IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 1, pp. 6–42. DOI:10.13154/tosc.v2020.i1.6-42

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

Chun Guo[1,2,3], Olivier Pereira[3],
Thomas Peters[3] and François-Xavier Standaert[3]

[1] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China, chun.guo@sdu.edu.cn
[2] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
[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 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 TETSponge is an appealing option for the implementation of low-energy AE 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.

Received: 2019-09-01, Revised: 2019-11-23, Accepted: 2020-01-23, Published: 2020-05-07

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 (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+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". TEDT also ensures security against Chosen Ciphertext Adversaries with misuse-resilience (meaning security for messages encrypted with *fresh, non-repeating* nonces,[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, TEDT 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 [YSPY10] 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 $\approx 2^{c/2}$ 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+19] to boost multi-user

[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 $\mathsf{CCAmL1}$ confidentiality up to $\approx 2^{n/2}$ queries and $\approx 2^{n_p}$ users, which corresponds to CCA security with misuse-resilience, in the presence of encryption leakages; (ii) multi-user $\mathsf{CIML2}$ authenticity up to $2^n/n^2$ queries and $\approx 2^{n_p}$ users.

For practical usability, we carefully designed $\mathsf{TETSponge}$ to achieve high black-box CCA security as well, i.e., up to $2^n/n^2$ queries and $\approx 2^{n_p}$ 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 AEs 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 $\mathsf{TETSponge}$.

The security analyses of $\mathsf{TETSponge}$ are complicated and divided into Sections 6 (integrity multi-user $\mathsf{CIML2}$) and 7 (multi-user $\mathsf{CCAmL1}$). 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 $\mathsf{lsb}_a(x)/\mathsf{msb}_a(x)$ the least/most significant a bits of x. $[\mathsf{num}]_a$ is the binary encoding of the integer num using a representation of a bits.

We denote by a $(q_1, \ldots, q_\omega, t)$-bounded adversary a probabilistic algorithm that has access to ω 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 Algo is denoted LAlgo. It runs both Algo and a *leakage function* $\mathsf{L}_{\mathsf{Algo}}$ which captures the additional information given by the implementation of Algo during its execution. LAlgo simply returns the outputs of both Algo and $\mathsf{L}_{\mathsf{Algo}}$ which all take the same input.

A random keyless permutation π, 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 Tweakable Block Cipher (TBC) with key space $\{0,1\}^\kappa$, tweak space $\{0,1\}^t$, and domain $\{0,1\}^n$, also denoted (κ, t, n)-TBC, is a mapping $\widetilde{\mathsf{E}} : \{0,1\}^\kappa \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ such that for any key $K \in \{0,1\}^\kappa$ and any tweak $T \in \{0,1\}^t$, $X \mapsto \widetilde{\mathsf{E}}(K,T,X)$ is a permutation of $\{0,1\}^n$. In this paper we only focus on (n,n,n)-TBC. An ideal TBC $\widetilde{\mathsf{IC}} : \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, 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].

$\widetilde{\mathsf{IC}}_K^T$ 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 $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ such that:

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

The message size ℓ_m uniquely determines the ciphertext size $\ell_c = \ell_m + \mathsf{oh}$, where the constant oh is the stretch. Given a key $k \leftarrow \mathcal{K}$, $\mathsf{Enc}_k(N, A, M) := \mathsf{Enc}(k, N, A, M)$ and $\mathsf{Dec}_k(N, A, C) := \mathsf{Dec}(k, N, A, C)$ 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 $\mathsf{L}_{\mathsf{Enc}}$ and a decryption leakage function $\mathsf{L}_{\mathsf{Dec}}$ 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* ($\mathsf{muCIML2}$) defined in [BGP$^+$19], which was built upon the single-user version $\mathsf{CIML2}$ 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) $\mathsf{INT\text{-}CTXT}$ security with leakages.

