such that the (key, tweak)-pair of the i-th primitive call of the a-th construction query collides with the (key, tweak)-pair of the j-th primitive call of the b-th construction query. We have $N_a \| [i]_{n-8} \| [d]_8 = N_b \| [j]_{n-8} \| [d']_8$. Moreover, we have $N_a \neq N_b$ due to the unique nonce in each challenge query. Hence

$$\Pr[\mathtt{Bad}_2] = 0.$$

$\mathtt{Bad}_3$**:** This event happens if the (key, tweak)-pair of a primitive call during some challenge query to $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$ collides with the (key, tweak)-pair of an ideal-primitive query. Let (N_a, A_a, M_a) be the a-th query to the challenge oracle $\mathcal{E}_K^2$ or $\$_{\mathcal{E}}$. Let the (key, tweak)-pair of the i-th primitive call i.e $(k_i, N_a \| [i]_{n-8} \| [d]_8)$ collide with an ideal-primitive query. Note that for each primitive call in the challenge oracle, the tweaks are unique. Hence, for any ideal-primitive query, the adversary can target the key of at most one internal primitive call. Since there are at most q_p ideal-primitive queries, we obtain

$$\Pr[\mathtt{Bad}_3] \leq \frac{q_p}{2^n}.$$

We define an event $\mathtt{Bad}$ that is true if and only if any of the above three conditions is satisfied. Hence

$$\Pr[\mathtt{Bad}] = \mathsf{Pr}[\mathtt{Bad}_1 \cup \mathtt{Bad}_2 \cup \mathtt{Bad}_3] \leq \frac{q_p}{2^n}\,. \tag{17}$$

The absence of $\mathtt{Bad}$ ensures a fresh (key, tweak)-pair for each internal primitive call in the queries to $\mathcal{E}_K^2$. Thus, there is no difference between the real and the ideal game $\mathcal{G}_1$ and $\mathcal{G}_2$ in the black-box scenario. Under leakage, however, there may be a difference. We will show that for an adversary $\mathcal{A}_{12}$ that tries to distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$, the probability of success of $\mathcal{A}_{12}$ is bounded by the maximal advantage of an F^*-LUP distinguisher $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ on the isolated primitive call. In the following, let $\rightarrow$ and $\leftarrow$ indicate forward and backward direction, respectively. Let $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ be an adversary for the F^*-LUP game defined in Algorithm 9 and $\mathcal{A}_{12}$ be the adversary that shall distinguish between Games Real(M) and Ideal(M). $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ simulates $\mathcal{A}_{12}$'s challenger as follows:

- For a primitive query $(\rightarrow, k, J, X, b)$ or $(\leftarrow, k, J, Y_b, b, s)$, $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ replies by quering its own primitive oracle $\tilde{\tilde{\mathsf{F}}}$.
- For a construction query (N_r, A_r, M_r) (in encryption direction) with message $M_r = M_r^1 \parallel M_r^2 \| \cdots \parallel M_r^m$, $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ replies as follows:
  - Choose $k_1, k_2 \xleftarrow{\$} \{0,1\}^n$.
  - Choose $j \in \{2, \ldots, \frac{m}{4}\}$.
  - For $i \leftarrow 1, 2, \ldots, j-1$, $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ will get $k_{2i+1}, k_{2i+2}, Y_{4i-3}, Y_{4i-2}, Y_{4i-1}, Y_{4i}$ and corresponding leakages by querying its primitive oracle with keys k_{2i-1}, k_{2i} and tweaks $N_r \| [i]_{n-8} \| [d]_8$.
  - $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ receives $k_{2j+1}, k_{2j+2}, Y_{4j-3}, Y_{4j-2}, Y_{4j-1}, Y_{4j}$ and corresponding leakages by querying its primitive oracle with keys k_{2j-1}, k_{2j} and tweaks $N_r \parallel [j]_{n-8} \parallel [d]_8$.
  - $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ will get $k_{2j+3}, k_{2j+4}, Y_{4j+1}, Y_{4j+2}, Y_{4j+3}, Y_{4j+4}$ by querying its own oracle with $(k_{2j-3}, N_j \| [j]_{n-8} \| [0]_8, N_j \| [j]_{n-8} \| [1]_8, N_j \| [j]_{n-8} \| [2]_8)$.

