Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction

Chun Guo\textsuperscript{1,2,3}, Olivier Pereira\textsuperscript{3},
Thomas Peters\textsuperscript{3} and François-Xavier Standaert\textsuperscript{3}

\textsuperscript{1} Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China, chun.guo@sdu.edu.cn
\textsuperscript{2} School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
\textsuperscript{3} Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM), UCLouvain, B-1348 Louvain-la-Neuve, Belgium
olivier.pereira@uclouvain.be, thomas.peters@uclouvain.be, fstandae@uclouvain.be

Abstract. The ongoing NIST lightweight cryptography standardization process highlights the importance of resistance to side-channel attacks, which has renewed the interest for Authenticated Encryption schemes (AEs) with light(er)-weight side-channel secure implementations. To address this challenge, our first contribution is to investigate the leakage-resistance of a generic duplex-based stream cipher. When the capacity of the duplex is of \(c\) bits, we prove the classical bound, i.e., \(\approx 2^{c/2}\), under an assumption of non-invertible leakage. Based on this, we propose a new 1-pass AE mode \texttt{TETSponge}, which carefully combines a tweakable block cipher that must have strong protections against side-channel attacks and is scarcely used, and a duplex-style permutation that only needs weak side-channel protections and is used to frugally process the message and associated data. It offers: (i) provable integrity (resp. confidentiality) guarantees in the presence of leakage during both encryption and decryption (resp. encryption only), (ii) some level of nonce misuse robustness. We conclude that \texttt{TETSponge} is an appealing option for the implementation of low-energy \(c\) in settings where side-channel attacks are a concern. We also provides the first rigorous methodology for the leakage-resistance of sponge/duplex-based AEs based on a minimal non-invertibility assumption on leakages, which leads to various insights on designs and implementations.

Keywords: Authenticated Encryption · Duplex Construction · Leakage-Resistance · Leveled Implementations · Multi-User/Beyond Birthday Security

1 Introduction

Problem statement. In 2013, the NIST initiated a lightweight cryptography project to understand the need for dedicated Authenticated Encryption with Associated Data (AEAD), which has led to the launching of a standardization process in 2019 [oST18]. In this context, resistance to side-channel attacks is identified as one of the desirable features that is missing from existing solutions. From an application viewpoint, it is clearly motivated by the observation that lightweight devices may be deployed in environments where they can be under physical control of an adversary, yet be responsible for critical tasks (e.g., automotive, drone-related). Maybe more worryingly, a lack of embedded security can also be the root of serious distributed attacks starting from seemingly non-critical connected objects, such as home lamps [RSWO17], for instance.
From a cryptographic viewpoint, NIST’s lightweight physical security goal challenges the current state of side-channel countermeasures, which typically imply significant overheads. For example, the cycle counts of the (optimized) masked software implementations of block ciphers by Goudarzi and Rivain presented at Eurocrypt 2017 [GR17] blows up by factors ranging from tens to hundreds for numbers of masking shares starting at 2, compared to a non-protected implementation. Significant (quadratic) overheads can also be observed in hardware as the number of shares increases, especially if the random generation of the shares is taken into account [GMK17].

This state of affairs raises the question of the design of AE modes allowing both efficient and light(er) weight implementations (e.g., supporting constant memory requirements in streaming applications) and embedding side-channel resistance features, so that the secure implementation of the mode can circumvent, at least in part, the costs associated to the protection of its underlying primitives. Besides, and if aiming at standardization, it is also desirable that such modes offer as many standard security features as possible. In this respect, we will be particularly interested in the following properties.

- Multi-user security, which is important in a context of large-scale deployment of lightweight device in a context of mass-surveillance and distributed adversaries [BT16].
- Security beyond the classical birthday bound, which increases key lifetime [GL17], a property that is particularly relevant in the case of lightweight devices for which key evolution protocols may cause undesirable costs or not be available at all.
- Nonce misuse resistance, or at least resilience, in order to mitigate errors in cipher usage [RS06, ADL17].

State-of-the-art. Taken separately, the design of lightweight symmetric primitives & modes and the design of leakage-resilient (or resistant) primitives & modes have been topics of quite intense research over the last years. For lightweight designs, we refer to the recent survey of Biryukov and Perrin [BP17], and to the CAESAR competition [cae19]. For leakage-resilient primitives, we refer to the line of works initiated by Dziembowski and Pietrzak’s leakage-resilient stream cipher [DP08], which has then been the seed for the design of PRGs, PRFs and PRPs [YSPY10, FPS12, DP10], with contrasted practical impact [BGS15]. For leakage-resistant authentication, encryption and AE modes, we refer to the CCS 2015 work of Pereira et al. [PSV15] and follow-ups [BKP +18, BPPS17, BMOS17, DM19]. We note that when it comes to encryption in the presence of leakage, we use the term leakage-resistant for security definitions that allow all the computations (including the computation of the “challenge ciphertext”) to leak, and the term leakage-resilient for security definitions that exclude it (following the terminology in [GPPS19, Sta19]).

Yet, significant gaps remain, particularly if we aim for a lighter single pass design. Concretely, let us focus on two recent proposals that explicitly seek to address side-channel attack resistance at the mode of operation level: one based on sponges, and the other based on tweakable block ciphers (TBCs). On the sponge side, ISAP was proposed at FSE 2017 by Dobraunig et al. as a potential solution for side-channel secure AE [DEM +17]. ISAP highlights the good properties of sponge-based constructions for side-channel security — an observation that was also made by the Keccak team in the design of Keyak [BDH +19]. In short, these designs are built on the assumption that some leakage can be tolerated as long as sponges have their capacity increased in proportion to the leakage.

In parallel to the work presented here, Dobraunig and Mennink [DM19] investigated the leakage-resilience of the keyed duplex construction in a systematic way, assuming that leakage leaves enough min-entropy in the state of the sponge. They investigate the security of the keyed duplex construction based on this assumption, and apply their methodology to ISAP and other schemes, focusing on confidentiality in the setting of nonce respecting and encryption leaking. Anticipating on the discussions below, our analysis of the keyed sponge construction is based on a strictly weaker assumption of hard-to-invert leakage
Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge

(see [FH15]). This leads to a slightly more complicated security analysis, and it is certainly interesting to be able to compare the two approaches. Besides, our TETSponge mode of operation aims at a strictly stronger form of confidentiality against moderate nonce reuse and encryption leakages, on top of authenticity against full nonce reuse and encryption & decryption leakages.

Turning back to AE modes, but moving to the TBC setting, TEDT [BGP\textsuperscript{+}19] was designed to encourage the so-called leveled implementations, where (expensive) protections against side-channel attacks are used in a minimal way, while the bulk of the computation can be executed by cheap and weakly protected circuits. The authors of TEDT argue that the leveled approach can bring significant performance gains compared to implementations where side-channel protections are uniformly used during encryption. Precisely, leveled implementations reduce the energy overheads of side-channel countermeasures, both in software implementations (in which case the gains are also visible in reduced cycle counts) and hardware ones, at the cost of a slightly increased code size or footprint. Regarding security, the leveled implementations of TEDT could ensure strong guarantees against leakages—corresponding to the top of the hierarchy of confidentiality and integrity definitions established in [GPPS19]. Namely, TEDT ensures Ciphertext Integrity with nonce Misuse resistance and Leakage in encryption and decryption (defined as CIML2 in [BPPS17]; also see Definition 2) in a liberal model where only the long-term key is “safe”. TETD also ensures security against Chosen Ciphertext Adversaries with misuse-resilience (meaning security for messages encrypted with fresh, non-repeating nonces,\textsuperscript{1} even if there is nonce-reuse otherwise) and leakage in encryption and decryption (defined as CCAmL2 in [BPPS17]; also see Definition 2), in two different leakage models. Yet, TETD requires two passes, hence requiring a memory as large as the plaintexts that are encrypted, which makes it hardly suitable for lightweight applications. Besides, it has rate 1/4, which is expected to be more expensive than the sponge-based approach of ISAP.

**Contribution.** Based on this state-of-the-art, we can rephrase our problem as:

*Can we design a single-pass leakage-resistant AE mode, and how can we argue about the form of leakage-resistance that it provides?*

The goal of efficient AE in 1 pass pinpoints the duplex construction [BDPV11] as the natural starting point (leakage-resistant AE modes for block ciphers typically refresh their keying material for every message block, inducing a significant overhead). To this end, our first contribution is a rigorous leakage security analysis of a general duplex-based stream encryption w.r.t. a leaky version of the classical eavesdropper security model. The analysis is made in the ideal permutation model since there is no other choice at present (see Section 3.4), and we naturally focus on oracle-free leakage functions—as in [YSFY10] and in the concurrent work [DM19]. As a compensation for this idealized analysis, all results are obtained under the weakest and easiest to validate leakage assumption, namely non-invertibility [FH15]. Building upon these, we prove security bounds that are expressive and easy-to-understand, translating to the classical \( 2^c/2 \) bound.

We then study how to extend the leakage-resistant duplex stream cipher into a 1-pass AEAD mode and what can be achieved. We propose a new AEAD mode TETsponge, standing for Tweakable (due to the use of TBC), Encrypt & Tag (the natural feature of duplex) Sponge, as our result. It *enhances the duplex stream cipher with two calls to an \( n \)-bit TBC used as a key derivation function (KDF) and a tag generation function (TGF)*. We also borrow the use of \( n_p \)-bit public key material from [BGP\textsuperscript{+}19] to boost multi-user

\textsuperscript{1}“Misuse-resilience” was due to [ADL17]. As discussed in [GPPS19], misuse-resistance in the sense of [RS06] is believed impossible in many leakage settings: briefly, when the inputs including nonces are arbitrarily controlled, the adversary could (almost) control the leakage traces, and this enables identifying small changes in the encrypted messages.
security. We show that these modifications cinch nice leakage AE security in the ideal TBC and permutation model;\(^2\) leveled implementations, where only the two TBC-calls are protected from side-channels, achieve: (i) multi-user CCAmL1 confidentiality up to \(\approx 2^{n/2}\) queries and \(\approx 2^{n^*}\) users, which corresponds to CCA security with misuse-resilience, in the presence of encryption leakages; (ii) multi-user CIML2 authenticity up to \(2^n/n^2\) queries and \(\approx 2^{n^*}\) users.

For practical usability, we carefully designed TETsponge to achieve high black-box CCA security as well, i.e., up to \(2^n/n^2\) queries and \(\approx 2^{n^*}\) users. To keep this submission concise and easy to follow, the black-box results are only presented in a separate supporting document as our main focus here is on leakage resistance.

Our contribution on the duplex stream cipher as well as the methodology for the leakage security analysis of duplex/sponge-based AEES are general and can be easily adapted to other duplex/sponge AEs.

Roadmap. In Section 2 we serve notations and the leakage security definitions used in this paper. Then Section 3 presents our first contribution, i.e., the analysis of the duplex-based stream cipher as well as the leakage assumptions in use. Building on this, Section 4 defines TETsponge.

The security analyses of TETsponge are complicated and divided into Sections 6 (integrity multi-user CIML2) and 7 (multi-user CCAmL1). But before them, we will first serve a useful intermediate result in Section 5.

## 2 Preliminaries

### 2.1 Notations and primitives

Given a bit-string \(x \in \{0,1\}^*\), \(|x|\) denotes its length. For any value \(x\), we denote by \(\lfloor x_a\rfloor / \lceil x_a\rceil\) the least/most significant \(a\) bits of \(x\). \([\text{num}]_a\) is the binary encoding of the integer \(\text{num}\) using a representation of \(a\) bits.

We denote by a \((q_1, \ldots, q_m, t)\)-bounded adversary a probabilistic algorithm that has access to \(\omega\) oracles, \(O_1, \ldots, O_\omega\), can make at most \(q_i\) queries to its \(i\)-th oracle \(O_i\), and can perform computation bounded by running time \(t\). Security notions define the oracles \(O_1, \ldots, O_\omega\) available to the adversary in a security experiment. In a proof in the ideal model, the adversary is also granted access to ideal objects (see later) that we do not always make explicit in the notation.

A leaking implementation of an algorithm \(\text{Algo}\) is denoted \(L_{\text{Algo}}\). It runs both \(\text{Algo}\) and a leakage function \(L_{\text{Algo}}\) which captures the additional information given by the implementation of \(\text{Algo}\) during its execution. \(L_{\text{Algo}}\) simply returns the outputs of both \(\text{Algo}\) and \(L_{\text{Algo}}\) which all take the same input.

A random keyless permutation \(\pi\), as used in sponge analyses, refers to a permutation of \(\{0,1\}^b\) drawn uniformly at random among the set of all permutations of \(\{0,1\}^b\), where \(b\) is the security parameter. A Tweakeble Block Cipher (TBC) with key space \(\{0,1\}^k\), tweak space \(\{0,1\}^t\), and domain \(\{0,1\}^n\), also denoted \((k,t,n)\)-TBC, is a mapping \(\tilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n\) such that for any key \(K \in \{0,1\}^k\) and any tweak \(T \in \{0,1\}^t\), \(X \mapsto \tilde{E}(K,T,X)\) is a permutation of \(\{0,1\}^n\). In this paper we only focus on \((n,n,n)\)-TBC. An ideal TBC \(I\tilde{C} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n\), with the same spirit as ideal (block) ciphers, is a TBC sampled uniformly from all \((n,n,n)\)-TBCs. In this case,

\(^2\)Multi-user security can also be proved under the standard strong tweakable pseudorandom permutation assumption, though limited to \(n/2\) bit. This limitation appears to be common: the ideal (tweakable) blockcipher model is used to obtain non-trivial beyond \(n/2\) bit multi-user security in several other works [BT16, BHT18].
Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge

\[ \tilde{\mathcal{C}}^T_K \] is a random independent permutation of \( \{ 0, 1 \}^n \) for each \( (K, T) \in \{ 0, 1 \}^n \times \{ 0, 1 \}^n \) even if the key \( K \) is public.

**Definition 1** (Nonce-based AEAD). A nonce-based authenticated encryption scheme with associated data is a tuple \( \text{AEAD} = (\text{Enc}, \text{Dec}) \) such that:

- \( \text{Enc} : K \times N \times AD \times M \to C \) maps a key \( k \in K \), a nonce \( N \in N \), some blocks of associated data \( A \in AD \), and a message \( M \in M \) to a ciphertext \( C \in C \).
- \( \text{Dec} : K \times N \times AD \times C \to M \cup \{ \perp \} \) maps \( k \in K \), \( N \in N \), \( A \in AD \), and \( C \in C \) to a message \( M \in M \) that is the decryption of that ciphertext, or to a special symbol \( \perp \) if integrity checking fails.

The message size \( \ell_m \) uniquely determines the ciphertext size \( \ell_c = \ell_m + \text{oh} \), where the constant \( \text{oh} \) is the stretch. Given a key \( k \leftarrow K \), \( \text{Enc}_k(N, A, M) := \text{Enc}(k, N, A, M) \) and \( \text{Dec}_k(N, A, C) := \text{Dec}(k, N, A, C) \) are deterministic functions whose implementations may be probabilistic.

### 2.2 Multi-user AE security with leakage

In general, leakage security definitions are stated w.r.t. implementations of a scheme (e.g., an AEAD), and both an encryption leakage function \( L_{\text{Enc}} \) and a decryption leakage function \( L_{\text{Dec}} \) are associated to the implementation(s). This models the real-world leaky implementations of the mathematical objects. Note that in theory, our leakage model is non-adaptive, as the leakages are a parameter of the to-be-studied implementations determined before the experiment starts rather than chosen by the adversary during the experiment. This restriction was motivated from the side-channel practice & the necessity for practical modes: see [YSPY10, FPS12] for some discussion. As more recently discussed in [BDF+17], adaptive leakage models also have limited relevance in the context of power and EM side-channels where the adversary has access to the (noisy) leakage of all the intermediate computations. They may be more relevant in the abstract probing model where one probe has to be excluded from the adversary’s view, hence making the adaptive selection of a probe an important feature of the analysis.

Pioneered by Rogaway and Shrimpton [RS06], nowadays black-box AE analyses typically follow all-in-one definitions that integrate both confidentiality and integrity. However, in front of an adversary with access to leakage, the adoption of separate definitions for integrity and confidentiality potentially offers more insight on which implementation-level properties are necessary/sufficient for which goal [Sta19]. This difference follows from the important general feature of physically observable cryptography that unpredictability is much easier to ensure than indistinguishability [MR04], which has a strong impact on the assumptions that may be needed to prove both notions. Also, different levels of robustness against nonce reuse may be achieved w.r.t. these notions: it was shown that when a nonce is arbitrarily reused (in the same flavor as the misuse-resistance notion [RS06]), integrity in the presence of leakage is achievable [BKP+18, BPPS17], yet confidentiality in the presence of leakage may not [BKP+18, GPPS19] (see also footnote 1). This difference is also reflected in the separate definitions.

#### 2.2.1 Integrity

In detail, regarding integrity, we rely on the multi-user Ciphertext Integrity with Misuse-resistance and Leakage (muCIML2) defined in [BGP+19], which was built upon the single-user version CIML2 introduced in [BKP+18, BPPS17]. The suffix 2 means two leakage sources, i.e., both encryption and decryption. In some sense, the definition is obtained by enhancing the traditional (multi-user) INT-CTX security with leakages.
Definition 2 (muCIML2 advantage). Given the implementation of a nonce-based authenticated encryption $\text{AEAD} = (\text{Enc}, \text{Dec})$ with leakage function pair $L = (L_{\text{Enc}}, L_{\text{Dec}})$, the multi-user ciphertext integrity advantage with misuse-resilience and leakage of an adversary $A$ against $\text{AEAD}$ with $u$ users is

$$\text{Adv}_{A,\text{AEAD},L,u}^{\text{muCIML2}} := \Pr[A^{L_{\text{Enc}}, L_{\text{Dec}}, \pi, \pi^{-1}, \vec{I}; \vec{I}^{-1}} \text{forges}],$$

where the probability is taken over the $u$ user keys $K = (K_1, \ldots, K_u)$, $K_i \xleftarrow{\$} K$, over $A$'s random tape and the ideal oracles $\pi$ and $\vec{I}$ and where:

- $L_{\text{Enc}}(i, N, A, M)$: if $1 \leq i \leq u$, outputs the cipher $\text{Enc}_{K_i}(N, A, M)$ and the leakage trace $L_{\text{Enc}}(K_i, N, A, M)$;
- $L_{\text{Dec}}(i, N, A, C)$: if $1 \leq i \leq u$, outputs $(\text{Dec}_{K_i}(N, A, C), L_{\text{Dec}}(K_i, N, A, C))$;
- The event “$A$ forges” means that any of $A$’s query to $L_{\text{Dec}}(i, N, A, C)$ returns $M \neq \bot$, while $C$ was not resulted from an earlier encryption query to $L_{\text{Enc}}(i, N, A, M)$.