Definition 2 ($\mathsf{muCIML2}$ advantage)**.** Given the implementation of a nonce-based authenticated encryption $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ with leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$, the multi-user ciphertext integrity advantage with misuse-*resistance* and leakage of an adversary $\mathcal{A}$ against AEAD with u users is

$$\mathbf{Adv}^{\mathsf{muCIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L},u} := \Pr\left[\mathcal{A}^{\mathsf{LEnc}_{\mathbf{K}},\mathsf{LDec}_{\mathbf{K}},\pi,\pi^{-1},\widetilde{\mathsf{IC}},\widetilde{\mathsf{IC}}^{-1}} \text{ forges}\right] ,$$

where the probability is taken over the u user keys $\mathbf{K} = (K_1, \ldots, K_u)$, $K_i \xleftarrow{\$} \mathcal{K}$, over $\mathcal{A}$'s random tape and the ideal oracles π and $\widetilde{\mathsf{IC}}$ and where:

- $\mathsf{LEnc}_{\mathbf{K}}(i, N, A, M)$: if $1 \leq i \leq u$, outputs the cipher $\mathsf{Enc}_{K_i}(N, A, M)$ and the leakage trace $\mathsf{L}_{\mathsf{Enc}}(K_i, N, A, M)$;
- $\mathsf{LDec}_{\mathbf{K}}(i, N, A, C)$: if $1 \leq i \leq u$, outputs $\big(\mathsf{Dec}_{K_i}(N, A, C), \mathsf{L}_{\mathsf{Dec}}(K_i, N, A, C)\big)$;
- The event "$\mathcal{A}$ forges" means that any of $\mathcal{A}$'s query to $\mathsf{LDec}_{\mathbf{K}}(i, N, A, C)$ returns $M \neq \perp$, while C was not resulted from an earlier encryption query to $\mathsf{LEnc}_{\mathbf{K}}(i, N, A, M)$.

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 $\mathcal{A}$ 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., $\mathcal{A}$ 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 $\mathsf{muCCAmL1}$, while the notion capturing the presence of 2 leakages sources (i.e., including decryption) is $\mathsf{muCCAmL2}$. We only recall the former definition.

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

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

where the security game $\mathsf{PrivK}^{\mathsf{muCCAmL1},d}_{\mathcal{A},\mathsf{AEAD},\mathsf{L},u}$ 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 π are non-invertible. This assumption was previously used by Yu et al. [YSPY10] 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

$\mathsf{PrivK}^{\mathsf{muCCAmL1},d}_{\mathcal{A},\mathsf{AEAD},\mathsf{L},u}$ is the output of the following experiment:

Initialization: generates u secret keys $K_1, \ldots, K_u \xleftarrow{\$} \mathcal{K}$ and sets $\mathcal{E}_{ch}, \mathcal{E}_1, \ldots, \mathcal{E}_u \leftarrow \emptyset$.

Leaking encryption queries: $\mathcal{A}$ gets adaptive access to $\mathsf{LEnc}(\cdot,\cdot,\cdot,\cdot)$,
$\mathsf{LEnc}(i, N, A, M)$ outputs $\perp$ if $(i, N, *, *) \in \mathcal{E}_{ch}$, else computes $C \leftarrow \mathsf{Enc}_{K_i}(N, A, M)$ and $\mathsf{leak}_{\mathsf{enc}} \leftarrow \mathsf{L}_{\mathsf{Enc}}(K_i, N, A, M)$, updates $\mathcal{E}_i \leftarrow \mathcal{E}_i \cup \{N\}$ and returns $(C, \mathsf{leak}_{\mathsf{enc}})$.

Decryption queries: $\mathcal{A}$ gets adaptive access to $\mathsf{Dec}(\cdot,\cdot,\cdot,\cdot)$,
$\mathsf{Dec}(i, N, A, C)$ outputs $\perp$ if $(i, N, A, C) \in \mathcal{E}_{ch}$, else returns $\mathsf{Dec}_{K_i}(N, A, C)$;

Challenge queries: on possibly many occasions $\mathcal{A}$ submits $(i, N_{ch}, A_{ch}, M^0, M^1)$,
If $|M^0| \neq |M^1|$ or $N_{ch} \in \mathcal{E}_i$ or $(i, N_{ch}, *, *) \in \mathcal{E}_{ch}$, returns $\perp$; Else computes $C^d \leftarrow \mathsf{Enc}_{K_i}(N_{ch}, A_{ch}, M^d)$ and $\mathsf{leak}^d_{\mathsf{enc}} \leftarrow \mathsf{L}_{\mathsf{Enc}}(K_i, N_{ch}, A_{ch}, M^d)$, updates $\mathcal{E}_{ch} \leftarrow \mathcal{E}_{ch} \cup \{(i, N_{ch}, A_{ch}, C^d)\}$ and finally returns $(C^d, \mathsf{leak}^d_{\mathsf{enc}})$;

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

Figure 1: The $\mathsf{PrivK}^{\mathsf{muCCAmL1},d}_{\mathcal{A},\mathsf{AEAD},\mathsf{L},u}$ game.