  - For all other i's in $[\frac{m}{4}]$, $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ will get k_i and corresponding leakages by following same query as $i \in [1, j-1]$.
  - $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ will compute $C_r^i = M_r^i \oplus Y_i, \ \forall\, i \in [1, m]$.
  - Send $C_r = C_r^1 \| C_r^2 \| \cdots \| C_r^m$ and all leakages.

- $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ stores all the primitive queries in the set τ_p.

At the end of an iteration, $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$ will output a set of keys used in the primitive queries by adversary $\mathcal{A}_{12}$ as $S = \{k : (k, J, *, *) \in \tau_p \wedge (J = N \| [j]_{n-8} \| [d]_8), d \in \{0, 1, 2\}\}$. The advantage of the adversary $\mathcal{A}_{12}$ will be inherited by $\mathcal{A}_{\mathsf{F}^*\text{-}\mathsf{LUP}}$. Moreover, there are at most σ_m choices for index j. Thus, in absence of `Bad`, we obtain

$$\Delta\mathcal{G}_{12} \leq \sum_{i=1}^{\sigma_m} \mathbf{Adv}^{\mathsf{F}^*\text{-}\mathsf{LUP}}_{\mathsf{FEDT^*.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\mathrm{in}}, \mathfrak{L}^{\mathrm{out}}}(q_c, q_p) + \frac{q_p}{2^n} \,. \tag{18}$$

Bounding the Difference between Ideal(M) and Ideal(\$). Let $\mathcal{A}_\oplus$ be an adversary for the $\mathsf{F\text{-}XOR}$ game defined in Algorithm 7 and $\mathcal{A}_{23}$ be an adversary that shall distinguish between the games Ideal(M) and Ideal(\$). $\mathcal{A}_\oplus$ simulates the challenger of $\mathcal{A}_{23}$ as follows:

- For a primitive query $(\rightarrow, k, t, X, s)$ or $(\leftarrow, k, t, Y_1 \| Y_2, b, s)$, $\mathcal{A}_\oplus$ replies by querying its own primitive oracle $\widetilde{\widetilde{\mathsf{F}}}$.

- For a construction query (in encryption direction) with a message $M_r = M_r^1 \| M_r^2 \| \cdots \| M_r^m$, $\mathcal{A}_\oplus$ replies as follows:
  - First, choose $j \in [1, m/4]$ and $k_0 \in \{0, 1\}^n$.
  - $\mathcal{A}_\oplus$ initializes an empty leakage list $\mathcal{L}$.
  - For $i = 1, 2, \ldots, 4j-4, 4j-1, \ldots, m$, $\mathcal{A}_\oplus$ will compute C_r^i by sampling Y_i's randomly and xoring with corresponding message block and compute leakages using $L^{\mathrm{in}}, L^{\mathrm{out}}$ as in the $\mathsf{F\text{-}LUP}$ game.
  - $\mathcal{A}_\oplus$ queries its challenger with key k_{2j}, tweak $N_r \| [j]_{n-8} \| [1]_8$, two message block M_r^{4j-3}, M_r^{4j-2} and get back C_r^{4j-3}, C_r^{4j-2} and corresponding leakages.
  - Send $C_r = C_r^1 \| C_r^2 \| \cdots \| C_r^m$ and all leakages.