### 2.2.2 Confidentiality

Regarding confidentiality, we try to achieve weaker nonce robustness, i.e., multi-user Chosen-Ciphertext Attack security with misuse-resilience and Leakage. Informally, the notions allow the adversary $A$ to query leaking encryption and (leaking) decryption oracles arbitrarily, and capture the confidentiality of several messages encrypted using fresh, non-repeating nonces. The latter goal is formalized via the “old-school” left-or-right paradigm, i.e., $A$ shall not tell apart encrypting $M_0$ from encrypting $M_1$. This choice follows from the conceptual difficulty to define the leakage of the idealized (random) objects that are used in real-or-random definitions [Sta19]. As mentioned, the misuse-resilience we guarantee captures confidentiality with fresh nonces in the challenges and was first introduced by Ashur et al. [ADL17].

Following [GPPS19, BGP+19], the notion capturing 1 leakage source (i.e., encryption only) is abbreviated as $\text{muCCAmL1}$, while the notion capturing the presence of 2 leakages sources (i.e., including decryption) is $\text{muCCAmL2}$. We only recall the former definition.

Definition 3 (muCCAmL1 advantages). Given the implementation of a nonce-based authenticated encryption $\text{AEAD} = (\text{Enc}, \text{Dec})$ with leakage function pair $L = (L_{\text{Enc}}, L_{\text{Dec}})$, the multi-user chosen-ciphertext advantage with misuse-resilience and leakage of an adversary $A$ against $\text{AEAD}$ with $u$ users is

$$\text{Adv}_{A,\text{AEAD},L,u}^{\text{muCCAmL1}} := \left| \Pr[\text{PrivK}^{\text{muCCAmL1,0}}_{A,\text{AEAD},L,u} \Rightarrow 1] - \Pr[\text{PrivK}^{\text{muCCAmL1,1}}_{A,\text{AEAD},L,u} \Rightarrow 1] \right|,$$

where the security game $\text{PrivK}^{\text{muCCAmL1,d}}_{A,\text{AEAD},L,u}$ is defined in Figure 1.

We will also use a notion of eavesdropper security with leakage. Since it is a bit specific for “one-time” stream ciphers, we defer it to Section 3.

### 2.3 Oracle-free probabilistic leakage functions

Most analyses of sponge-based constructions rely on the ideal (permutation) model (see Section 3.4 for an alternative). We follow that practice, which has the advantage of offering an easy compatibility with quite minimal leakage assumptions: we will assume that leakages resulting from each call to the permutation $\pi$ are non-invertible. This assumption was previously used by Yu et al. [YSFY10] on a random oracle-based PRG. This approach also comes with the important benefit that it can be easily measured/challenged by cryptanalytic practice (as will be detailed in the next section), and therefore might lead
Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge

As mentioned, one of our main contributions is the leakage security of a general duplex-based AE, Ascon [DEMS19], and TETsponge that will be introduced. Formally, the cipher \( \text{DuStr}[\pi] \) is defined in Figure 2. For simplicity we assume the size of the inputs |A| and |M| are always multiples of \( r \) (otherwise it would be too complicated to follow), and we use an offset \( \delta_1 \) to model the commonly used domain separation bits in duplex AEs, which depends on the concrete schemes and is typically of only 1 or 2 bits. For preciseness we also make the leakages (which follows the convention from Section 2.3) of each step explicit.

For the leakage security of \( \text{DuStr}[\pi] \), we follow [PSV15] and leverage the security model (left-or-right) Eavesdropper security with Leakage (EavL). It states that two distinct messages encrypted by \( \text{DuStr}[\pi] \) are indistinguishable. Formally, recall that we have \( \text{LDuStr}_{\pi}( IV, A, M ) := ( \text{DuStr}_{\pi}( IV, A, M ) ) \) according to our conventions, where the leakage function \( \text{LDuStr}_{\pi} \) includes all the leakages listed in Figure 2.

---

**PrivK_{\muCCAmL1,d}^{\text{muCCAmL1,d}}** is the output of the following experiment:

**Initialization:** generates \( u \) secret keys \( K_1, \ldots, K_u \) \( \in \mathbb{K} \) and sets \( E_{ch}, E_1, \ldots, E_u \leftarrow \emptyset \).

**Leaking encryption queries:** \( A \) gets adaptive access to \( \text{LEnc}( \cdot, \cdot, \cdot, \cdot ) \).

\[
\text{LEnc}( i, N, A, M ) \text{ outputs } \bot \text{ if } (i, N, \cdot, \cdot) \in E_{ch}, \text{ else computes } C \leftarrow \text{Enc}_{K_i}( N, A, M ) \text{ and leak}_{\text{enc}} \leftarrow \text{LEnc}( K_i, N, A, M ), \text{ updates } E_i \leftarrow E_i \cup \{ N \} \text{ and returns } ( C, \text{leak}_{\text{enc}} ) .
\]

**Decryption queries:** \( A \) gets adaptive access to \( \text{Dec}( \cdot, \cdot, \cdot, \cdot ) \).

\[
\text{Dec}( i, N, A, C ) \text{ outputs } \bot \text{ if } (i, N, A, C) \in E_{ch}, \text{ else returns } \text{Dec}_{K_i}( N, A, C );
\]

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

If \( |M^0| \neq |M^1| \) or \( N_{ch} \in E_i \) or \( (i, N_{ch}, \cdot, \cdot) \in E_{ch} \), returns \( \bot \); Else computes \( C^d \leftarrow \text{Enc}_{K_i}( N_{ch}, A_{ch}, M^d ) \) and \( \text{leak}^d_{\text{enc}} \leftarrow \text{LEnc}( K_i, N_{ch}, A_{ch}, M^d ) \), updates \( E_{ch} \leftarrow E_{ch} \cup \{ (i, N_{ch}, A_{ch}, C^d) \} \) and finally returns \( ( C^d, \text{leak}^d_{\text{enc}} ) \).

**Finalization:** \( A \) outputs \( d' \) which is defined as the output of the game.

Figure 1: The \( \text{PrivK}_{\muCCAmL1,d}^{\text{muCCAmL1,d}} \) game.

to a better understanding of how to implement and design modes from a real-world perspective.

Formally, for \( F \in \{ \pi, \oplus \} \), i.e., the components of a duplex, each computation of \( F \) comes with leakages \( L_F \), which is an efficient probabilistic function of the values involved in the computation. For the permutation \( \pi \), we further split the leakage into an input and an output part, i.e., we write \( ( L^\text{in}_F(S^\text{in}), L^\text{out}_F(S^\text{out}) ) \) for the leakage due to evaluating \( \pi(S^\text{in}) \rightarrow S^\text{out} \) or \( \pi^{-1}(S^\text{out}) \rightarrow S^\text{in} \). This distinction between \( L^\text{in}_F \) and \( L^\text{out}_F \) allows to independently quantify the secrecy of the input and the output which better reflects the designers implementation goals for each functions/calls.

We require that for any \( F \), the leakage function \( L_F \) must have no access to the ideal oracle \( \pi \). This restriction effectively excludes the artificial “future computation attacks” from the model: it guarantees that \( L_F \) only leaks information about the computation that is happening in the device rather than the computation that may happen in “future” calls of \( \pi \). Oracle-freeness restriction was first used by Yu et al. [YSPY10], and recently by the concurrent work of DM [DM19]. For the sake of space, we refer to [DP08] and [YSPY10] for detailed discussion about “future computation attacks”.

### 2.4 A general duplex-based stream cipher and its security models

As mentioned, one of our main contributions is the leakage security of a general duplex-based stream cipher, denoted \( \text{DuStr}[\pi] \), that constitutes the basis of many duplex-based AEs (e.g., Ascon [DEMS19], and TETsponge that will be introduced). Formally, the cipher \( \text{DuStr}[\pi] \) is defined in Figure 2. For simplicity we assume the size of the inputs |A| and |M| are always multiples of \( r \) (otherwise it would be too complicated to follow), and we use an offset \( \delta_1 \) to model the commonly used domain separation bits in duplex AEs, which depends on the concrete schemes and is typically of only 1 or 2 bits. For preciseness we also make the leakages (which follows the convention from Section 2.3) of each step explicit.
The duplex stream cipher $\text{DuStr}_B[\pi](IV, A, M)$, $|B| = \kappa$:

1. Computes $S_0' \leftarrow IV \parallel B$, $S_1 \leftarrow \pi(S_0')$. **The leakages** of this step are $L^{\text{in}}_0(S_0')$ and $L^{\text{out}}_1(S_1)$;
2. For $i = 1, \ldots, \nu, \nu = |A|/r$, computes $S'_i \leftarrow (A[i] || 0^r) \oplus S_i$ and $S_{i+1} \leftarrow \pi(S'_i)$. **The leakages** are $L^{\text{in}}_i(S'_i), L^{\text{out}}_i(S_{i+1}), L_B(\text{msb}_r(S_i), A[i])$;
3. $S_{i+1} \leftarrow S_{i+1} \oplus (0^r || \delta_1)$ for a fixed offset $\delta_1$. **The leakages** are $L_B(\text{lsb}_r(S_{i+1}), \delta_1)$;
4. For $i = 1, \ldots, \ell, \ell = |M|/r$, computes $j \leftarrow i + r$, $C[j] \leftarrow \text{msb}_r(S_j) \oplus M[i]$, $S'_j \leftarrow C[j] \parallel \text{lsb}_r(S_j)$, $S_{j+1} \leftarrow \pi(S'_j)$. **Leakages** are $L_B(\text{msb}_r(S_j), M[i])$ and $L^{\text{in}}_i(S'_j), L^{\text{out}}_i(S_{j+1})$;

**Figure 2:** The duplex-based stream cipher $\text{DuStr}$ and the involved leakages.

Then, the **leakage eavesdropper advantage** of an adversary $\mathcal{A}^\pi$ against $\text{LDuStr}$ is

$$\text{Adv}^\text{Eav}_{\text{LDuStr}}(\mathcal{A}^\pi) := \left| \Pr[\mathcal{A}^\pi(\text{LDuStr}_B(IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\text{LDuStr}_B(IV, A, M^1)) \Rightarrow 1] \right|.$$  \hspace{1cm} (1)

This notion can be seen as the leaky version of the black-box eavesdropper security, or an extremely weakened muCCAmL1 setting that only allows the adversary to make a single query to the challenge leaking encryption oracle and have its ciphertext and encryption leakages (as such, it can also be viewed as “single-query leakage CCA”). For ease of use, we further define

$$\text{Adv}^\text{Eav}_{\text{LDuStr}}(q_\pi, t, \ell) := \max \left\{ \text{Adv}^\text{Eav}_{\text{LDuStr}}(\mathcal{A}) \right\},$$ \hspace{1cm} (2)

where the maximal is taken over all adversaries $\mathcal{A}$ that makes $q_\pi$ queries to $\pi$, runs in time $t$, and observes the ciphertext and leakages from encrypting a message with $\ell$ blocks.

### 2.5 The H-coefficient technique

Our proofs crucially rely on Patarin’s H-coefficient technique [Pat08]. We thereby provide a quick overview, following the presentation of [CS14]. Fix a distinguisher $D$ that makes a bounded number of $q$ queries to its oracles. The technique addresses the settings where $D$’s aim is to distinguish between two worlds: a “real world” and an “ideal world”. Assume wlog that $D$ is deterministic. The execution of $D$ defines a transcript that includes the sequence of queries and answers received from its oracles; $D$’s output is a deterministic function of its transcript. Thus, if $T_{re}$ and $T_{id}$ denote the probability distributions on transcripts induced by the real and ideal worlds, respectively, then $D$’s distinguishing advantage is upper bounded by the statistical distance between $T_{re}$ and $T_{id}$, which can be further bounded by the H-coefficient main lemma [CS14]. In brief, we have

**Lemma 1.** Fix a distinguisher $D$. Let $\mathcal{T} = \mathcal{T}_{\text{good}} \cup \mathcal{T}_{\text{bad}}$ be a partition of the set of attainable transcripts $\mathcal{T}$. Assume that there exists $\varepsilon_1$ such that for any $\tau \in \mathcal{T}_{\text{good}}$, one has

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

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

The definitions of “transcripts” crucially depend on the concrete problem, and could be found in the proofs latter.

### 3 Leakage security of the duplex stream cipher $\text{DuStr}[\pi]$

In this section we investigate the leakage confidentiality of the duplex-based stream cipher $\text{DuStr}[\pi]$. Leakage confidentiality is in general hard to achieve (and analyze) and
assumptions that “bound” the information in leakages are clearly necessary, as otherwise cleartext messages may leak in full. In this respect, we first formalize the two assumptions that we use to bound the leakages’ informativeness, i.e., the non-invertibility assumption in Section 3.1, and the bounded XOR leakage assumption in Section 3.2. Based on these, in Section 3.3 we finally prove its EavL security. We serve some discussion in Section 3.4.

### 3.1 Bounding the leakages: non-invertibility restriction

**Motivation.** As mentioned before, we assume that (some of) the leakage functions are essentially non-invertible. In detail, note that the following sequence of actions appear in virtually all duplex-based AEs: (1) squeezing (the most significant bits, wlog) from a secret b-bit state $S_i$: $Y_i \leftarrow \text{msb}_b(S_i)$; (2) modifying the state with a b-bit offset $\Delta$: $S_i' \leftarrow S_i \oplus \Delta$; (3) deriving a new state: $S_{i+1} \leftarrow \pi(S_i')$. After the 3rd step, the subsequent computations are irrelevant to the state $S_i$. We therefore view this as a basic unit and bound its leakages on $S_i$.

Let us reconsider the above unit. Note that for the call $\pi(S_i')$ in step (3), in many cases the piece $\text{msb}_b(S_i')$ is a non-secret ciphertext block, e.g., see Fig. 6 (on the other hand, the piece $\text{lsb}_b(\Delta)$ captures the possible XOR of domain separation bits). Therefore, the least significant bits $\text{lsb}_b(S_i')$ are the critical secret, and the subsequent actions are compromised as long as $\text{lsb}_b(S_i')$ is recovered. These $c$ bits are involved in three actions. The 1st is of course the call $\pi(S_i')$. The others are: (a) the XOR of $\Delta$, and (b) the “previous” $\pi$-call $\pi(*) \to S_i$ that “produces” $S_i$—following the convention in duplex papers, we didn’t present this action in the above unit, but it’s clearly relevant. We shouldn’t be restricted to $\text{lsb}_b(S_i')$, as we may also consider initializing the duplex with a key of $\kappa \neq c$ bits. Therefore, in our assumption we consider $\omega$ bits $\text{lsb}_b(S_i')$ for generality, and assume that the side-channel adversary cannot predict the value of $\text{lsb}_b(S_i')$ within a limited number of guesses, even if all the other involved $(b - \omega)$-bit values are chosen by him (this simplification also emphasizes the crucial role of $\text{lsb}_b(S_i')$).

**Definition.** Formally, we define

$$\text{Adv}^{\text{inv}[\omega]}(A) := \Pr[\text{ch} \leftarrow \{0,1\}^\omega, \mathcal{G} \leftarrow A^\text{x}(\text{leak}) : \text{ch} \in \mathcal{G}],$$

where $\mathcal{G}$ is a finite set of guesses, and $A$’s input leak is a list of leakages depending on three values $y^{\text{in}}, y^{\text{pre}} \in \{0,1\}^{b-\omega}$, and $\delta \in \{0,1\}^\omega$ chosen by $A$, i.e.,

$$\text{leak} = [L^\text{out}_x(y^{\text{pre}}||\text{ch}), L^\oplus_\delta(\delta||\text{ch}), L^\text{in}_\pi(y^{\text{in}}||δ \oplus \text{ch})].$$

**Further Insights.** To clarify, the random state $s_{\text{ch}}$ is the secret that is to be challenged by $A$. $A$ is required to choose $y^{\text{pre}}, y^{\text{in}},$ and $\delta$ (playing the role of the aforementioned $\text{lsb}_b(\Delta)$) to “fill in the gap” and gets the leakages, as if $y^{\text{pre}}||s_{\text{ch}}$ is the aforementioned current state $S_i$ of the “unit”, which is modified to $y^{\text{in}}||δ \oplus s_{\text{ch}}$ and results in a call to $\pi(y^{\text{in}}||\delta \oplus s_{\text{ch}})$). See Fig. 3 (Left) for illustration. As mentioned, $s_{\text{ch}}$ is involved in three actions, which exactly correspond to the three leakages.