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

Formally, for $\mathsf{F} \in \{\pi, \oplus\}$, 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 π, we further split the leakage into an *input* and an *output* part, i.e., we write $(\mathsf{L}^{in}_{\pi}(S^{in}), \mathsf{L}^{out}_{\pi}(S^{out}))$ for the leakage due to evaluating $\pi(S^{in}) \rightarrow S^{out}$ or $\pi^{-1}(S^{out}) \rightarrow S^{in}$. This distinction between L^{in}_{π} and L^{out}_{π} 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* π. 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 π. 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 $\mathsf{DuStr}[\pi]$, that constitutes the basis of many duplex-based AEs (e.g., Ascon [DEMS19], and $\mathsf{TETSponge}$ that will be introduced). Formally, the cipher $\mathsf{DuStr}[\pi]$ 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 δ_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 $\mathsf{DuStr}[\pi]$, we follow [PSV15] and leverage the security model *(left-or-right) Eavesdropper security with Leakage* (EavL). It states that two distinct messages encrypted by $\mathsf{DuStr}[\pi]$ are indistinguishable. Formally, recall that we have $\mathsf{LDuStr}_B[\pi](IV, A, M) := (\mathsf{DuStr}_B[\pi](IV, A, M), \mathsf{L}_{\mathsf{DuStr}}(B, IV, A, M))$ according to our conventions, where the leakage function $\mathsf{L}_{\mathsf{DuStr}}$ includes all the leakages listed in Figure 2.

The duplex stream cipher $\mathsf{DuStr}_B[\pi](IV, A, M)$, $|B| = \kappa$:

(1) Computes $S'_0 \leftarrow IV \| B$, $S_1 \leftarrow \pi(S'_0)$. **The leakages** of this step are $\mathsf{L}^{in}_\pi(S'_0)$ and $\mathsf{L}^{out}_\pi(S_1)$;

(2) For $i = 1, \ldots, \nu$, $\nu = |A|/r$, computes $S'_i \leftarrow (A[i] \| 0^c) \oplus S_i$ and $S_{i+1} \leftarrow \pi(S'_i)$. **The leakages** are $\mathsf{L}^{in}_\pi(S'_i), \mathsf{L}^{out}_\pi(S_{i+1})$, $\mathsf{L}_\oplus(\mathsf{msb}_r(S_i), A[i])$;

(3) $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r \| \delta_1)$ for a fixed offset δ_1. **The leakages** are $\mathsf{L}_\oplus(\mathsf{lsb}_c(S_{\nu+1}), \delta_1)$;

(4) For $i = 1, \ldots, \ell$, $\ell = |M|/r$, computes $j \leftarrow i+\nu$, $C[i] \leftarrow \mathsf{msb}_r(S_j) \oplus M[i]$, $S'_j \leftarrow C[i] \| \mathsf{lsb}_c(S_j)$, $S_{j+1} \leftarrow \pi(S'_j)$. **Leakages** are $\mathsf{L}_\oplus(\mathsf{msb}_r(S_j), M[i])$ and $\mathsf{L}^{in}_\pi(S'_j), \mathsf{L}^{out}_\pi(S_{j+1})$;

(5) Returns $\mathbf{c} = C[1] \| \ldots \| C[\ell]$.

Figure 2: The duplex-based stream cipher DuStr and the involved leakages.