At the end, $\mathcal{A}_\oplus$ relays the output of $\mathcal{A}_{23}$ to its challenger. As the only difference between the two games is in the j-th position, the advantage of $\mathcal{A}_{23}$ will be inherited by $\mathcal{A}_\oplus$. Since we have to consider it for all the message blocks to reduce to the full $\mathcal{A}_{23}$ security, we obtain

$$\Delta\mathcal{G}_{23} \leq \sigma_m \cdot \mathbf{Adv}^{\mathsf{F\text{-}XOR}}_{\mathsf{FEDT.Enc}[\widetilde{\mathsf{F}}], \mathfrak{L}^{\oplus}, \mathfrak{L}^{\mathrm{out}}}(p, q_c) \,. \tag{19}$$

Bounding the Difference between Ideal(\$) and Real(\$). It is easy to see that distinguishing between Ideal(\$) and Real(\$) – that is between games $\mathcal{G}_3$ and $\mathcal{G}_4$ – is the same setting as distinguishing between Ideal(M) and Real(M), i.e. between games $\mathcal{G}_1$ and $\mathcal{G}_2$. Thus,

$$\Delta\mathcal{G}_{34} \leq \Delta\mathcal{G}_{12} \,. \tag{20}$$

The result follows from combining Euations (18), (19), and (20).

6 Conclusion

This work introduced two leakage-resilient AEAD schemes, FEDT and FEDT*, which utilize fork-ciphers as a primitive. FEDT and FEDT* achieve state-of-the-art throughput, beyond-birthday-bound security akin to previous constructions like TEDT and TEDT2, and propose a new hash function inspired by the TBC-based double-block-length hashing. While recent advancements indicate that efficiency can be enhanced with a larger tweakey, the exploration of more efficient hash functions based on fork-ciphers remains an interesting open problem. Comparing the performance of our proposed schemes with existing ones through concrete implementations is left as an open problem. Moreover, FEDT and FEDT* provide Grade-3 security and thus set the stage for future endeavors, with a new avenue being the construction of a Grade-2 leakage-resilient AEAD security based on fork-ciphers.

Acknowledgments

We thank the reviewers of IACR Communications in Cryptology for their fruitful comments. Eik List has been supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – LI 4223/1-1.

References

[ABL+14] 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. doi:10.1007/978-3-662-45611-8_6.

[ALP+19] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A New Primitive for Authenticated Encryption of Very Short Messages. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT II*, volume 11922 of *Lecture Notes in Computer Science*, pages 153–182. Springer, 2019. doi:10.1007/978-3-030-34621-8_6.

[BBC+20] 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. doi:10.1007/978-3-030-56784-2_13.

[BBKN01] 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. doi:10.1007/3-540-44647-8_18.

[Ber14] Daniel J. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. last update 20 Feb 2019, last accessed 18 July 2023, 2014. URL: https://competitions.cr.yp.to/caesar.html.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Strong Authenticity with Leakage Under Weak and Falsifiable Physical Assumptions. In Zhe Liu and Moti Yung, editors, *Inscrypt*, volume

12020 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2019. doi:10.1007/978-3-030-42921-8_31.

[BGP+20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a Leakage-Resistant AEAD Mode for High Physical Security Applications. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):256–320, 2020. doi:10.13154/tches.v2020.i1.256-320.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient Leakage-Resilient MACs Without Idealized Assumptions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT II*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021. doi:10.1007/978-3-030-92075-3_4.

[BKP+18] 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. doi:10.1145/3196494.3196525.

[BMOS17] 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. doi:10.1007/978-3-319-70694-8_24.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000. doi:10.1007/3-540-44448-3_41.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Transactions on Symmetric Cryptology*, 2017(3):271–293, 2017. doi:10.13154/tosc.v2017.i3.271-293.

[BPS19] Francesco Berti, Olivier Pereira, and François-Xavier Standaert. Reducing the Cost of Authenticity with Leakages: a CIML2-Secure AE Scheme with One Call to a Strongly Protected Tweakable Block Cipher. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi, editors, *AFRICACRYPT*, volume 11627 of *Lecture Notes in Computer Science*, pages 229–249. Springer, 2019. doi:10.1007/978-3-030-23696-0_12.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-Security in Private-Key Cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003. doi:10.1007/3-540-36563-X_1.

[CDD+19] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR Transactions on Symmetric Cryptology*, 2019(4):119–146, 2019. doi:10.13154/tosc.v2019.i4.119-146.

[CJRR99] 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. doi:10.1007/3-540-48405-1_26.