In such an invertibility game, the power of $A$ is quantified along three dimensions, i.e., the number $q_\pi$ of its queries to $\pi$, the running time $t$, and the number $N_G$ of allowed guesses (i.e., $|\mathcal{G}| \leq N_G$; clearly the larger $N_G$, the higher $\text{Adv}^{\text{inv}[\omega]}(A)$). To simplify, we further define

$$\text{Adv}^{\text{inv}[\omega]}(q_\pi, t, N_G) := \max_{q_\pi, t, N_G,A} \left\{ \text{Adv}^{\text{inv}[\omega]}(A) \right\},$$

When $\omega = c$, the assumption captures the secrecy of the “capacity part”, which is in line with the intuition proposed for ISAP [DEM+17].
Figure 3: (Left) Illustrating the Inv[\omega] assumption. The value in red (i.e., \texttt{s}_{ch}) is the critical secret. The involved values are as defined with Eq. (3), while the values \texttt{S}^{pre} and \texttt{S}^{out} are only mentioned in the subsequent security tester. (Middle) The “basic” message manipulating operation. The value \texttt{y}_{ch} in red is the secret. (Right) A summary: which values should be somehow ensured secret (i.e., the \texttt{lb}_{\omega} bits of the input, and the entire output, as stressed by the red bold lines).

**Tester: measuring leakage in practice.** The concrete value of the advantage \texttt{Adv}^{\text{Inv}[\omega]} can be measured for a specific implementation on a specific device by running the best known side-channel key/secret recovery adversary \texttt{A} against the following tester. This along with our theorems later (e.g., Theorem 3) allows to determine how many plaintext blocks can be processed before key updating.

1. **Tester for non-invertibility**\texttt{Adv}^{\text{Inv}[\omega]}(q_{s}, t, N_{G})
2. Let the adversary \texttt{A} serve \(s - \omega\) bit values \(\texttt{y}^{pre}\) and \(\texttt{y}^{in}\) and \(\omega\)-bit offset \(\delta\)
3. Pick the secret: \(\texttt{s}_{ch} \leftarrow \{0,1\}^{s}\)
4. Compute \(\texttt{S}^{pre} \leftarrow \pi^{-1}(\texttt{y}^{pre}||\texttt{s}_{ch})\), \(\texttt{s}_{h} \leftarrow \texttt{s}_{ch} \oplus \delta\), and \(\texttt{S}^{out} \leftarrow \pi^{-1}(\texttt{y}^{in}||\texttt{s}_{h})\)
5. Serve \texttt{A} with the leakages resulting from step 4. Following our notations, this gives \texttt{A} the leakages \(\texttt{L}^{out}_{\omega}(\texttt{y}^{pre}||\texttt{s}_{ch}), \texttt{L}_{\oplus}(\texttt{s}_{ch}, \delta)\), and \(\texttt{L}^{in}_{\omega}(\texttt{y}^{in}||\texttt{s}_{h})\).
6. Let \texttt{A} output \(N_{G}\) guesses \(G = \{s_{1}, \ldots, s_{N_{G}}\}\), and \texttt{A} wins as long as \(\texttt{s}_{ch} \in G\)

### 3.2 (In)Distinguishability of the XOR leakages

**Motivation.** An implementation that aims at preserving confidentiality shall offer some indistinguishability property for its message manipulations. To minimize assumptions & ease measuring in practice, we follow the methodology of [PSV15, BGP+19]: we define such an assumption w.r.t. the information an adversary might extract from the “basic” message manipulation made in a keyed duplex (i.e., the aforementioned unit), and then reduce the confidentiality of the “bigger” encryption to this assumption. Note that this approach is more fine-grained than just assuming XORs leak-free.

**Definition.** Note that, when running the aforementioned unit, a part of the state—typically of \(r\) bit,—is extracted as a key stream block, see Fig. 3 (middle). Concerning the involved leakages, and with the goal of bounding LOR distinguishing advantages with leakages in mind, we define

\[
\text{Adv}^{\text{LORL}}(\texttt{A}) := \Pr[y_{ch} \leftarrow \{0,1\}^{c}, c \leftarrow y_{ch} \oplus m^{0} : \texttt{A}^{c}(c^{0}, \text{leak}_{0}) \Rightarrow 1] - \Pr[y_{ch} \leftarrow \{0,1\}^{c}, c^{1} \leftarrow y_{ch} \oplus m^{1} : \texttt{A}^{c^{1}}(c^{1}, \text{leak}_{1}) \Rightarrow 1],
\]

(6)

where \text{leak}_{d} again depends on a \(c\)-bit value \(s\) chosen by \texttt{A}:

\[
\text{leak}_{d} = [\texttt{L}^{out}_{\omega}(y_{ch}||s), \texttt{L}_{\oplus}(y_{ch}, m^{d})].
\]

(7)

**Further Insights.** Concretely, the sensitive data is the key stream block \(y_{ch}\). It is the output of a permutation-call, hence the presence of \(\texttt{L}^{out}_{\omega}(y_{ch}||s)\). Then, \(y_{ch}\) is used to mask
the message block, and thus $L_{\oplus}(y_{\chi}, m^d)$ comes. We also define

$$\text{Adv}^{\text{LORL}}(q_{\pi}, t) := \max_{\mathcal{A}} \left\{ \text{Adv}^{\text{LORL}}(\mathcal{A}) \right\}. \quad (8)$$

While not necessarily negligible, this advantage can also be measured by a tester:

1. **Tester for LORL** $\text{Adv}^{\text{LORL}}(q_{\pi}, t)$
2. Let the challenging adversary $\mathcal{A}$ serve $s$ and $(m^0, m^1)$
3. Pick the secret: $y_{\chi} \leftarrow \{0, 1\}^\nu$, $d \leftarrow \{0, 1\}$
4. Compute $c \leftarrow y_{\chi} \oplus m^d$
5. Generate the permutation leakages: repeating $S_{\text{pre}} \leftarrow \pi^{-1}(y_{\chi}\|s)$ for $p$ times
6. Serve $\mathcal{A}$ with $c$ and the leakage traces resulted from steps 4 and 5. This gives $\mathcal{A}$ the tuple $(c, [L_{\oplus}^{\text{opt}}(y_{\chi}\|s), L_{\oplus}(y_{\chi}, m^d)])$
7. Let $\mathcal{A}$ output the guess $d'$, $\mathcal{A}$ wins as long as $d' = d$.

As discussed in [HLWW16, PSV15, Sta19], if a single XOR of the message leaks a single bit, then no confidentiality would spring up. Thus, it is legitimate to focus on protecting this part of the implementations. Concretely, and while it may not be possible to guarantee that $\text{Adv}^{\text{LORL}}(q_{\pi}, t)$ is negligible, the advantages of this methodology are two-fold: in theory, it could help establish somewhat best possible mode-level leakage security for relevant applications; in practice, it allows us to faithfully reduce the confidentiality to very simplified one-time components, for a single computation from a fixed random string, which further makes it easier to study and to protect as isolated components. To ease understanding, a summary of the critical values and their leakages is given in Fig. 3 (right).

### 3.3 Leakage EavL security of DuStr[$\pi$]

With the above preparations, in this subsection we establish the leakage eavesdropper security of the stream cipher DuStr[$\pi$]. Formally, we have

**Theorem 1.** Regarding the EavL security of DuStr[$\pi$] defined in Eq. (2), we have

$$\text{Adv}^{\text{EavL}}_{\text{DuStr}}(q_{\pi}, t, \ell) \leq \frac{(t + 2)^2}{2\epsilon} + \ell \cdot \text{Adv}^{\text{LORL}}(q_{\pi}, t^*) + 2\text{Adv}^{\text{Inv}[c]}(q_{\pi}, t^*, 2q_{\pi})$$

$$+ 2(\ell + 1) \cdot \text{Adv}^{\text{Inv}[c]}(q_{\pi}, t^*, 2q_{\pi}), \quad (9)$$

where $t^* = O(t + \ell t_1)$, and $t_1$ is as defined in Lemma 2.

Theorem 1 only gives rise to a *one-time* encryption scheme since it only concerns the encryption of a single message. But as will be seen, it can be used to establish AE security in an almost modular manner. Regarding the bounds, the term $\ell \cdot \text{Adv}^{\text{LORL}}(q_{\pi}, t^*)$ reflects the reduction to the “minimal message manipulation” (as discussed earlier), and the factor $\ell$ reflects a (seemingly) unavoidable leakage security loss. The terms $2\text{Adv}^{\text{Inv}[c]}(q_{\pi}, t^*, 2q_{\pi}) + 2(\ell + 1) \cdot \text{Adv}^{\text{Inv}[c]}(q_{\pi}, t^*, 2q_{\pi})$ capture the hardness of side-channel secret recovery, and they are roughly of some birthday type

$$O\left(q_{\pi} + \frac{\ell + t}{\mu_c \cdot 2^c} \right) + O\left(\ell \cdot q_{\pi} + \frac{\ell + t}{\mu_c \cdot 2^c} \right),$$

for some parameters $\mu_c$ and $\mu_c$ that depend on the concrete conditions. It is nowadays a common assumption that with such a small data complexity (2 or 3 leakage traces) $\mu_c$ and $\mu_c$ can be made very small [DEM+17], so we “restore” the classical $2^{c/2}$ birthday security. The birthday bounds are essentially tight w.r.t. our assumptions: a collision between the (internal) secret $c$-bit state values allows the adversary to obtain more than 2 leakages about a single secret value, which is beyond our assumption (security with 2 leakages). So our assumption Eq. (4), though a bit conservative, tightly restores the classical $2^{c/2}$ birthday bound.
The ideal stream cipher \text{IdealS}(IV, A, M):

1. Samples \( B \leftarrow \{0, 1\}^r \);
2. Computes \( S_0 \leftarrow IV||B \) and samples \( S_i \leftarrow \{0, 1\}^b \). The \text{leakages} of this step are \( L^\text{in}_i(S_0) \) and \( L^\text{out}_i(S_1) \);
3. For \( i = 1, \ldots, \nu = |A|/r \), computes \( S'_i \leftarrow (A[i]||0^r) \oplus S_i \) and samples \( S_{i+1} \leftarrow \{0, 1\}^b \). The \text{leakages} are \( L^\text{in}_i(S'_i), L^\text{out}_i(S_{i+1}), L_\oplus(\text{msb}_i(S_i), A[i]) \);
4. \( S_{n+1} \leftarrow S_{n+1} \oplus (0^r || \delta_1) \). The \text{leakages} are \( L_\oplus(\text{lsb}_i(S_{n+1}), \delta_i) \);
5. For \( i = 1, \ldots, \ell = |M|/r \), computes \( j \leftarrow i + \nu \), \( C[i] \leftarrow \text{msb}_i(S_j) \oplus M[i] \), \( S'_j \leftarrow C[i] || \text{lsb}_i(S_j) \) and samples \( S_{j+1} \leftarrow \{0, 1\}^b \). The \text{leakages} are \( L_\oplus(\text{msb}_i(S_j), M[i]) \) and \( L^\text{out}_i(S_{j+1}) \);
6. Returns \( c = C[1] || \ldots || C[\ell] \).

**Figure 4:** The ideal stream cipher \text{IdealS} and the involved leakages.

**Proof.** Technically, it is hard to prove Theorem 1 “directly”, and thus we follow [PSV15] and introduce an idealized stream cipher \text{IdealS} in Figure 4 for “relay”. Briefly, \text{IdealS} is obtained via replacing all the internal actions \( S_{i+1} \leftarrow \pi(S'_i) \) in \text{DuStr}[\pi] by sampling \( S_{i+1} \leftarrow \{0, 1\}^b \).

By our conventions, we have \( \text{LidealS}(IV, A, M) := (\text{IdealS}(IV, A, M), L_{\text{idealS}}(IV, A, M)) \), where \( L_{\text{idealS}} \) includes all the leakages in Figure 4.

Regarding \text{IdealS}, we have the following two lemmas, the proofs of which are deferred to subsections 3.3.1 and 3.3.2 respectively, for cleaness.

**Lemma 2.** For every \((q_\pi, t)\)-bounded distinguisher \( D^\pi \) and every adversary-chosen triple \( \text{tpl} = (IV, A, M) \) such that \((A, M)\) has \( \ell \) blocks in total, it holds

\[
\begin{align*}
\Pr[D^\pi(LDuStr_B[\pi](\text{tpl})) \Rightarrow 1] - \Pr[D^\pi(\text{LidealS}(\text{tpl})) \Rightarrow 1] \\
&\leq (\ell + 2)^2 2^{c+1} + \text{Adv}^{\text{Inv}[\pi]}(q_\pi, t^*, 2q_\pi) + (\ell + 1) \cdot \text{Adv}^{\text{Inv}[\pi]}(q_\pi, t^*, 2q_\pi),
\end{align*}
\]

(10)

where \( t^* = O(t + \ell t_1) \), and \( t_1 \) is the total time needed for evaluating \( L^\text{in}_i, L^\text{out}_i, L_\oplus, \) and the xor of two \( r \)-bit values.

**Lemma 3.** For every pair of \( \ell \)-block messages \( M^0 \) and \( M^1 \) and \((q_\pi, t)\)-bounded adversary \( A^\pi \), it holds

\[
\begin{align*}
\Pr[A^\pi(\text{IdealS}(IV, A, M^0)) \Rightarrow 1] - \Pr[A^\pi(\text{IdealS}(IV, A, M^1)) \Rightarrow 1] \\
&\leq \ell \cdot \text{Adv}^{\text{LOR}[\pi]}(q_\pi, O(t + \ell t_1)),
\end{align*}
\]

where \( t_1 \) is as defined in Lemma 2.

These enable us to establish Eq. (10). In detail, for any \((q_\pi, t)\)-bounded adversary \( A^\pi \), we have

\[
\begin{align*}
\Pr[A^\pi(LDuStr_B[\pi](IV, A, M^0)) \Rightarrow 1] - \Pr[A^\pi(LDuStr_B[\pi](IV, A, M^1)) \Rightarrow 1] \\
&\leq \left[ \Pr[A^\pi(\text{IdealS}(IV, A, M^0)) \Rightarrow 1] - \Pr[A^\pi(\text{IdealS}(IV, A, M^1)) \Rightarrow 1] \right] \\
&\quad + \sum_{b=0,1} \left| \Pr[A^\pi(LDuStr_B[\pi](IV, A, M^b)) \Rightarrow 1] - \Pr[A^\pi(\text{LidealS}(IV, A, M^b)) \Rightarrow 1] \right|.
\end{align*}
\]

For each \( b \), Lemma 2 indicates

\[
\begin{align*}
\Pr[A^\pi(LDuStr_B[\pi](IV, A, M^b)) \Rightarrow 1] - \Pr[A^\pi(\text{LidealS}(IV, A, M^b)) \Rightarrow 1] \\
&\leq \text{Adv}^{\text{Inv}[\pi]}(q_\pi, O(t + \ell t_1), 2q_\pi) + (\ell + 1) \cdot \text{Adv}^{\text{Inv}[\pi]}(q_\pi, O(t + \ell t_1), 2q_\pi) + (\ell + 2)^2 2^{c+1}.
\end{align*}
\]
Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge

Therefore,

$$\begin{align*}
\Pr(A^r(\text{LDuStr}_B[\pi](IV, A, M^o)) \Rightarrow 1) - \Pr(A^r(\text{LDuStr}_B[\pi](IV, A, M^1)) \Rightarrow 1)
\leq & \frac{(\ell + 2)^2}{2^\kappa} + \ell \cdot \text{Adv}^{\text{LOR}}(q_\pi, O(t + \ell t_t)) + 2\text{Adv}^{\text{Inv}[c]}(q_\pi, O(t + \ell t_t), 2q_\pi) \\
& + 2(\ell + 1) \cdot \text{Adv}^{\text{Inv}[c]}(q_\pi, O(t + \ell t_t), 2q_\pi),
\end{align*}$$

as claimed. \qed

### 3.3.1 Proof of Lemma 2

The crux is to bound certain “bad events” during the execution of \text{IdealS}, for which we leverage the non-invertible leakage assumption. For simplicity, wlog we consider the case of $|A| = 0$, as this maximizes the amount of information obtained from the encrypting process.

**Preparations.** Denote $G_1(\mathcal{D}, \text{LDuStr}_B[\pi], \pi)$ and $G_2(\mathcal{D}, \text{IdealS}, \pi)$ the games capturing the interactions between $\mathcal{D}$ and the real (LDuStr$_B$[\pi], \pi) and the ideal (\text{IdealS}, \pi) resp., and simplify them as $G_1$ and $G_2$. We will prove the indistinguishability of $G_1$ and $G_2$, using the $H$-coefficient technique. To this end, note that the real adversarial transcripts could be summarized as two lists $\tau_r$ and $\tau_\pi$: the former includes the ciphertext as well as the leakages (its concrete representation won’t be needed in this proof), while the latter $\tau_\pi = ((S_0^n, S_0^{out}), \ldots, (S_q^n, S_q^{out}))$ includes the adversarial permutation queries and responses, and indicates the $i$-th query is either forward $\pi(S_i^n) \rightarrow S_i^{out}$ or backword $\pi^{-1}(S_i^{out}) \rightarrow S_i^n$. Besides, at the end of the interaction, we reveal the involved internal state values $S = (S_0, S_1, \ldots)$ to $\mathcal{D}$ (see Figures 2 or 4 for the meaning of $S_0, S_1, \ldots$), and append it to the transcript. Clearly, this does not reduce its advantage.

Note that the interactions with the (real or ideal) stream cipher additionally rely on $r$, the random coins of the distinguisher $\mathcal{D}$ & the involved leakage functions. Yet, it can be seen that during two games $G_1(\mathcal{D}, \text{LDuStr}_B[\pi_1], \pi_1)$ and $G_2(\mathcal{D}, \text{IdealS}, \pi_2)$, if the following conditions are fulfilled, then the queries and responses of $\mathcal{D}$ are the same, and thus $\mathcal{D}$ outputs the same:

- $\pi_1(S_n^i) = S_{out}^i$ and $\pi_2(S_n^i) = S_{out}^i$ for any $(S_n^i, S_{out}^i) \in \tau_\pi$;
- the internal state values produced in the two games are the same $S$;
- the random coins $r$ used in $G_1$ and $G_2$ are the same.