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

$$\mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(\mathcal{A}^\pi) := \Big| \Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B(IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B(IV, A, M^1)) \Rightarrow 1] \Big|. \quad (1)$$

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

$$\mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(q_\pi, t, \ell) := \max \left\{ \mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(\mathcal{A}) \right\}, \quad (2)$$

where the maximal is taken over all adversaries $\mathcal{A}$ that makes q_π queries to π, runs in time t, and observes the ciphertext and leakages from encrypting a message with ℓ 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 $\mathcal{D}$ that makes a bounded number of q queries to its oracles. The technique addresses the settings where $\mathcal{D}$'s aim is to distinguish between two worlds: a "real world" and an "ideal world". Assume wlog that $\mathcal{D}$ is deterministic. The execution of $\mathcal{D}$ defines a *transcript* that includes the sequence of queries and answers received from its oracles; $\mathcal{D}$'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 $\mathcal{D}$'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 $\mathcal{D}$. Let $\mathcal{T} = \mathcal{T}_{good} \cup \mathcal{T}_{bad}$ be a partition of the set of attainable transcripts $\mathcal{T}$. Assume that there exists ε_1 such that for any $\tau \in \mathcal{T}_{good}$, one has*

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

and that there exists ε_2 such that $\Pr[T_{id} \in \mathcal{T}_{bad}] \leq \varepsilon_2$. *Then* $\mathbf{Adv}(\mathcal{D}) \leq \varepsilon_1 + \varepsilon_2$.

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 DuStr[π]

In this section we investigate the leakage confidentiality of the duplex-based stream cipher DuStr[π]. 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: $Y_i \leftarrow \mathsf{msb}_r(S_i)$; (2) modifying the state with a b-bit offset Δ: $S_i' \leftarrow S_i \oplus \Delta$; (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 $\mathsf{msb}_r(S_i')$ is a non-secret ciphertext block, e.g., see Fig. 6 (on the other hand, the piece $\mathsf{lsb}_c(\Delta)$ captures the possible XOR of domain separation bits). Therefore, the least significant bits $\mathsf{lsb}_c(S_i')$ are the critical secret, and the subsequent actions are compromised as long as $\mathsf{lsb}_c(S_i')$ 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 Δ, and (b) the "previous" π-call $\pi(\star) \rightarrow S_i$ 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 $\mathsf{lsb}_c(S_i')$, as we may also consider initializing the duplex with a key of $\kappa \neq c$ bits. Therefore, in our assumption we consider ω bits $\mathsf{lsb}_\omega(S_i')$ for generality, and assume that the side-channel adversary cannot predict the value of $\mathsf{lsb}_\omega(S_i')$ 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 $\mathsf{lsb}_\omega(S_i')$).

Definition. Formally, we define

$$\mathbf{Adv}^{\mathsf{Inv}[\omega]}(\mathcal{A}) := \Pr\big[s_{ch} \xleftarrow{\$} \{0,1\}^\omega, \mathcal{G} \leftarrow \mathcal{A}^\pi(\mathsf{leak}) : s_{ch} \in \mathcal{G}\big], \tag{3}$$

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

$$\mathsf{leak} = \big[\mathsf{L}_\pi^{out}(y^{pre}\|s_{ch}), \mathsf{L}_\oplus(\delta, s_{ch}), \mathsf{L}_\pi^{in}(y^{in}\|(\delta \oplus s_{ch}))\big]. \tag{4}$$

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

In such an invertibility game, the power of $\mathcal{A}$ is quantified along three dimensions, i.e., the number q_π of its queries to π, 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 $\mathbf{Adv}^{\mathsf{Inv}[\omega]}(\mathcal{A})$). To simplify, we further define

$$\mathbf{Adv}^{\mathsf{Inv}[\omega]}(q_\pi, t, N_G) := \max_{(q_\pi,t,N_G)\text{-}\mathcal{A}} \Big\{\mathbf{Adv}^{\mathsf{Inv}[\omega]}(\mathcal{A})\Big\}, \tag{5}$$

When $\omega = c$, the assumption captures the secrecy of the "capacity part", which is in line with the intuition proposed for ISAP [DEM+17].