[DDLM23] Nilanjan Datta, Avijit Dutta, Eik List, and Sougata Mandal. On the Security of Triplex- and Multiplex-Type Constructions with Smaller Tweaks. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *INDOCRYPT I*, volume 14459 of *Lecture Notes in Computer Science*, pages 25–47. Springer, 2023. doi:10.1007/978-3-031-56232-7_2.

[DEM$^+$17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – Towards Side-Channel Secure Authenticated Encryption. *IACR Transactions on Symmetric Cryptology*, 2017(1):80–105, 2017. doi:10.13154/tosc.v2017.i1.80-105.

[DEM$^+$20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Transactions on Symmetric Cryptology*, 2020(S1):390–416, 2020. doi:10.13154/tosc.v2020.iS1.390-416.

[DNT19] 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. doi:10.1007/978-3-030-17653-2_15.

[Dwo04] 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. doi:10.6028/NIST.SP.800-38C.

[Dwo07] 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. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf.

[GKP20] Chun Guo, Mustafa Khairallah, and Thomas Peyrin. AET-LR: Rate-1 Leakage-Resilient AEAD based on the Romulus Family. In *NIST LWC Workshop*, 2020. last accessed 24 June 2024. URL: https://csrc.nist.rip/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/AET-LR-lwc2020.pdf.

[GP99] 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. doi:10.1007/3-540-48059-5_15.

[GPPS18] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated Encryption with Nonce Misuse and Physical Leakages: Definitions, Separation Results, and Leveled Constructions. Cryptology ePrint Archive, Paper 2018/484, 2018. URL: https://eprint.iacr.org/2018/484.

[GPPS19] 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. doi:10.1007/978-3-030-30530-7_8.

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. *IACR Transactions on Symmetric Cryptology*, 2020(1):6–42, 2020. doi:10.13154/tosc.v2020.i1.6-42.

[GSF13] 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. doi:10.1007/978-3-642-40349-1_23.

[Hir06] 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. doi:10.1007/11799313_14.

[HOM06] 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. doi:10.1007/11767480_16.

[HPY07] Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007. doi:10.1007/978-3-540-76900-2_7.

[HRRV15] 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. doi:10.1007/978-3-662-47989-6_24.

[KJJ99] 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. doi:10.1007/3-540-48405-1_25.

[KR16] Ted Krovetz and Phillip Rogaway. OCB (v1.1), 2016. URL: https://competitions.cr.yp.to/round3/ocbv11.pdf.

[Lis21] 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. doi:10.1007/978-3-030-88238-9_14.

[MV04] 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. doi:10.1007/978-3-540-30556-9_27.

[Nai19] Yusuke Naito. Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT*, volume

11774 of *Lecture Notes in Computer Science*, pages 65–85. Springer, 2019. doi:10.1007/978-3-030-30530-7_4.

[Ost90] Rafail Ostrovsky. Efficient Computation on Oblivious RAMs. In Harriet Ortiz, editor, *STOC*, pages 514–523. ACM, 1990. doi:10.1145/100216.100289.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *CCS*, pages 96–108. ACM, 2015. doi:10.1145/2810103.2813626.

[RBBK01] 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. doi:10.1145/501983.502011.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *CCS*, pages 98–107. ACM, 2002. doi:10.1145/586110.586125.

[RS06] 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. doi:10.1007/11761679_23.

[SPS$^+$22] Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Cassiers, and Corentin Verhamme. Triplex: an Efficient and One-Pass Leakage-Resistant Mode of Operation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):135–162, 2022. doi:10.46586/tches.v2022.i4.135-162.

[SPS24] Yaobin Shen, Thomas Peters, and François-Xavier Standaert. Multiplex: TBC-Based Authenticated Encryption with Sponge-Like Rate. *IACR Transactions on Symmetric Cryptology*, 2024(2):1–34, Jun. 2024. URL: https://tosc.iacr.org/index.php/ToSC/article/view/11618, doi:10.46586/tosc.v2024.i2.1-34.

[TMC$^+$23] 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. URL: https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8454.pdf.

[VMKS12] 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. doi:10.1007/978-3-642-34961-4_44.