It is because all the internal actions in $G_1(\mathcal{D}, \text{LDuStr}_B[\pi_1], \pi_1)$ and $G_2(\mathcal{D}, \text{IdealS}, \pi_2)$ give rise to the same results (note that various other randomness sources such as the coins of the leakage functions have also been included in the above). With the above considerations, we summarize all the randomness in what we call *extended transcripts*, i.e., $\tau = (\tau_\pi, S, r)$.

Note that $\tau_r$ is absent in $\tau$, as it can be recovered from $r$ and $S$.

With respect to some fixed distinguisher $\mathcal{D}$, an extended transcript $\tau = (\tau_\pi, S, r)$ is *attainable* if there exists randomness $(r, \pi)$ such that using $r$, the ideal execution of $G_2(\mathcal{D}, \text{IdealS}, \pi)$ yields $(\tau_\pi, S)$. We denote $T$ the set of attainable transcripts.

Given a set $\tau_\pi$ and a random permutation $\pi$, we say that $\pi$ extends $\tau_\pi$, denoted $\pi \vdash \tau_\pi$, if $\pi(S_n^i) = S_{out}^i$ for all $(S_n^i, S_{out}^i) \in \tau_\pi$. It is easy to see that for any attainable transcript $\tau = (\tau_\pi, S, r)$, the event $T_{id} = \tau$ happens if and only if $\pi \vdash \tau_\pi$, $S$ is generated in the execution, and the randomness $r$ is used (by $\mathcal{D}$ and $L$). This means

$$\Pr[T_{id} = \tau] = \Pr[\pi \vdash \tau_\pi] \cdot \Pr[r] \cdot \Pr[S \text{ is sampled in the ideal world}] = \Pr[\pi \vdash \tau_\pi] \cdot \Pr[r] \cdot \frac{1}{2^\kappa} \cdot \frac{1}{2^{(\ell + 1)b}},$$

(11)
As long as the
To this end,
for an event
BadInner
collide, the condition (B-1) cannot be fulfilled. Therefore,

\[
Pr[\text{BadInner} \in \mathcal{T}_{\text{bad}}]. \quad \text{An attainable transcript } \tau = (\tau, S, r) \text{ is bad, if any of following conditions is fulfilled:}
\]

- (B-1) contradiction between state value: there exists two distinct indices i, j ∈ [0, ..., ℓ] such that \( S_i' = S_j' \land S_{i+1} \neq S_{j+1} \), or \( S_i' \neq S_j' \land S_{i+1} = S_{j+1} \);
- (B-2) exposure of secret state: if any of the following is fulfilled:
  - there exists i ∈ {0, ..., ℓ} such that \( (S_i', \ast) \in \tau \); or
  - there exists i ∈ {1, ..., ℓ + 1} such that \( (\ast, S_i) \in \tau \).

As long as the c-bit (uniformly distributed) capacity pieces of \( S_0', S_1, S_2, ..., S_{\ell+1} \) don’t collide, the condition (B-1) cannot be fulfilled. Therefore,

\[
Pr[(B-1)] \leq \left( \frac{\ell + 2}{2} \right)^2 \cdot \frac{(\ell + 2)^2}{2^{c+1}}. \quad (12)
\]

To bound \( Pr[(B-2)] \), we need the non-invertible leakage assumption. Consider an execution of \( G_2 \) with the inputs \((IV, M)\). We divide (B-2) into three subevents:

1. BadInner: that occurs when there exists an index i ∈ {1, ..., ℓ} such that \((\ast, S_i) \in \tau\) or \((S_i', \ast) \in \tau\);
2. BadInit: that occurs when \((S_0', \ast) \in \tau\);
3. BadFinal: that occurs when \((\ast, S_{\ell+1}) \in \tau)\.

Consider BadInner first. We follow Yu et al. [YSPY10, Appendix A] (i.e., their argument for an event Query, with somewhat similar meaning): given an adversary \( \mathcal{D}^\tau \), we construct an adversary \( \mathcal{A}^\tau \) such that

\[
\mathbf{Adv}_{\text{inv}[c]}^{\text{iv}[c]}(\mathcal{A}^\tau) \leq Pr_{\tau, S, \pi}[\text{BadInner in } \mathcal{D}^\tau(\text{IdealS}(IV, M))]. \quad (13)
\]

To this end, \( \mathcal{A}^\tau \) runs an instance of \( \mathcal{D} \), and keeps \( \tau, \pi \), i.e., the set of \( \mathcal{D}^\tau \)'s queries to \( \pi \). \( \mathcal{A}^\tau \) simulates the following process against \( \mathcal{D} \):

1. \( \mathcal{A}^\tau \) guesses an index i ∈ [1, ℓ], samples a key \( B \leftarrow \{0, 1\}^c \), sets \( S_0' \leftarrow IV \parallel B \), and initializes a list leak with the leakages \( L^{\text{in}}(S_0') \);
2. For j = 1, ..., i − 1, \( \mathcal{A}^\tau \) samples the new state \( S_j \leftarrow \{0, 1\}^b \), computes \( C[j] \leftarrow \text{msb}(S_j) \parallel M[j] \), and:
   - \( \mathcal{A}^\tau \) computes \( S_j' \leftarrow C[j] || (\text{lsb}(S_j) \parallel \delta_1) \) and adds the leakages \( L^{\text{in}}(\text{lsb}(S_j), \delta_1) \) to leak when \( j = 1 \), and
   - \( \mathcal{A}^\tau \) computes \( S_j' \leftarrow C[j] || \text{lsb}(S_j) \) otherwise.
3. \( \mathcal{A}^\tau \) further adds the leakages \( L^{\text{out}}(S_j), L^{\text{out}}(S_j'), \text{ and } L^{\text{in}}(\text{msb}(S_j), M[j]) \) to leak;
4. \( \mathcal{A}^\tau \) samples \( y \leftarrow \{0, 1\}^c \) and computes \( C[i] \leftarrow y \parallel M[i] \). \( \mathcal{A}^\tau \) then submits \( y, C[i] \), and \( \delta = \delta_1 \) when \( i = 1 \) or \( \delta = 0 \) (when \( i > 1 \)) to its \( \text{inv}[c] \) challenger and obtains the leakages
   \[
   \text{leak}_{ch} = \left[ L^{\text{out}}(y \parallel s_{\text{ch}}), L^{\text{in}}(s_{\text{ch}}, \delta), L^{\text{in}}(C[i] || (s_{\text{ch}} \parallel \delta)) \right]
   \]
   for the challenge secret \( s_{\text{ch}} \in \{0, 1\}^c \). \( \mathcal{A}^\tau \) then adds the leakages \( L^{\text{in}}(y, M[i]) \), \( \text{leak}_{ch} \) to leak. This means \( y \parallel s_{\text{ch}} \) is taken as \( S_i \)—though \( \mathcal{A}^\tau \) doesn’t know its value.
5. \( \mathcal{A}^\tau \) then completes the remaining encrypting: for \( j = i + 1, ..., \ell \), \( \mathcal{A}^\tau \) samples \( S_j \leftarrow \{0, 1\}^b \), computes \( C[j] \leftarrow \text{msb}(S_j) \parallel M[j] \), \( S_j' \leftarrow C[j] || \text{lsb}(S_j) \), and adds \( L^{\text{out}}(S_j), L^{\text{out}}(S_j'), \) and \( L^{\text{in}}(\text{msb}(S_j), M[j]) \) to leak.
(5) Finally, $A^\pi$ samples $S_{i+1} \leftarrow \{0,1\}^b$, adds $L^\text{out}_i(S_{i+1})$ to the list leak, returns $(C[1]|\ldots|C[\ell], \text{leak})$ to $D$, and outputs $G := \{\text{lsb}_b(S^\text{in}), \text{lsb}_c(S^\text{out}) : (S^\text{in}, S^\text{out}) \in \tau_\pi\}$. The strategy of $A^\pi$ is to make a uniform guess on the position of the first inner secret value that appears in $\tau_\pi$, as this value is the “first”, its being queried was necessarily due to the corresponding leakages (rather than the compromising of the other inner states). This guess will be correct with probability $1/\ell$. Then, $A^\pi$ simulates $\text{IdealS}(IV,M)$ and provides the leakages to $D$, except for the $i$ index, for which the leakages are replaced by those obtained from an $\text{Inv}$ challenger. Now if the guess on the index $i$ is correct, then all the inputs sent to $D$ are distributed exactly as those in a normal execution of $G_2$. Therefore, when $D$ halts, if $D$ made a query with $s_{ch}$, then outputting the aforementioned set $G$ (based on $\tau_\pi$) would break the $\text{Inv}$ game. So we have

$$\Pr[s_{ch} \in G \mid \text{BadInner in } G_2(D, \text{IdealS}, \pi)] \geq \frac{1}{\ell}.$$ 

Now, we observe that

$$\Pr[s_{ch} \in G \mid \text{BadInner in } G_2(D, \text{IdealS}, \pi)] \leq \frac{\Pr[s_{ch} \in G]}{\Pr[\text{BadInner in } G_2(D, \text{IdealS}, \pi)]}.$$

And it can be seen $A$ is $(q_\pi,t^*,2q_\pi)$-bounded, with $t^* = O(t + \ell t_1)$. By this,

$$\Pr[\text{BadInner in } G_2(D, \text{IdealS}, \pi)] \leq \ell \cdot \Pr[s_{ch} \in G] \leq \ell \cdot \text{Adv}^{\text{inv}[\pi]}(A) \quad (\text{Eq. (3)})$$

$$\leq \ell \cdot \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi). \quad (\text{Eq. (5)})$$

These finish the analysis of BadInner. For the events BadInit and BadFinal, similar arguments could establish

$$\Pr[\text{BadInit}] \leq \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi), \quad \Pr[\text{BadFinal}] \leq \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi).$$

The three terms plus Eq. (12) yield

$$\Pr[T_{\text{id}} \in T_{\text{good}}] \leq (\ell + 1) \cdot \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi) + \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi) + \frac{\ell + 2}{2^{\ell + 1}}. \quad (14)$$

**Summarizing.** For any good transcript $\tau = (\tau_\pi, S, r) \in T_{\text{good}}$, we have

$$\Pr[T_{\tau_\pi} = \tau] = \Pr[\pi \vdash \tau_\pi] \cdot \Pr[r] \cdot \Pr[S \text{ is derived in the real world}] = \Pr[\pi \vdash \tau_\pi] \cdot \Pr[r] \cdot \frac{1}{2^\ell} \cdot \Pr[\forall i \in [0,\ell] : \pi(S_i) = S_{i+1} \mid \pi \vdash \tau_\pi].$$

Conditioned on $\neg(\text{B-1})$ and $\neg(\text{B-2})$, it can be seen that, for all $i \in \{0,\ldots,\ell\}$, it holds $\Pr[\pi(S_i) = S_{i+1} \mid \pi \vdash \tau_\pi \land \forall j < i : \pi(S_j) = S_{j+1}] \geq \frac{1}{2^\ell}$. This means $\Pr[\forall i \in \{0,\ldots,\ell\} : \pi(S_i) = S_{i+1} \mid \pi \vdash \tau_\pi] \geq \frac{1}{2^{\ell+1}}$. Therefore, $\Pr[T_{\tau_\pi} = \tau] \geq \Pr[T_{\text{id}} = \tau]$ by Eq. (11), and thus $\Pr[T_{\text{id}} \in T_{\text{good}}]$ constitutes the final bound, i.e.,

$$\Pr[D^\pi(\text{LDuStr})[\pi](IV,M)] \Rightarrow 1 - \Pr[D^\pi(\text{IdealS}(IV,M)] \Rightarrow 1\right] \leq \frac{(\ell + 2)^2}{2^{\ell + 1}} + (\ell + 1) \cdot \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi) + \text{Adv}^{\text{inv}[\pi]}(q_\pi,t^*,2q_\pi).$$
3.3.2 Proof of Lemma 3

Again we assume $|A| = 0$ for simplicity. Let $M_0 = M_0[0] \ldots M_0[\ell]$ and $M_1 = M_1[0] \ldots M_1[\ell]$. We start by building a sequence of $\ell + 1$ messages $M_{h,0}, \ldots, M_{h,\ell}$ starting from $M_0$ and modifying its blocks one by one, to obtain $M_1$. That is, $M_{h,i} := M_0[i] \ldots \|M_0[i+1] \ldots M_0[\ell]$. For any $i$, assuming a $(q_e, t)$-bounded adversary $A^\tau$ against $L_{IdealS}(IV, M_{h, i-1})$ and $L_{IdealS}(IV, M_{h, i})$, we build a $(q_e, O(t + \ell t))$-bounded adversary $A^\tau_2$ against the distribution defined in Eq. (6). In detail, $A^\tau_2$ proceeds in four steps:

1. $A^\tau_2$ samples $B \leftarrow \{0, 1\}^c$, initializes an empty list leak, and sets $S'_0 \leftarrow IV \| B$; 
2. for $j = 1, \ldots, i - 1$, $A^\tau_2$ samples $S_j \leftarrow \{0, 1\}^b$, computes $C[j] \leftarrow m_{sb}(S_j) \oplus M_0[j]$, $S'_j \leftarrow C[j]|\|\{\text{msb}(S_j) \oplus \delta_i\}$ when $j = 1$, and adds $L_{\text{out}}(S'_{j-1}), L_{\text{out}}(S_j)$, and $L_{\text{in}}(m_{sb}(S_j), M_0[j])$ (and $L_{\text{in}}(\text{msb}(S_j), M_1[j])$) to leak; 
3. $A^\tau_2$ samples $s \leftarrow \{0, 1\}^c$ and submits $s$ to the LORL challenger. Assume that the outputs are $(c^s, \text{leak}_s)$ with 

\[
\text{leak}_s = [L_{\text{out}}(y_{ch} \| s), L_{\text{in}}(y_{ch}, m^b)].
\]

Then adds $L_{\text{out}}((M_0[i+1] \| 0^c) \oplus S_{i-1}), L_{\text{out}}(y_{ch} \| s)$, and $L_{\text{in}}(y_{ch}, m^b)$ to leak; 
4. $A^\tau_2$ starts from $c^s|C$ to emulate the remaining actions of $L_{IdealS}$ encrypting the tail $M_0[i+1] \ldots M_0[\ell]$ to obtain $C[i+1] \ldots C[\ell]$. Eventually, $A^\tau_2$ serves the ciphertext $C[0] \ldots C[i-1]|c^s|C[i+1] \ldots C[t]$ (and $\text{msb}(S_{i+1})$, when $= 1$) as well as all the generated simulated leakages to $A^\tau$, and outputs whatever $A^\tau$ outputs.

It can be seen depending on whether the input tuple received by $A^\tau_2$ captures the LORL challenger encrypting $M_0[i]$ or $M_1[i]$, the inputs to $A^\tau$ capture $L_{IdealS}$ encrypting $M_{h, i-1}$ or $M_{h, i}$. Moreover, $A^\tau_2$ is $(q_e, O(t + \ell t))$-bounded if $A^\tau$ is $(q_e, t)$-bounded. Therefore,

\[
\Pr[A^\tau(L_{IdealS}(IV, M_{h, i-1})) = 1] - \Pr[A^\tau(L_{IdealS}(IV, M_{h, i})) = 1] 
\leq A_{\text{adv-LORL}}(q_e, O(t + \ell t))
\]

by Eq. (8). This along with a simple summation implies the main claim.

3.4 Discussion

Where does leakage-resistance come from? We note that, in (the model of) a duplex, the internal state is kept evolving by calling a perfectly random permutation. Since the adversary cannot recover the secret, he cannot predicate what would be the next internal permutation call. Therefore, the subsequent state is derived via a fresh permutation-call and thus random and secret.

On reducing to PRP assumptions. An alternative solution to study keyed sponge/duplex is to reduce them to the PRP security of a “Partial-Key” Even-Mansour (PKEM) cipher [ADMV15]. It is thus natural to ask whether $EavL$ could also be reduced to the PRP security of PKEM with leakages. While this direction is in general an interesting open problem, the tricky issue is that the PKEM-based representation in [ADMV15] depicted in Figure 5 contains a plenty of “imaginary” XORs that do not actually happen in reality. How to model the leakage of these XORs is not directly obvious.

Forward security. Keyed sponges are not forward secure [BDPV10], in the sense that the exposure of a $b$ bit state enables recovering the earlier states. Consequently, the final permutation call must hide its output state to some extent. This is captured by the bad condition BadFinal in the proof of Lemma 2.
Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge

Figure 5: The PKEM-based representation of a keyed sponge with $\kappa$-bit key/initial seed $B$. The “partial-key” Even-Mansour cipher is inside the dashed rectangles. Internal actions of XORing $B$ cancel. The construction turns basically the same as the keyed sponge.

4 Description of TET Sponge

We design TET Sponge to turn the single-user EavL-secure duplex stream cipher into an authenticated encryption with stronger multi-user security guarantees. The specification, Section 4.2, will come after some intuitions in Section 4.1.

4.1 Design considerations

By Theorem 1, the duplex is a nice starting point for efficient 1-pass leakage-resilient AEs. Indeed, the final duplex state is typically truncated as the tag. But typically, in a leveled implementation, only the initial keyed part would be strongly protected against side-channel attacks, and this causes the concern of decryption leakages. First, decryption leakages enable recovering the internal state of (a leveled implementations of) the duplex [AFM19], after which universal forgery is possible. Second, verifying the tag requires to perform the entire computation to recover the valid tag and to compare it with the one included in the ciphertext. Hence, this valid tag may be leaked, constituting trivial forgeries and breaking CIML2. It is natural to ask how to remedy those issues.

Towards the first issue, our observation is that an additional keyed finalization function helps. Namely, even if the internal state has been recovered in full, the construction then collapses to a Hash-then-MAC authenticator, with the duplex being a keyless hash and the keyed finalization being a fixed length MAC. Thus, if the finalization is carefully protected, strong integrity with nonce-misuse and leakages is restored.

Towards the second issue, assuming $n$-bit tag size, then an approach to cope with the leaking tag is to rely on an $n$-bit invertible primitive (e.g., a block cipher was used in [BPPS17]), so that the verification only compares (and leaks) pre-images of tags without ever computing the valid tag, which then remains unpredictable. Inversion may be challenging though, as the duplex permutation size $b$ is typically much larger than any ordinary tag size $n$. Due to the presence of leakages, it is hard to build a “small” invertible primitive from the large permutations (Feistel-based solution consumes $O(n)$ rounds [DP10]).

With these considerations, we use a tweakable block cipher (TBC) for the finalization to increase robustness against decryption leakages. In detail, we derive a $2n$-bit (hash) digest from the keyed duplex, and then use an $(n,n,n)$-TBC to absorb this digest and generate the $(n$-bit) tag. This solves all the above. The motivation for employing $2n$-bit digests is to boost the complexity of hash collision-based forgery attacks to $2^n$, which is inherited from [BGP+19].

Of course, if the resistance to side-channel attacks is not a primary concern in a specific implementation, it remains possible to only use the TBC in forward mode, and recompute the valid tag as part of the ciphertext verification process. This may simplify
the implementation a bit, but results in a weaker protection against side-channel attacks.

But when side-channel attacks are in scope, the TBC shall be well protected. Therefore, following [PSV15, DEM+17, BGP+19], we also use it to derive the duplex-key from the nonce and "setup" the duplex. Now, as long as the internal state is not fully exposed by SCAs, it enables the inner keyed duplex to maintain some level of privacy and integrity; even if the state is leaked in full, the scheme collapses to Hash-then-TBC authenticator, and integrity is retained (so that severe malware attacks like [RSWO17] are not be possible here). The use of "public keys" in order to offer multi-user security beyond \(|K|/2\) is also inherited from [BGP+19]. Briefly, the public keys create a strong independence between different users and thus reduce the effectiveness of offline computation against the multiple user keys. Besides, the actions involving these public keys do not need any additional physical protection.

It is tempting to ask why bother integrity with nonce misuse-resistance and decryption leakages in a 1-pass scheme that is not designed for nonce-reuse at all and loses confidentiality against decryption leakages. The reason is that we choose 1-pass as a strict efficiency target but we push the achievable limit of (nonce and leakage) robustness as far as possible and extract the techniques to reach it.

### 4.2 Specification

**Parameters.** TET Sponge[\(\pi, \tilde{E}\)] is built on an \((n, n, n)\)-TBC \(\tilde{E}\) and a b-bit permutation \(\pi\) with \(b = r + c\). The key is \(K||PK\), where \(|K| = n\) and \(|PK| = n_p\). We stress that only \(K\) has to be kept secret, but \(PK\) can be public. The secret key \(K\) is picked uniformly at random in \(\{0, 1\}^n\). The public part \(PK\) should be unique per session for separation. The easiest way to non-interactively ensure uniqueness is to pick \(PK\) uniformly from \(\{0, 1\}^{n_p}\). If \(n_N = |N|\) denotes the fixed length of the nonces, we require that \(n_p \leq r\), \(n_N + n_p + n \leq r + c\), and \(2n \leq r + c + 1\). Yet, we recommend \(n_p \approx n\) and \(c \approx 2n\) and we actually choose \(n_p = n - 1\) and \(c = 2n\) as this leads to muCIML2 (and black-box CCA) security up to \(2^n/n^2\) queries.

![Figure 6: TET Sponge][1] \(\pi, \tilde{E}, PK\) AEAD for \(\nu = 2\) blocks of associated data and \(\ell = 2\) message blocks. Dark squares indicate the "leak-free" TBC where the triangle denotes the key input and the small black rectangle denotes the tweak input. The value \(1||0^{\ell-2}\) is inserted only if \(|A[\nu]| < r\), resp. \(|M[\ell]| < r\).

**The encryption.** As shown in Fig. 6, upon encrypting \((N, A, M)\), the mode first derives an \(n\)-bit initial seed \(B\) from \(N\), using a strongly protected TBC-call to \(\tilde{E}^{PK||0^*}(N||0^*)\). The seed \(B\) is then used as the key of the duplex to process \(A\) and \(M = M[1]|\ldots|M[\ell]\) and produce \(c = C[1]|\ldots||C[\ell]\). Note that 2 bits are used for domain separation, in order to distinguish \(M\) from \(A\) and to mark if the last blocks of \(A\) and \(M\) are of full \(r\) bits or not.

Let \(U||V\) be the most significant \(2n - 1\) bits of the final state with \(|U| = n\). As discussed, another strongly protected TBC-call is made, which generates the \(n\)-bit tag \(Z = \tilde{E}^{V||0^*}(U)\). The final ciphertext is \(C[1]|\ldots||C[\ell]|Z\).
The decryption. Upon decrypting $(N, A, C)$, $C = C[1] \ldots \|C[t]\|Z$, the mode first recovers the initial seed $B$ via $B = E_{PK}^{0^t}(N\|0^s)$, and then runs the duplex on $A$ and $C[1] \ldots \|C[t]\|$ to recover $M$ and the $2n - 1$ bit truncated state $U\|V$. Finally, it makes an inverse TBC call $U^* = (E_{K}^{V})^{-1}(Z)$, and outputs $M$ if and only if there is a match $U = U^*$. In such a way, invalid decryption only leaks meaningless random values $U^*$ (instead of the correct tag) which increases robustness against decryption leakages at the mode level (formally, helps achieve CIML2).

The encryption and decryption are formally described in Figure 7. Note that when $c$ is empty while $A$ passes the integrity checking, Dec explicitly returns a special value true, so that it can be used for authenticating $A$.

Note that since we expect the duplex to act as a keyless hash when the internal state has been leaked, we cannot follow the more efficient design approaches that encroach the capacity of the duplex/sponge — including full-state [MRV15, DMV17] and concurrent absorption [SY15].

5 Idealizing the TBC calls in TETSPONGE

We will prove lower bounds for the leakage integrity and confidentiality of TETSPONGE in the subsequent two sections. Both of the two proofs will rely on an idealized TETSPONGE variant TETSPONGE$_{\pi,\bar{S}IC_{K,PK}}$, which internally makes calls to a secret ideal TBC $\bar{S}IC$ instead of the public $IC$.

This means the key derivation and tag generation calls in TETSPONGE$_{\pi,\bar{S}IC_{K,PK}}$ will not be influenced by the adversarial queries (to $IC$).

We will consider a setting motivated by the unbounded leakage assumption: a distinguisher is interacting with either the real world (TETSPONGE$_{\pi,IC_{K,PK}}$, $\pi$, $IC$) or the idealized world (TETSPONGE$_{\pi,\bar{S}IC_{K,PK},\pi,IC}$), but all the intermediate values inside the mode TETSPONGE$_{\pi,IC_{K,PK}}$/TETSPONGE$_{\pi,\bar{S}IC_{K,PK}}$ are completely given. Note that this means the adversary could observe inputs/outputs of the internal TBC-calls from the interaction with TETSPONGE$_{\pi,IC_{K,PK}}$/TETSPONGE$_{\pi,\bar{S}IC_{K,PK}}$. Though, he cannot choose the inputs/outputs in arbitrary. For example, for the internal TBC-calls due to key derivations, the tweak has to be of the form $PK|0$ with $PK$ uniformly distributed, the properties of which will be helpful for our proof. We show that $D$ cannot distinguish the two worlds up to $\approx 2^n/n^2$ queries. Formally,

Lemma 4. Under the unbounded leakage assumption, it holds

$$\left| \Pr[D_{\text{TETSPONGE}}|\pi,\bar{S}IC_{K,PK},\pi,\pi^{-1},\bar{IC},\bar{IC}^{-1} \Rightarrow 1] - \Pr[D_{\text{TETSPONGE}}|\pi,IC_{K,PK},\pi,\pi^{-1},\bar{IC},\bar{IC}^{-1} \Rightarrow 1]\right| \leq \frac{u}{2^{n_p}} + \frac{16Q^2}{2^n} + \frac{(2n + 1)Q + n^2q_{IC}}{2^n}.$$  \hspace{1cm} (15)

The remaining of this section are all devoted to prove Lemma 4. The idea is simple, while the details are intricate due to the analysis of the non-standard sponge hashing.

5.1 Preparations

We rely on the $H$-coefficient technique [CS14]. For this, we summarize the information involved during the adversarial interaction in transcripts. The definition, which is a bit complicated, is divided into several paragraphs for cleanness.

3Regarding this idealization, note that we cannot simply replace the TBC-calls of the $u$ users by calls to $u$ independent tweakable random permutations, as otherwise the birthday term $\frac{u^2}{2^u}$ emerges.
algorithm \texttt{Enc}_{K,PK}(N,A,M)
\begin{algorithmic}
\State $\ell \leftarrow \lceil |M|/r \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
\State parse $M$ as $M[1]\ldots M[\ell]$, with $|M[1]| = \ldots = |M[\ell-1]| = r$ and $1 \leq |M[\ell]| \leq r$
\State parse $A$ as $A[1]\ldots A[\nu]$, with $|A[1]| = \ldots = |A[\nu-1]| = r$ and $1 \leq |A[\nu]| \leq r$
\State $B \leftarrow E_K^{PK}[0^{n-n_P}](N|0^{n-n_N})$
\State $IV \leftarrow N||PK||0^{n-n_N-n_P}$
\State $S_0 \leftarrow IV||B$, $S_1 \leftarrow \pi(S_0)$
\If{$\nu \geq 1$}
\For{$i = 1$ to $\nu - 1$}
\State $S_i \leftarrow S_i \oplus (A[i]|0^\nu)$
\State $S_{i+1} \leftarrow \pi(S_i)$
\EndFor
\If{$|A[\nu]| < r$}
\State $A[\nu] \leftarrow A[\nu]|10^{\nu-|A[\nu]|-1}$
\State $S_0 \leftarrow S_0 \oplus (0^\nu||[1]_2|0^{\nu-2})$
\State $S_\nu \leftarrow S_\nu \oplus (A[\nu]|0^\nu)$
\State $S_{\nu+1} \leftarrow \pi(S_\nu)$
\EndIf
\If{$\ell \geq 1$}
\State $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^\nu||[2]_2|0^{\nu-2})$
\EndIf
\If{$i = 1$ to $\ell - 1$}
\State $j \leftarrow i + \nu$
\State $C[i] \leftarrow \text{msb}_{\nu}(S_j) \oplus M[i]$
\State $S_j \leftarrow C[i]\|\text{lsb}_\nu(S_j)$
\State $S_{j+1} \leftarrow \pi(S_j)$
\State $C[\ell] \leftarrow \text{msb}_{\nu\ell}(S_{\nu+\ell}) \oplus M[\ell]$
\EndIf
\EndIf
\EndAlgorithm
\end{algorithmic}

algorithm \texttt{Dec}_{K,PK}(N,A,C)
\begin{algorithmic}
\State $\ell \leftarrow \lceil |C|-n \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
\State parse $C$ as $C[1]\ldots C[\ell]|Z$, with $|C[1]| = \ldots = |C[\ell] = r$, $1 \leq |C[\ell]| \leq r$, and $|Z| = n$
\State parse $A$ as $A[1]\ldots A[\nu]$, with $|A[1]| = \ldots = |A[\nu-1]| = r$ and $1 \leq |A[\nu]| \leq r$
\State $B \leftarrow \tilde{E}^{PK}_{\nu}[0^{n-n_P}](N|0^{n-n_N})$
\State $IV \leftarrow N||PK||0^{n-n_N-n_P}$
\State $S_0 \leftarrow IV||B$, $S_1 \leftarrow \pi(S_0)$
\If{$\nu \geq 1$}
\For{$i = 1$ to $\nu - 1$}
\State $S_i \leftarrow S_i \oplus (A[i]|0^\nu)$
\State $S_{i+1} \leftarrow \pi(S_i)$
\EndFor
\If{$|A[\nu]| < r$}
\State $A[\nu] \leftarrow A[\nu]|10^{\nu-|A[\nu]|-1}$
\State $S_\nu \leftarrow S_\nu \oplus (0^\nu||[1]_2|0^{\nu-2})$
\State $S_\nu \leftarrow S_\nu \oplus (A[\nu]|0^\nu)$
\State $S_{\nu+1} \leftarrow \pi(S_\nu)$
\EndIf
\EndIf
\If{$\ell \geq 1$}
\State $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^\nu||[2]_2|0^{\nu-2})$
\EndIf
\EndIf
\EndAlgorithm

\begin{figure}[h]
\centering
\begin{tabular}{c}
\textbf{Specification of TET-Sponge[π,E] using a b-bit permutation $\pi$ and an $(n,n,n)$-TBC $\tilde{E}$}.
\end{tabular}
\end{figure}
Permutation query transcripts. We summarize the adversarial queries to the random permutation $\pi$ in
\[
\tau_\pi = (\{ S_{\pi,0}^1, S_{\pi,0}^{\text{out}} \}, \ldots, \{ S_{\pi,n}^\omega, S_{\pi,n}^{\text{out}} \}).
\]
From our unbounded-leakage assumption, all the (encryption and decryption) queries made to TETSpSonge give all the ephemeral intermediate values to $\mathcal{D}$. Therefore, these queries result in records of the form $(S_{\pi,0}^1, S_{\pi,0}^{\text{out}})$. To make a distinction, we denote by $\tau_\pi^*$ the union of these leakage records and the adversarial query transcript $\tau_\pi$. It can be seen that:

- upon an encryption query $\text{Enc}_{K_i,PK_i}(N, A, M)$, TETSpSonge makes at most $\lceil \frac{|A|}{r} \rceil + \lceil \frac{|M|}{r} \rceil + 1$ queries to $\pi$;
- upon a decryption query $\text{Dec}_{K_i,PK_i}(N, A, C)$, the number of internal $\pi$-calls is (similarly) at most $\lceil \frac{|A|}{r} \rceil + \lceil |C| \rceil + 1$, where $C = c || Z$.

Therefore, when interacting with the idealized scheme TETSpSonge[$\pi$, $\text{SI[C]}_K$, $PK$], the number of internal $\pi$-calls is at most $\sigma + q_e + q_d$, and we have
\[
Q := |\tau_\pi^*| \leq \sigma + q_e + q_d + q_\pi.
\]

Hashing transcripts. Due to unbounded leakage assumption, for every query with the nonce $N$, the public-key $PK$, the associated data $A$, and the ciphertext $c$, the duplex behaves as evaluating a keyless hash function on $(N, PK, A, c)$, and all the underlying permutation calls are known by the adversary. Note that $c$ may be directly specified by a decryption query, or the response of an encryption query; though, this does not matter, since the inputs that affect the hash outputs are the same. For convenience, we introduce $\tau_\nu^*$, which is defined as the list of the inputs and outputs of this duplex hashing. Formally, a tuple $((N, PK, A, c, S), N \in \{0, 1\}^n, PK \in \{0, 1\}^n, A, c \in \{0, 1\}^*$, and $S \in \{0, 1\}^0$ is in $\tau_\nu^*$ if and only if there exists a sequence of $\pi$ queries $(S_{\nu, 0}^1, S_{\nu, 0}^{\text{out}})$, $(S_{\nu, 1}^1, S_{\nu, 1}^{\text{out}})$, ..., $(S_{\nu,n}^\omega, S_{\nu,n}^{\text{out}})$ in $\tau_\nu^*$ such that:

- $S_{\nu,0}^n = N || PK || 0^n || B$ for some $B \in \{0, 1\}^n$,
- With $\nu = \lceil |A|/r \rceil$ and $\ell = \lceil |M|/r \rceil$, it holds $\omega = \nu + \ell$, and
  - For $i = 1, \ldots, \nu - 1$, $\text{lsb}_c(S_{\nu,i}^1) = \text{lsb}_c(S_{\nu,i}^{\text{out}})$, $\text{msb}_c(S_{\nu,i}^1) \oplus \text{msb}_c(S_{\nu,i}^{\text{out}}) = A[i]$;
  - When $\lceil |A|/r \rceil < \nu$, it holds $\text{lsb}_c(S_{\nu}^1) = \text{lsb}_c(S_{\nu-1}^{\text{out}}) \oplus (0^r || [1]_2 || 0^{|C[\ell]|} - 2)$ and $\text{msb}_c(S_{\nu}^1) \oplus \text{msb}_c(S_{\nu-1}^{\text{out}}) = A[|\nu|] 10^e$; when $|A|/r = \nu$, it holds $\text{lsb}_c(S_{\nu}^1) = \text{lsb}_c(S_{\nu-1}^{\text{out}})$ and $\text{msb}_c(S_{\nu}^1) = A[|\nu|];$
  - If $\ell > 1$:
    - $\text{lsb}_c(S_{\nu,i}^{\text{in}}) = \text{lsb}_c(S_{\nu,i}^{\text{out}}) \oplus (0^r || [2]_2 || 0^{|C[\ell]|} - 2)$, and $\text{msb}_c(S_{\nu,i}^{\text{in}}) = C[i+1]$;
    - For $i = \nu + 2, \ldots, \omega - 1$, $\text{lsb}_c(S_{\nu,i}^1) = \text{lsb}_c(S_{\nu,i}^{\text{out}})$, $\text{msb}_c(S_{\nu,i}^1) = C[i + 1]$;
    - When $|C[\ell]| < \nu$, $\text{lsb}_c(S_{\nu}^1) = \text{lsb}_c(S_{\nu-1}^{\text{out}}) \oplus (0^r || [1]_2 || 0^{|C[\ell]|} - 2)$, $\text{msb}_c(S_{\nu}^1) = C[|\ell|] 10^e$; when $|C[\ell]| = \nu$, $\text{lsb}_c(S_{\nu}^1) = \text{lsb}_c(S_{\nu-1}^{\text{out}})$, and $\text{msb}_c(S_{\nu}^1) = C[|\ell|].$
  - If $\ell = 1$:
    - $\text{lsb}_c(S_{\nu,0}^1) = \text{lsb}_c(S_{\nu,0}^{\text{out}}) \oplus (0^r || [3]_2 || 0^{|C[\ell]|} - 2)$, $\text{msb}_c(S_{\nu,0}^1) = C[|\ell|] 10^e$; when $|C[\ell]| = \nu$, $\text{lsb}_c(S_{\nu,0}^1) = \text{lsb}_c(S_{\nu,0}^{\text{out}}) \oplus (0^r || [2]_2 || 0^{|C[\ell]|} - 2)$, and $\text{msb}_c(S_{\nu,0}^1) = C[|\ell|].$
- $S = S_{\nu}^\omega$.

Note that the $(2n - 1)$-bit hashing digest is $U || V = \text{msb}_{2n-1}(S_{\nu})$.

Keys. The “public-keys” $PK = (PK_1, \ldots, PK_u)$ are certainly in the adversarial transcripts. Following [CS14], we make one step further and augment the transcript with the secret user keys $K = (K_1, \ldots, K_u)$. Formally, we will append both $PK$ and $K$ to the transcript.
5.2 Establishing Eq. (15)

First of all, for a transcript $\tau$, we define two quantities $\mu_{PK}$ and $\mu_V$, the maximum multiplicity of $PK$ and $V$, as

$$
\mu_{PK} := \max_{pk \in \{0,1\}^n} \left| \{ i \in \{1,\ldots,u \} : PK_i = pk \} \right|,
$$

and

$$
\mu_V := \max_{v \in \{0,1\}^{n-1}} \left| \{ (N,PK,A,c),S \in \tau^*_K : V = v \} \right|.
$$

(17)

With these two quantities, we start the main argument by defining bad transcripts.

5.2.1 Bad transcripts for idealizing TETSponge

**Definition 4.** An attainable transcript $\tau$ is bad, if any of the following is fulfilled:

- (B-1) $\mu_{PK} \geq n + 1$, $\mu_V \geq n + 1$.
- (B-2) there exists a query $(K,T,X,Y) \in \tau^*_\text{SIC}$ such that $(K,T,*,*) \in \tau^*\text{IC}$.

Otherwise $\tau$ is good. The set of bad transcripts is $\mathcal{T}_{\text{bad}}$.

**Intuitions of the conditions.** $G_2$ deviates from $G_1$ in the use of the secret ideal TBC $\hat{\text{IC}}$. The difference thus stems from the contradictions between $\tau^*_\text{IC}$ and $\tau^*_{\text{SIC}}$, which is captured by (B-2). Though, the straightforward approach to bound $\Pr[(B-2)]$ gives rise to a poorer bound $\Pr[(B-2)] = O(uq^2/2^n)$. To rescue, we note that a query $(K,T,*,*) \in \tau^*_{\text{IC}}$ can only “hit” queries in $\tau^*_{\text{SIC}}$ with the specific tweak $T$. Thus precisely, the probability that each adversarial query to $\hat{\text{IC}}$ is $O(\mu_T/2^n)$ rather than $O(|\tau^*_{\text{SIC}}|/2^n)$, where $\mu_T$ is the maximum number of times a specific tweak value appears in $\tau^*_{\text{SIC}}$. All these considerations motivated us to introduce the condition (B-1).
Analyzing (B-1). As $PK_1, \ldots, PK_u$ are uniformly distributed, we have

$$\Pr[\mu_{PK} \geq n + 1] \leq \left( \frac{u}{n + 1} \right) \cdot \left( \frac{1}{2} \right)^n \leq \left( \frac{u}{2^n} \right)^{n+1} \leq \left( \frac{u}{2^n} \right)^{n+1},$$

where the last inequality comes from $(n+1)! \geq \left( \frac{n+1}{e} \right)^{n+1} \geq 2^{n+1} \geq 2^n$ since $n+1 \geq 6 > 2e$. Furthermore, when $u \leq 2^n$ and $n_p \leq n$, we have

$$\Pr[\mu_{PK} \geq n + 1] \leq \left( \frac{u}{2^n} \right)^{n+1} \leq \frac{u}{2^n}. \quad (18)$$

To reason about $\mu_V$, we analyze the multi-semicollision property of the involved sponge-based hashing. In detail, we consider the game $G_2$ capturing the interaction of $D$ with the ideal world ($\text{TET Sponge} [\pi, \text{SIC}]_K, PK, \pi, IC$). We define several simple bad events during this interaction:

- (B-11) Right after a forward query $\pi(S_i^{\text{in}}) \rightarrow S_i^{\text{out}}$ is made (by either the adversary or the construction $\text{TET Sponge} [\pi, \text{SIC}]$), there exists an (earlier) query $(S_i', S_i')$ such that $\text{lsb}_{c-2}(S_i^{\text{out}}) = \text{lsb}_{c-2}(S_i')$ or $\text{lsb}_{c-2}(S_i^{\text{out}}) = \tilde{\text{lsb}}_{c-2}(S_i')$.
- (B-12) Right after a backward query $\pi^{-1}(S_i^{\text{out}}) \rightarrow S_i^{\text{in}}$ is made,
  - there exists another query $(S_i', S_i')$ such that $\text{lsb}_{c-2}(S_i^{\text{in}}) = \tilde{\text{lsb}}_{c-2}(S_i')$;
  - or, there exists an SIC query $(K, PK_i, 0, N, 0^*, B) \in \tau_{\text{SIC}}$ (which is due to $\text{TET Sponge}$ deriving a key $B$) such that $S_i^{\text{in}} = N||PK_i||0^*||B$.
- (B-13) At any time, there exist $n + 1$ forward queries $(S_1^{\text{in}}, S_1^{\text{out}}), \ldots, (S_{n+1}^{\text{in}}, S_{n+1}^{\text{out}})$ such that $\text{mid}(S_1^{\text{out}}) = \ldots = \text{mid}(S_{n+1}^{\text{out}})$, where $\text{mid}(S_i^{\text{out}}) = \text{msb}_{2^n-1}(\tilde{\text{lsb}}_{c-1}(S_i^{\text{out}}))$, i.e., extracting $n - 1$ bits from the middle.
- (B-14) Right after a forward SIC query $\tilde{\text{SIC}}_K^{PK||0^*} (N||0^*) \rightarrow B$ is made, there exists a query $(S_i^{\text{in}}, S_i^{\text{out}})$ such that $S_i^{\text{in}} = N||PK||0^*||B$. Note that this concerns with the TBC call made by $\text{TET Sponge}$ during deriving initial keys.

We denote by $q_1$ the number of forward queries, and by $q_2$ that of backward queries. Clearly, $q_1 \leq Q$, $q_2 \leq q_e$ (as $\text{TET Sponge}$ does not make backward $\pi$ queries), and $q_1 + q_2 \leq Q$. With these, consider a forward query $\pi(S_i^{\text{in}}) \rightarrow S_i^{\text{out}}$. Its response $S_i^{\text{out}}$ is uniformly distributed in a set of size at least $2^b - Q$. Consider any “target” $(S_i^{\text{in}}, S_i^{\text{out}})$. To have $\text{lsb}_{c-2}(S_i^{\text{out}}) = \text{lsb}_{c-2}(S_i^{\text{in}})$, $S_i^{\text{out}}$ shall be in a set of size at most $2^{b+2}$. Therefore, when $Q \leq 2^{b}/2$, we have

$$\Pr[\text{lsb}_{c-2}(S_i^{\text{out}}) = \text{lsb}_{c-2}(S_i^{\text{in}})] \leq \frac{2^{b+2}}{2^b - Q} \leq \frac{2^{b+3}}{2^b} = \frac{8}{2^e}.$$ 

This calculation trick will be frequently used in the remaining analysis without explicitly mentioning. Similarly, $\Pr[\text{lsb}_{c-2}(S_i^{\text{out}}) = \text{lsb}_{c-2}(S_i^{\text{in}})] \leq \frac{8}{2^e}$. As the number of “targets” is at most $Q$, we have

$$\Pr[(B-11)] \leq q_1 \cdot Q \cdot \left( \frac{8}{2^e} + \frac{8}{2^e} \right) \leq \frac{16q_1Q}{2^e}.$$ 

In a similar vein, it can be seen (using $(n+1)! \geq 2^{n+1}$) that

$$\Pr[(B-12)] \leq q_2 \cdot Q \cdot \frac{8}{2^e} + q_2 \cdot (q_e + q_d) \cdot \frac{2}{2^e} \leq \frac{10q_2Q}{2^e}, \quad \text{and}$$

$$\Pr[(B-13)] \leq \left( \frac{q_1}{n+1} \right)^n \cdot \left( \frac{2}{2^{n-1}} \right)^n \leq \left( \frac{8Q}{2^e} \right)^{n+1} \cdot \frac{1}{8(n+1)}! \leq \left( \frac{4Q}{2^e} \right)^{n+1} \cdot \frac{1}{8} \leq \frac{Q}{2^n},$$

since $4Q \leq 2^n$. 

Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge
To bound \( \Pr[(B-14)] \), we define a set
\[
\tau^*_a[N, PK] := \{ B \in \{0, 1\}^n : (N \| PK \| 0^* \| B, *) \in \tau^*_a \}.
\] (19)

Then, for a certain \( \tilde{\text{SIC}} \) query \( \tilde{\text{SIC}}^P_{K}(N \| 0^*) \rightarrow B \), we have
\[
\Pr [B \in \tau^*_a[N, PK]] \leq \frac{|\tau^*_a[N, PK]|}{2^n} \leq \frac{2|\tau^*_a[N, PK]|}{2^n}.
\]

Summing over all the \( \tilde{\text{SIC}} \) queries, we find
\[
\Pr[(B-14)] \leq \sum_{(K, PK\| 0^*, N\| 0^*, B) \in \tau_{\tilde{\text{SIC}}}} \frac{2|\tau^*_a[N, PK]|}{2^n} \leq \sum_{i=1}^{n} 2 \sum_{N \in \{0, 1\}^n} \frac{|\tau^*_a[N, PK]|}{2^n} \leq \mu_{PK} \cdot \sum_{N \in \{0, 1\}^n} \frac{2|\tau^*_a[N, PK]|}{2^n} \leq \frac{2nQ}{2^n},
\]

since \( \sum_{N \in \{0, 1\}^n, PK \in \{0, 1\}^n} |\tau^*_a[N, PK]| = |\tau^*_a| \leq Q \).

Let \( \text{Bad}_1 := (B-11) \lor (B-12) \lor (B-13) \lor (B-14) \). We now show that \( \mu_V \leq n \) given \( \neg \text{Bad}_1 \), so that \( (q_1 + q_2 \leq Q) \)
\[
\Pr[\mu_V \geq n + 1] \leq \Pr[\text{Bad}_1] \leq \frac{16q_1Q}{2^n} + \frac{10q_2Q}{2^n} + \frac{Q}{2^n} + \frac{2nQ}{2^n} \leq \frac{16Q^2}{2^n} + \frac{(2n + 1)Q}{2^n}.
\]

For this, we argue that distinct hashing evaluations induce distinct final \( \pi \) queries, which is stated formally in the following lemma.

**Lemma 5.** Conditioned on \( \neg \text{Bad}_1 \), we show that two distinct tuples \( ((N, PK, A, c), S) \) and \( ((N', PK', A', c'), S') \) in \( \tau^*_a \) necessarily have \( S \neq S' \). Moreover, both \( S \) and \( S' \) were the outputs of forward \( \pi \) queries during the interaction.

The proof is deferred to subsection 5.3 for cleanliness. By Lemma 5, the “final” \( \pi \) queries of the \( |\tau^*_a| \) hash records are \( |\tau^*_a| \) distinct forward queries. Conditioned on \( \neg \text{(B-13)} \), the number of semi-collisions on \( V \) within these final \( \pi \) queries is at most \( n \). Therefore, the claim \( \mu_V \leq n \) follows.

**Analyzing (B-2) conditioned on \( \neg \text{(B-1)} \).** Note that in the ideal world, for any query \( (K, T, X, Y) \in \tau_{\tilde{\text{SIC}}} \), the key \( K \) is from the dummy key-tuple \( \mathbf{K} \), and is uniformly distributed. Then, using an auxiliary set
\[
\tau^*_c[T] := \{ K \in \{0, 1\}^n : \langle K, T, *, * \rangle \in \tau_{\tilde{\text{SIC}}} \},
\]
we can bound
\[
\Pr[(B-2)] \leq \sum_{\langle K, T, *, * \rangle \in \tau_{\tilde{\text{SIC}}}} \Pr [K \in \tau^*_c[T]] \leq \sum_{t \in \{0, 1\}^{n-1}; \langle K, t, 0^* \rangle \in \tau_{\tilde{\text{SIC}}}} \frac{|\tau^*_c[t]|}{2^n} + \sum_{V \in \{0, 1\}^{n-1}; \langle K, V, 1^* \rangle \in \tau_{\tilde{\text{SIC}}}} \frac{|\tau^*_c[V]|}{2^n}.
\]

\[C_1\] and \[C_2\]
By the construction, the $\bar{\text{SiC}}$ queries $(K, t || 0, *, *)$ are necessarily key derivation queries, for which $K = K_i$ and $t || 0 = PK_i || 0^*$ for some user index $i$. Since $\mu_{PK} \leq n$, we have

$$C_1 = \sum_{i=1}^{u} \frac{\tau_{IC}[PK_i][0^*]}{2^n} \leq \mu_{PK} \cdot \sum_{PK \in \{0, 1\}^n} \frac{\tau_{IC}[PK][0^*]}{2^n} \leq n \cdot \sum_{PK \in \{0, 1\}^n} \frac{\tau_{IC}[PK][0^*]}{2^n}.$$ 

Moreover, for any $\bar{\text{SiC}}$ query $(K_i, V || 1, *, *)$, i.e., tag generation query, there necessarily exists at least one hash record $((N, PK, A, c), S) \in \tau_{ic}^*$ such that $PK = PK_i$. By this,

$$C_2 = \sum_{i=1}^{u} \frac{\tau_{IC}[V || 1]}{2^n} \leq \mu_{PK} \cdot \sum_{PK \in \{0, 1\}^n} \frac{\tau_{IC}[V || 1]}{2^n} \leq n \cdot \sum_{V \in \{0, 1\}^{n-1}} \frac{\tau_{IC}[V || 1]}{2^n}.$$ 

Therefore,

$$\Pr[\{(B-2) | \neg(B-1)\} \leq n \cdot \sum_{PK \in \{0, 1\}^n} \frac{\tau_{IC}[PK][0^*]}{2^n} + n^2 \cdot \sum_{V \in \{0, 1\}^{n-1}} \frac{\tau_{IC}[V || 1]}{2^n} \leq n^2 \cdot \sum_{t \in \{0, 1\}^n} \frac{\tau_{IC}[t]}{2^n} \leq \frac{n^2 q_{\text{ic}}}{2^n},$$

which allows us to conclude

$$\Pr[T_{id} \in \mathcal{T}_{bad}] \leq \Pr[(B-1)] + \Pr[(B-2) | \neg(B-1)] \leq \frac{u}{2n^p} + \frac{16Q^2}{2^n} + \frac{(2n + 1)Q + n^2 q_{\text{ic}}}{2^n}.$$ 

5.2.2 Finishing the application of the H-coefficient technique

Now consider a good transcript $\tau = (\tau_{ik}^*, \tau_{ic}^*, \tau_{ic}, \text{PK}, K)$. Let

$$\tau_{ic}^* [K, T] := \{(X, Y) \in \{0, 1\}^n \times (K, T, X, Y) \in \tau_{ic}^* \}.$$ 

With this notation, it is clear that

$$\Pr[T_{id} = \tau] = \Pr[\text{PK}] \cdot \Pr[\pi \vdash \tau_{ik}^*] \cdot \Pr[I\hat{C} \vdash \tau_{ic}^*] \cdot \prod_{(K, T)} \frac{1}{(2^n)^{\tau_{ic}^* [K, T]}}.$$ 

Furthermore,

$$\Pr[T_{re} = \tau] = \Pr[\text{PK}] \cdot \Pr[\pi \vdash \tau_{ik}^*] \cdot \Pr[I\hat{C} \vdash \tau_{ic}^* \mid \bar{I}\hat{C} \vdash \tau_{ic}] \cdot \Pr[I\hat{C} \vdash \tau_{ic}]$$

$$= \Pr[\text{PK}] \cdot \Pr[\pi \vdash \tau_{ik}^*] \cdot \Pr[I\hat{C} \vdash \tau_{ic}^* \mid \bar{I}\hat{C} \vdash \tau_{ic}] \cdot \Pr[I\hat{C} \vdash \tau_{ic}] \cdot \Pr[I\hat{C} \vdash \tau_{ic}],$$

Since $\tau$ is good,

$$\Pr[I\hat{C}_{k}(X) = Y \text{ for all } (K, T, X, Y) \in \tau_{ic}^* \mid \bar{I}\hat{C} \vdash \tau_{ic}]$$

$$= \Pr[I\hat{C}_{k}(X) = Y \text{ for all } (K, T, X, Y) \in \tau_{ic}^*] = \prod_{(K, T)} \frac{1}{(2^n)^{\tau_{ic}^* [K, T]}}.$$
In conclusion, for any good transcript τ we have $\Pr[T_{re} = \tau] = \Pr[T_{md} = \tau]$, and thus

$$|\Pr[D_{\text{TET Sponge}}[\pi, \mathcal{C}, \mathbf{PK}; \pi, \pi^{-1}, \mathcal{I}, \mathcal{C}^{-1}] = 1] - \Pr[D_{\text{TET Sponge}}[\pi, \mathcal{C}, \mathbf{PK}; \pi, \pi^{-1}, \mathcal{I}, \mathcal{C}^{-1}] = 1]| \leq \frac{u}{2^{2n}} + \frac{16Q^2}{2^c} + \frac{(2n + 1)Q + n^2q_\mathcal{C}}{2^n}.$$

### 5.3 Proof of Lemma 5

Assume that the query chains underlying the two hashing ((N, PK, A, c), S) and ((N′, PK′, A′, c′), S′) are (S′′, S′′′), ..., (S′′, S′′′), ..., (S′′, S′′′) respectively, and let $|[A]/r| = \nu$, $|[M]/r| = \ell$, $|[A′]/r| = \nu′$, and $|[M′]/r| = \ell′$. Then proving $S \neq S′$ means proving $S′_n \neq S′_n$. Note that, conditioned on $\neg(B-12)$ and $\neg(B-14)$, the first records (S′′, S′′′) and (S′′, S′′′) were necessarily resulted from forward $\pi$ queries. Then, by iteratively applying $\neg(B-12)$, it can be seen all the queries in the two chains are forward: this establishes the second claim. To show $S′_n \neq S′_n$, we distinguish several cases as follows.

#### 5.3.1 Case 1: (N, PK) ≠ (N′, PK′)

Then $S′_n \neq S′_n$, i.e., the two chains are already distinct at the first $\pi$ queries. By $\neg(B-11)$, we have $S′_n \neq S′_n$; similarly, iteratively applying $\neg(B-11)$ eventually results in the desired conclusion $S′_n \neq S′_n$.

#### 5.3.2 Case 2: (N, PK) = (N′, PK′)

This means (A, c) ≠ (A′, c′). We define $\overline{X}$ as:

- $\overline{X} = X$ when $|X|$ is a multiple of $r$, and
- $\overline{X} = X||10^n$ otherwise.

Then we have to consider several subcases.

#### Subcase 2.1: $\overline{A}/e \neq \overline{A′}/e’$

It is clear that there exists an index $i$ such that $S′_i \neq S′_i$. By $\neg(B-11)$, $S′_i \neq S′_i$ for any $j > i$, and thus $S′_n \neq S′_n$.

#### Subcase 2.2: $\overline{A}/e = \overline{A′}/e′$ and $\nu = \nu′$.

Since (A, c) ≠ (A′, c′), it has to be $|A[\nu]| < r$ or $|C[\ell]| < r$ or $|A′[\nu′]| < r$ or $|C′[\ell′]| < r$. Now,

- If $|A[\nu]| < r$ and $|A′[\nu′]| < r$, then $S′_n \neq S′_n$ due to the separation constant $[1]2||0^{n-2}$. Thus by $\neg(B-11)$, $S′_j \neq S′_j$ for any $j > \nu$ and further $S′_n \neq S′_n$.
- Else, either $|C[\ell]| < r$ or $|C′[\ell′]| < r$, then $S′_n \neq S′_n$ due to the separation constant $[2]2||0^{n-2}$.

#### Subcase 2.3: $\overline{A}/e = \overline{A′}/e′$, and $\nu \neq \nu′$.

Wlog $\nu > \nu′$: then it has to be $\ell′ \geq 1$. Now,

- If $|A[\nu′]| < r$, then $S′_n \neq S′_n$ since the separation constant $[1]2||0^{n-2}$ is only XORed into $S′_n$ and all the subsequent calls are distinct.
- Else, if $\ell′ = 1$, then $S′_n \neq S′_n$ since $S′_n$ is obtained by XORing $[3]2||0^{n-2}$ with $S′_n$ while $S′_n$ is obtained by XORing either $[1]2||0^{n-2}$ or 0′ (depending on whether $|A[\nu′]| < r$).
- Else, i.e., $\ell′ > 1$, then $S′_n \neq S′_n$ since $S′_n$ is obtained by XORing $[2]2||0^{n-2}$ with $S′_n$ while $S′_n$ is obtained by XORing either $[1]2||0^{n-2}$ or 0′.

In all, in all cases, we will have $S′_n \neq S′_n$. Thus the claims.
6 Leakage integrity of TETSPONGE

6.1 Concrete leakage assumptions and result

For TETSPONGE, even very weak implementations could ensure integrity: as long as the protected TBC calls are secure against key recovery attacks (e.g., DPA exploiting multiple queries to the TBCs), integrity is ensured even if all the other intermediate values (in the duplex) are leaked in full. Such a leakage assumption was previously called “unbounded” [BKP+18]. Formally, we define \( L^* = (L^*_\text{Enc}, L^*_\text{Dec}) \), where:

- \( L^*_\text{Enc} \) consists of the following information appearing during the encryption:
  - \( \{S^{in}, S^{out}\} \) for each internal call to \( \pi(S^{in}) \rightarrow S^{out} \), and
  - \( \{T, X, Y\} \) for each internal call to \( E^K(X) \rightarrow Y \) or \( (E^K)^{-1}(Y) \rightarrow X \) (i.e., all values are completely leaked except for the key \( K \)), and
  - \( \{a, b\} \) for each internal XOR action \( a \oplus b \).

- \( L^*_\text{Dec} \) consists of the above that are generated during the decryption.

We further write \( \vec{q} = (q_c, q_d, q_\text{CIML}_2, q_\pi) \), and denote by \( (\vec{q}, \sigma) \)-adversaries the adversaries that respectively make \( q_c, q_d, q_\text{CIML}_2, \) and \( q_\pi \) queries to \( L^*_\text{Enc} \), \( L^*_\text{Dec} \), \( \text{IC} \), and \( \pi \), and have at most \( \sigma \) blocks (of \( r \) bits) in all their queried plaintext and ciphertext including associated data. With these, we have the following result for TETSPONGE.

**Theorem 2.** Assume \( u \leq 2^n \), \( n_p \leq n \), \( n \geq 5 \), \( Q = \sigma + q_c + q_d + q_\pi \leq \min \{2^n/4, 2^b/2\} \), and leakage \( L^* \) is “unbounded” as above. Then in the ideal TBC and permutation model, for any \( (\vec{q}, \sigma) \)-adversary \( A \) it holds

\[
\text{Adv}^{\text{muCIML2}}_{\mathcal{A}, \text{TETSPONGE}, L^*} \leq \frac{3u}{2^n} + \frac{32Q^2}{2^n} + \frac{7nQ + n^2q_\pi}{2^n}.
\]  

(20)

Inside the bounds, \( q_\pi, q_\text{CIML}_2 = t \) reflect the time complexity. If \( n = 128 \), and with our chosen parameters \( n_p = n - 1 \) and \( c = 2n \), the bound simplifies to \( \frac{3u}{2^n} + \frac{21Q(t + 2\sigma)}{2^n} \), implying high security up to \( 2^{124} \) users, \( 2^{114} \) computations, and roughly \( 2^{120} \) message blocks.

6.2 Proof of Theorem 2

Lemma 4 indicates that the forgery probability of a \( \vec{q} \)-bounded adversary \( A \) against the real construction \( \text{TETSPONGE}[\pi, \text{IC}] \) is close to that against the ideal \( \text{TETSPONGE}[\pi, \text{IC}] \), i.e.,

\[
\text{Adv}^{\text{muCIML2}}_{\mathcal{D}, \text{TETSPONGE}[\pi, \text{IC}], \mathcal{K}, PK, L^*} - \text{Adv}^{\text{muCIML2}}_{\mathcal{D}, \text{TETSPONGE}[\pi, \text{IC}], \mathcal{K}, PK, L^*} \leq \frac{u}{2^n} + \frac{16Q^2}{2^n} + \frac{(2n + 1)Q + n^2q_\pi}{2^n}.
\]  

(21)

The remaining part is devoted to bounding \( \text{Adv}^{\text{muCIML2}}_{\mathcal{D}, \text{TETSPONGE}[\pi, \text{IC}], \mathcal{K}, PK, \mu, L^*} \). Consider the muCIML2 game capturing the interaction between \( A \) and \( \text{TETSPONGE}[\pi, \text{IC}], \mathcal{K}, PK \). We define an event \( \text{CHAIN} \): at any time, for the \( i \)-th user there exists a hash record \((N, PK_i, A, c), S)\) and a \( \text{IC} \) query \((K_i, V^* || 1, U^*, Z)\) (that corresponds to generating a tag) such that \( \text{msb}_{2n-1}(S) = U^* || V^* \), while there was no encryption query of the form \( L^*_\text{Enc}(i, N, A, \ast) \rightarrow c || Z \). Obviously, it is impossible to compute a forgery as long as \( \text{CHAIN} \) does not happen.

To ease the analysis, we split \( \text{CHAIN} \) into several simple bad events and show that \( \text{CHAIN} \) cannot happen as long as none of these events occur. Concretely,

- (C-1) There exist two user indices \( j, \ell \) such that \( K_j || PK_j = K_\ell || PK_\ell \), or \( \mu_{PK} \geq n + 1 \).
• (C-2) Right after a forward $\pi$ query $\pi(S^{in}) \rightarrow S^{out}$ is made,
  - (C-21) there exists an (earlier) $\pi$ query $(S^{in}', S^{out}')$ such that $\text{lsb}_{c-2}(S^{out}) = \text{lsb}_{c-2}(S^{in})$, $\text{lsb}_{c-2}(S^{out}) = \text{lsb}_{c-2}(S^{out})$, or $\text{msb}_{2n-1}(S^{out}) = \text{msb}_{2n-1}(S^{out})$;
  - (C-22) there exists a $\text{SIC}$ query $(K, V || 1, U, Z)$ (that corresponds to a tag generation) such that $\text{msb}_{2n-1}(S^{out}) = U || V$; or
• (C-3) At any time, there exist $n+1$ forward $\pi$ queries $(S^1, S^{out}), \ldots, (S^{n+1}, S^{out})$ such that $\text{mid}(S^{out}) = \ldots = \text{mid}(S^{out})$, where $\text{mid}(S^{out}) = \text{msb}_{2n-1}(\text{lsb}_{c-1}(S^{out}))$. 
• (C-4) Right after a backward $\pi$ query $\pi^{-1}(S^{out}) \rightarrow S^{in}$ is made, if:
  - (C-41) there exists another $\pi$ query $(S^{in}', S^{out})$ such that $\text{lsb}_{c-2}(S^{in}) = \text{lsb}_{c-2}(S^{out})$, or
  - (C-42) there exists a $\text{SIC}$ query $(K, PK || 0^*, N || 0^*, B)$ (that corresponds to a key derivation) such that $S^{in} = N || PK || 0^* || B$. 
• (C-5) Right after a (necessarily forward) key derivation query $\text{SIC}^{PK || 0^*}(N || 0^*) \rightarrow B$ happens, there exists a $\pi$ query $(S^{in}, S^{out})$ such that $S^{in} = N || PK || 0^* || B$.
• (C-6) Right after an inverse $\text{SIC}$ query/tag generation query $(\text{SIC}^{-1}_K || 1) \rightarrow U$, there exists a $\pi$ query $(S^{in}, S^{out})$ such that $\text{msb}_{2n-1}(S^{out}) = U || V$.

Some of the conditions have been analyzed when proving Lemma 4. First, using $u \leq 2^{np} \leq 2^n$ we have

$$\text{Pr}[(C-1)] \leq \frac{u^2}{2^{n+np}} + \frac{u}{2^{np}} \leq \frac{2u}{2^{np}}.$$ 

Second, (C-21) is the condition (B-11) in Section 5.2 enhanced with $\text{msb}_{2n-1}(S^{out}) = \text{msb}_{2n-1}(S^{out})$, thus $\text{Pr}[(C-21)] \leq \frac{16q_1Q + 2Q^2}{2^{np}}$ (with $q_1$ being the number of forward $\pi$ queries).

And it is clear that $\text{Pr}[(C-22)] \leq \frac{2np}{2^{np}}$. Thus (using $q_e + q_d \leq Q$)

$$\text{Pr}[(C-2)] \leq \frac{16q_1Q}{2^n} + \frac{4Q^2 + 4(q_e + q_d)Q}{2^{2n}} \leq \frac{16q_1Q}{2^n} + \frac{Q + q_e + q_d}{2^n} \leq \frac{16q_1Q}{2^n} + \frac{2Q}{2^n}.$$ 

The last inequality stems from $Q \leq 2^n/4$.

The condition (C-3) is the same as the condition (B-13) in Section 5.2, thus $\text{Pr}[(C-3)] \leq \frac{Q}{2^n}$. The condition (C-4) is the condition (B-12) in Section 5.2, thus $\text{Pr}[(C-4)] \leq \frac{16q_1Q}{2^n}$. (C-5) is the condition (B-14) in Section 5.2, thus

$$\text{Pr}[(C-5) \mid \neg (C-1)] \leq \frac{2nQ}{2^n}. \quad (22)$$

Finally consider (C-6). Conditioned on $\neg(C-2)$ and $\neg(C-3)$, an analysis similar to the previous one for $\mu_V$ (also in Section 5.2) indicates that the number of distinct records $((N_1, PK_1, A_1, e_1, S_1), (N_2, PK_2, A_2, e_2, S_2), \ldots)$ in $\tau^*_K$ with $\text{mid}(S_1) = \text{mid}(S_2) = \ldots$ is at most $n$. Therefore, for each inverse query $(\text{SIC}^{-1}_K || 1) \rightarrow U$, there are at most $n$ “target” $U$ values, and thus

$$\text{Pr}[(C-6)] \leq \frac{4q_d}{2^n} \leq \frac{2nq_d}{2^n}. \quad (23)$$

Define $\text{Bad} := (C-1) \lor (C-2) \lor (C-3) \lor (C-4) \lor (C-5) \lor (C-6)$, then we have

$$\text{Pr}[\text{Bad}] \leq \frac{2u}{2^{np}} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n}. \quad (24)$$

Below we show $\text{Pr}[\text{CHAIN} \mid \neg \text{Bad}] = 0$, so that $\text{Pr}[\text{CHAIN}] \leq \text{Pr}[\text{Bad}]$. Assume otherwise, then consider the last adversarial action before $\text{CHAIN}$ happens:
6.2.1 Case 1: $A$ makes a $\pi$ query

If this query is forward, then the fact that it causes the event CHAIN contradicts $\neg(C-2)$; if this query is backward, then it contradicts $\neg(C-4)$.

6.2.2 Case 2: $A$ makes an encryption query $\text{LEnc}_{K,PK}(i,N,A,M)$

Note that all the key derivation queries/$\text{T Sic}$ queries of the form $(K_i,PK_i||0^r,X,Y) \in \tau_{\text{Sic}}$ have nothing to do with the CHAIN event. Regarding the remaining possibilities, we further distinguish two subcases:

- Subcase 2.1: a subsequent (forward) $\pi$ query causes CHAIN event. This again contradicts $\neg(C-2)$.
- Subcase 2.2: the subsequent (new) tag generation query $\text{T Sic}_{K,\pi}(U) \rightarrow Z$ causes the event CHAIN. Assume that the involved hash record is $((N,PK_i,A,c),S)$ which corresponds to the $i$-th user. The assumption means there exists another hash record of the $j$-th user $((N',PK_j,A',c'),S')$ such that $\text{msb}_{2n-1}(S) = \text{msb}_{2n-1}(S')$. To reach a contradiction, we further distinguish two cases:
  - Subcase 2.3.1: $(N,A,c) \neq (N',A',c')$. Then as argued before (in Section 5.2), conditioned on $\neg(C-21)$, the two involved hash chains are different, and thus $\text{msb}_{2n-1}(S) = \text{msb}_{2n-1}(S')$ would contradict $\neg(C-21)$.
  - Subcase 2.3.2: $(N,A,c) = (N',A',c')$. Then it has to be $i \neq j$. Now, if $K_i \neq K_j$, then the new Sic query $\text{T Sic}_{K,\pi}(U) \rightarrow Z$ has nothing to do with the $j$-th user. Otherwise, it holds $PK_i = PK_j$ by $\neg(C-1)$, which means the two involved hash chains are different, and thus $\text{msb}_{2n-1}(S) = \text{msb}_{2n-1}(S')$ would contradict $\neg(C-21)$.

6.2.3 Case 3: $A$ makes a decryption query

We also consider two subcases:

- Subcase 3.1: a subsequent (forward) $\pi$ query causes CHAIN event. Then it again contradicts $\neg(C-2)$.
- Subcase 3.2: the subsequent (new) tag generation query $(\text{T Sic}_{K,\pi})^{-1}(Z) \rightarrow U$ causes CHAIN event. This contradicts $\neg(C-6)$.

As conclusion, we have

$$\text{Adv}^{\text{nuCIML2}}_{2,\text{TET Sponge}||\pi,\text{T Sic}_{PK,L},\tau_{\text{Sic}}.u} \leq \Pr[\text{Bad}] \leq \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n}. \quad (25)$$

Together with Eq. (15) this yields Eq. (20) (using $4 < 5 < n$):

$$\frac{u}{2^{n_p}} + \frac{16Q^2}{2^n} + \frac{(2n + 1)Q + n^2q_d}{2^n} \leq \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n} \leq \frac{3u}{2^{n_p}} + \frac{32Q^2}{2^n} + \frac{5nQ + 2nq_d + n^2q_d}{2^n}. \quad (26)$$

7 Leakage confidentiality of TET Sponge

7.1 Concrete leakages assumption and result

For confidentiality the implementations shall satisfy the bounded leakages assumptions in Section 3, i.e., non-invertibility & bounded XOR leakages; and we will apply Theorem 1.
Following the notations of Section 3, we naturally define the leakage function \( L = (L_{\text{Enc}}, L_{\text{Dec}}) \) of our implementation as follows:

- \( L_{\text{Enc}} \) consists of the leakages that are generated during the encryption:
  - the leakages \( L_{\text{Enc}}^i(S^{in}) \) and \( L_{\text{Enc}}^{out}(S^{out}) \) generated by all the internal calls to \( \pi(S^{in}) \rightarrow S^{out} \), and
  - the leakages \( L_{\text{Enc}}(a, b) \) generated by all the internal actions \( a \oplus b \).
- \( L_{\text{Dec}} = \emptyset \) since \( \mu_{\text{CCamL1}} \) is CCA with encryption leakages only.

For the TBC \( \tilde{E} \), we simply assume its leakage function \( L_{\tilde{E}} \) returns nothing, i.e., \( \tilde{E} \) is (indistinguishable to) leak-free. All the analyses can be easily modified to incorporate non-empty \( L_{\tilde{E}} \), but we eschew for simplicity.

For \( q = (q_m, q_e, q_d, q_{\text{IC}}, q_\pi) \), \((q, t, \sigma)\)-adversaries are adversaries making at most \( q_m, q_e, q_d, q_{\text{IC}}, \) and \( q_\pi \) queries respectively to the non-challenge LEnc, the challenge LEnc, Dec, \( \text{IC} \), and \( \pi \), running at most in time \( t \), and having \( \sigma \) blocks in all their (challenge & non-challenge) queries including associated data.

**Theorem 3.** Assume \( u \leq 2^{n_p}, n_p \leq n, n \geq 5, \sigma + q_e + q_d + q_m + q_\pi \leq \min \{2^{n/4}, 2^{n/2}\}, \) and leakage \( L = (L_{\text{Enc}}, L_{\text{Dec}}) \) is defined as above. Then in the ideal TBC and permutation model, for any \((q, t, \sigma)\)-adversary \( A \), it holds

\[
\text{Adv}_{A, TET\text{Sponge}, L, u}^{\mu_{\text{CCamL1}}} \leq \frac{6u}{2^{n_p}} + \frac{65Q^2}{2^c} + \frac{8(n + 1)Q + 4nq_d + 2n^2q_{\text{IC}}}{2^n} + \sigma \text{Adv}_{L^{\text{ORL}}}^{L^{\text{ORL}}}(Q, t^*),
\]

where \( \text{Adv}_{L^{\text{ORL}}}^{L^{\text{ORL}}} \) and \( \text{Adv}_{L^{\text{ORL}}}^{L^{\text{ORL}}} \) are defined in Eqs. (8) and (5) resp, \( Q = \sigma + q_e + q_d + q_m + q_\pi \), \( t^* = O(t + \sigma t_1) \), and \( t_1 \) is the total time for evaluating \( L^{in} \) and \( L^{out} \).

The concrete security is mainly limited by the terms “inherited” from Theorem 1. As discussed, the terms \( 2q_e \text{Adv}_{L^{\text{ORL}}}^{L^{\text{ORL}}}(Q, t^*, 2Q) + 2(\sigma + q_\pi) \text{Adv}_{L^{\text{ORL}}}^{L^{\text{ORL}}}(Q, t^*, 2Q) \) are in \( O\left(q_e \cdot \frac{q_d + \sigma + t \mu_c}{2^{2c}}\right) + O\left(\sigma \cdot \frac{2^{n/2} + t \mu c}{2^{2c}}\right) \) for some specific parameters \( \mu_a \) and \( \mu_c \). Though, the influence of the number \( u \) of users on the security remains negligible: once \( u \leq 2^{n_p}/5 \), TET\text{Sponge} is secure up to the birthday \( 2^{n/2} \) complexity—it is smaller than \( 2^{c/2} \) due to the shorter initial seed \( B \).

### 7.2 Proof of Theorem 3

The core idea follows the standard eavesdropper-to-CPA reduction: to prove that encrypting \( M^0_1, M^0_2, \ldots \) is close to encrypting \( M^1_1, M^1_2, \ldots \), we will replace the encryption of \( M^0_1 \) by \( M^1_1 \) in turn, and the gap of each such replacement could be bounded by Theorem 1. We just need additional arguments for the decryption queries (due to the CCA setting).

Formally, we start by defining \( G_0 \) as the game \( \text{PrivK}_{\mu_{\text{CCamL1}}}^{\mu_{\text{CCamL1}}, 0} \) and \( G_0^* \) as the game \( \text{PrivK}_{\mu_{\text{CCamL1}}}^{\mu_{\text{CCamL1}}, 1} \). And we say a decryption query \( \text{Dec}_{\text{PK}}(i, N, A, C) \) is trivial if the encryption action \( \text{Enc}_{\text{PK}}(i, N, A, \ast) \rightarrow C \) happens before. The subsequent proof consists of several small steps presented in the following paragraphs.

#### 7.2.1 Idealizing TET\text{Sponge}

We introduce two new games \( G_1 \) and \( G_1^* \): \( G_1 \), resp. \( G_1^* \), is obtained from \( G_0 \), resp. \( G_0^* \), via replacing the internal IC-calls (made by TET\text{Sponge}[\pi, IC]) by SIIC-calls. By Lemma 4, we have

\[
| \text{Pr}[G_1 \Rightarrow 1] - \text{Pr}[G_0 \Rightarrow 1] | \leq \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n + 1)Q + n^2q_{\text{IC}}}{2^n},
\]
We then introduce two other intermediate games $G_2$ and $G_2^*$, which differ from $G_1$ and $G_1^*$ in that:

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

The gaps are actually the probability of forgery against the idealized TET Sponge construction, which could be found in Eq. (24), Section 6. Therefore,

$$
\left| \Pr[G_2 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1] \right| \leq \frac{2u}{2^n} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n},
$$

(29)

where $Q = \sigma + q_e + q_d + q_m + q_\pi$.

### 7.2.2 Excluding non-trivial decryption queries

We then introduce two other intermediate games $G_2$ and $G_2^*$, which differ from $G_1$ and $G_1^*$ in that:

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

The gaps are actually the probability of forgery against the idealized TET Sponge construction, which could be found in Eq. (24), Section 6. Therefore,

$$
\left| \Pr[G_2 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1] \right| \leq \frac{2u}{2^n} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n},
$$

(30)

$$
\left| \Pr[G_2^* \Rightarrow 1] - \Pr[G_1^* \Rightarrow 1] \right| \leq \frac{2u}{2^n} + \frac{16Q^2}{2^n} + \frac{(2n + 3)Q + 2nq_d}{2^n}.
$$

(31)

### 7.2.3 The main hybrid-based argument

Then, we denote the $q_e$ challenge tuples by

$$(i_1, N_1, A_1, M_1^0, M_1^1), \ldots, (i_{q_e}, N_{q_e}, A_{q_e}, M_{q_e}^0, M_{q_e}^1),$$

and use $q_e$ hops to replace $M_1^0, \ldots, M_{q_e}^0$ by $M_1^1, \ldots, M_{q_e}^1$ in turn, to show that $G_2$ can be transited to $G_2^*$. For convenience, we define $G_{3,0} = G_2$, and define a sequence of games

$G_{3,1}, G_{3,2}, \ldots, G_{3,q_e},$

such that in the $j$-th system $G_{3,j}$, the first $j$ messages processed by the challenge encryption oracle are $M_1^0, \ldots, M_{q_e}^0$, while the remaining $q_e - j$ messages being processed are $M_j^1, \ldots, M_{q_e}^1$. It can be seen that $G_{3,q_e} = G_2$.

We then show that for $j = 1, \ldots, q_e$, $G_{3,j-1}$ and $G_{3,j}$ are indistinguishable for $A^{\pi, \tilde{\pi}}$. For this, from $A^{\pi, \tilde{\pi}}$ we build an adversary $A_2^{\pi, \tilde{\pi}}$, such that $| \Pr[G_{3,j-1} \Rightarrow 1] - \Pr[G_{3,j} \Rightarrow 1]|$ is related to the bound proved in Theorem 1. In detail, initially, $A_2^{\pi, \tilde{\pi}}$ samples $u$ distinct keys $K_1 \parallel T_1, \ldots, K_u \parallel T_u$ for subsequent simulations. It also keeps a pair of tables (STABLE, STABLE$^{-1}$), which have entries of the form STABLE($K, T, X$) = $Y$ and STABLE$^{-1}$($K, T, Y$) = $X$, to simulate the secret ideal TBC $\text{SI}$ via lazy sampling. $A_2^{\pi, \tilde{\pi}}$ then runs $A^{\pi, \tilde{\pi}}$. Upon $A^{\pi, \tilde{\pi}}$’s action, $A_2^{\pi, \tilde{\pi}}$ reacts as follows.

#### Upon $A^{\pi, \tilde{\pi}}$ making a query to $\tilde{\pi}$ or $\pi$: $A_2^{\pi, \tilde{\pi}}$ simply relays the query to its corresponding oracle and relays the response.

#### Upon $A$ making a (non-challenge) encryption query $(i^*, N^*, A^*, M^*)$: $A_2^{\pi, \tilde{\pi}}$ distinguishes two cases.
If \((K_i^*, PK_i^*||0^*, N^*||0^*) \notin \text{STABLE}\), then \(A_2^{\pi, I_C}\) samples an initial key \(B^*\) such that \((K_i^*, PK_i^*||0^*, B^*) \notin \text{STABLE}^{-1}\), defines \(\text{STABLE}(K_i^*, PK_i^*||0^*, N^*||0^*) \leftarrow B^*\) and \(\text{STABLE}^{-1}(K_i^*, PK_i^*||0^*, B^*) \leftarrow N^*||0^*\), and then runs the encryption \(\text{DuStr}_B[\pi](N^*||PK_i^*||0^*, A^*, M^*)\) to get the ciphertext \(c^*||U^*||V^*\) and leakages.

\(A_2^{\pi, I_C}\) then computes \(Z^* \leftarrow \text{STABLE}(K_i^*, V^*||1, U^*)\) (if \((K_i^*, V^*||1, U^*) \notin \text{STABLE}\) then \(A_2^{\pi, I_C}\) defines the entry \(\text{STABLE}(K_i^*, V^*||1, U^*)\) to a newly sampled value \(Z^*\). For this entire process \(A_2^{\pi, I_C}\) has to make \(\ell_i + 1\) queries to \(\pi\) and consume \(O(\ell_i t_i)\) time. Finally, \(A_2^{\pi, I_C}\) returns the outputs \(c^*||Z^*\) and the leakages to \(A^{\pi, I_C}\).

**Upon \(A\) making a non-trivial decryption query \((i^*, N^*, A^*, C^*)\):** \(A_2^{\pi, I_C}\) parses \(C^* = c^*||Z^*\), and computes the hashing digest \(U^*||V^*\) accordingly. Then,

- if \((K_i^*, V^*||1, Z^*) \notin \text{STABLE}\), \(A_2^{\pi, I_C}\) samples \(U^*\) such that \((K_i^*, V^*||1, U^*) \notin \text{STABLE}\), and defines the two table entries as \(\text{STABLE}(K_i^*, V^*||1, U^*) \leftarrow Z^*\), \(\text{STABLE}^{-1}(K_i^*, V^*||1, Z^*) \leftarrow U^*\);
- if \((K_i^*, V^*||1, U^*) \in \text{STABLE}, A_2^{\pi, I_C}\) just sets \(U^* \leftarrow \text{STABLE}^{-1}(K_i^*, V^*||1, Z^*)\).

Now \(A_2^{\pi, I_C}\) returns \((\bot, U^*)\) to \(A^{\pi, I_C}\) (regardless of \(U^* = U^*\) or not). This requires \(A_2^{\pi, I_C}\) to make \(\ell_i + 1\) queries to \(\pi\).

**Upon \(A\) submitting the \(\ell\)-th challenge tuple \((i_\ell, N_\ell, A_\ell, M^0_\ell, M^1_\ell)\):** it necessarily holds \((K_i^*, PK_i^*||0^*, N_i^*||0^*) \notin \text{STABLE}\), since

- Previous encryption queries to the \(i_\ell\)-th user cannot define \(\text{STABLE}(K_i^*, T_i^*, C_0(N_i))\) due to the challenge nonce-respecting restriction on \(A\);
- Previous encryption queries to the \(i_\ell\)-th users cannot define \(\text{STABLE}(K_i^*, T_i^*, C_0(N_i))\) since \(K_i^*||T_i^* \neq K_i^*||T_i^*\).

With these in mind, depending on \(\ell, A_2^{\pi, I_C}\) reacts as follows:

- When \(\ell < j\), it encrypts \(M^0_\ell\) and returns. In detail, \(A_2^{\pi, I_C}\) samples \(B_\ell\), defines the entries \(\text{STABLE}(K_i^*, PK_i^*||0^*, N_i^*||0^*) \leftarrow B_\ell\) and \(\text{STABLE}^{-1}(K_i^*, PK_i^*||0^*, B_\ell) \leftarrow N_i^*||0^*\), and then runs \(\text{DuStr}_B[\pi](M^0_\ell) \rightarrow c^*||U_\ell||V_\ell\), generates the tag \(Z_\ell\) accordingly and returns \(c^*||Z_\ell\) to \(A^{\pi, I_C}\). The cost is similar to the non-challenge encryption queries.
- When \(\ell = j\), it relays \(M^0_\ell\) and \(M^1_\ell\) to the eavesdropper \(EavL\) challenger of \(\text{DuStr}\) to obtain \(c^*||U_\ell||V_\ell\) and leakages \(\text{leak}_{eavL}\), then generates the tag \(Z_\ell\) accordingly and returns \(c^*||Z_\ell\) to \(A^{\pi, I_C}\).
- When \(\ell > j\), it simply encrypts \(M^1_\ell\) and returns. The details are similar to the described case \(\ell < j\).

It can be seen that, depending on whether \(b = 0\) or \(1\), the whole process is the same as either \(G_{3,j-1}\) or \(G_{3,j}\). As remarked before, besides running \(A^{\pi, I_C}, A_2^{\pi, I_C}\) internally processes at most \(q_m + q_e + q_d\) queries. Therefore, for \(G_{2,j}\) and \(G_{2,j-1}\), \(A_2^{\pi, I_C}\) makes at
most $\sigma + q_m + q_e + q_d + q_\pi = Q$ queries to $\pi$ and spends $t^* = O(t + \sigma t_l)$ computations (mainly on evaluating the leakage functions). By all the above, we have

$$|\Pr[G_{3,j} \Rightarrow 1] - \Pr[G_{3,j-1} \Rightarrow 1]| \leq \text{Adv}_{E_{\text{avL}}^{DuStr}}(Q, t^*, \ell_j),$$

where the maximum is taken over all $A$ making $Q$ permutation queries, running in time $t^*$, and choosing inputs with $\ell_j$ blocks. Thus

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq \sum_{j=1}^{q_e} |\Pr[G_{3,j} \Rightarrow 1] - \Pr[G_{3,j-1} \Rightarrow 1]| \leq \sum_{j=1}^{q_e} \text{Adv}_{E_{\text{avL}}^{DuStr}}(Q, t^*, \ell_j).$$

(32)

Summing over Eqs. (28), (29), (30), (31), and (32), we finally have

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq \frac{6u}{2^n} + \frac{64Q^2}{2^c} + \frac{(8n + 8)Q + 4nq_d + 2n^2q_d}{2^n} + \sum_{j=1}^{q_e} \text{Adv}_{E_{\text{DuStr}}^{DuStr}}(Q, t^*, \ell_j).$$

(33)

The term $\sum_{j=1}^{q_e} \text{Adv}_{E_{\text{DuStr}}^{DuStr}}(Q, t^*, \ell_j)$ is bounded by Theorem 1 as

$$\frac{Q^2}{2^c} + \sigma \cdot \text{Adv}_{LORL}^2(Q, t^*) + 2q_e \cdot \text{Adv}_{\text{Inv}[\pi]}^2(Q, t^*, 2Q) + 2(\sigma + q_e) \cdot \text{Adv}_{\text{Inv}[\pi]}^2(Q, t^*, 2Q).$$

We thus have Eq. (27).

8 Conclusion

Sponge-based designs come with a complete state refresh at every iteration, a feature that has been thought to provide good protection against leakage for quite some time. In parallel with the recent work of Dobraniag and Mennink [DM19], we provide formal tools that support this intuition, using a weaker physical assumption of non-invertible leakages, but at the cost of more involved analyses. Due to their good efficiency properties, such Sponge-based designs appear as a useful ingredient for the design of lightweight modes of operation with resilience to leakage, as targeted by the ongoing NIST lightweight cryptography standardization process.

In the second part of the paper, we design and analyze TETSPONGE: a one-pass mode of operation that combines the good features of the duplex sponges with other recent advances in leakage-resistant designs in order to provide strong integrity with leakage in encryption and decryption, and strong confidentiality with leakage in encryption. Protecting against decryption leakage at the mode level would require a second pass, and we analyze proposals in this direction in the extended version of the paper. TETSPONGE also offers beyond birthday security in the multi-user setting and nonce misuse-resilience.

TETSPONGE directly serves as a basis for the Spook round-2 candidate to the NIST lightweight crypto standardization effort.\(^4\) Spook was designed with the idea of leveled implementations in mind, where only its two external TBCs are strongly protected against side-channel attacks. Its components (i.e., the TBC and the permutation) were selected in order to allow efficient masking (so minimum AND complexity and depth) while also enabling resource sharing for unprotected implementations. It is expected that this

\(^4\)https://www.spook.dev/
approach should deliver significant energy gains when side-channel attacks are a concern, while only implying limited (code size or area) overheads otherwise.

In view of the wide variety of security targets that the protection against leakage entails (e.g., leakage in encryption only or in encryption and decryption, with leakage-resilience or resistance, with or without beyond-birthday guarantees and multi-user security – see [GPPS19]), we expect that the tools developed here could serve to analyze a variety of other existing AE schemes (e.g., other round-2 candidates of the NIST standardization effort) while also motivating the design of new ones. Turning the abstract analyses in this work into concrete guidelines for hardware implementers and evaluating the overheads of strongly protected AE schemes is another important track for future investigations.

Acknowledgments

We thank the reviewers for comments that help reshape our work, and Itamar Levi for discussion w.r.t. side-channel state recovery attack against duplex. Thomas Peters is a postdoctoral researcher and François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC project SWORD (724725), and the European Union and Walloon Region FEDER USERMedia project 501907-379156. Chun Guo was partly supported by the Program of Qilu Young Scholars of Shandong University.

References


Towards Low-Energy Leakage-Resistant AE from the Duplex Sponge