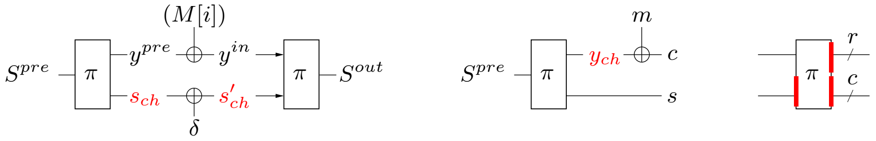

Figure 3: (Left) Illustrating the $\mathsf{Inv}[\omega]$ assumption. The value in red (i.e., s_{ch}) is the critical secret. The involved values are as defined with Eq. (3), while the values S^{pre} and S^{out} are only mentioned in the subsequent security tester. (Middle) The "basic" message manipulating operation. The value y_{ch} in red is the secret. (Right) A summary: which values should be somehow ensured secret (i.e., the lsb_ω 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 $\mathbf{Adv}^{\mathsf{Inv}[\omega]}$ can be measured for a specific implementation on a specific device by running the best known side-channel key/secret recovery adversary $\mathcal{A}$ 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.

```
Tester for non-invertibility $\mathbf{Adv}^{\mathsf{Inv}[\omega]}(q_\pi, t, N_G)$
Let the adversary $\mathcal{A}$ serve $b-\omega$ bit values $y^{pre}$ and $y^{in}$ and $\omega$-bit offset $\delta$
Pick the secret: $s_{ch} \xleftarrow{\$} \{0,1\}^\omega$
Compute $S^{pre} \leftarrow \pi^{-1}(y^{pre}\|s_{ch})$, $s'_{ch} \leftarrow s_{ch} \oplus \delta$, and $S^{out} \leftarrow \pi^{-1}(y^{in}\|s'_{ch})$
Serve $\mathcal{A}$ with the leakages resulting from step 4. Following our notations, this gives $\mathcal{A}$ the
    leakages $\mathsf{L}_\pi^{out}(y^{pre}\|s_{ch})$, $\mathsf{L}_\oplus(s_{ch}, \delta)$, and $\mathsf{L}_\pi^{in}(y^{in}\|s'_{ch})$.
Let $\mathcal{A}$ output $N_G$ guesses $\mathcal{G} = \{s_1, \ldots, s_{N_G}\}$, and $\mathcal{A}$ wins as long as $s_{ch} \in \mathcal{G}$
```

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

$$\mathbf{Adv}^{\mathsf{LORL}}(\mathcal{A}) := \Bigg| \Pr\Big[y_{ch} \xleftarrow{\$} \{0,1\}^r, c^0 \leftarrow y_{ch} \oplus m^0 : \mathcal{A}^\pi(c^0, \mathsf{leak}_0) \Rightarrow 1\Big] - \Pr\Big[y_{ch} \xleftarrow{\$} \{0,1\}^r, c^1 \leftarrow y_{ch} \oplus m^1 : \mathcal{A}^\pi(c^1, \mathsf{leak}_1) \Rightarrow 1\Big]\Bigg| , \quad (6)$$

where leak_d again depends on a c-bit value s chosen by $\mathcal{A}$:

$$\mathsf{leak}_d = \big[\mathsf{L}_\pi^{out}(y_{ch}\|s), \mathsf{L}_\oplus(y_{ch}, m^d)\big]. \quad (7)$$

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 $\mathsf{L}_\pi^{out}(y_{ch}\|s)$. Then, y_{ch} is used to mask the message block, and thus $\mathsf{L}_{\oplus}(y_{ch}, m^d)$ comes. We also define

$$\mathbf{Adv}^{\mathsf{LORL}}(q_\pi, t) := \max_{\mathcal{A}} \Big\{ \mathbf{Adv}^{\mathsf{LORL}}(\mathcal{A}) \Big\}. \tag{8}$$

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

```
1: Tester for LORL $\mathbf{Adv}^{\mathsf{LORL}}(q_\pi, t)$
2: Let the challenging adversary $\mathcal{A}$ serve $s$ and $(m^0, m^1)$
3: Pick the secret: $y_{ch} \xleftarrow{\$} \{0,1\}^r$, $d \xleftarrow{\$} \{0,1\}$
4: Compute $c \leftarrow y_{ch} \oplus m^d$
5: Generate the permutation leakages: repeating $S_{pre} \leftarrow \pi^{-1}(y_{ch} \| s)$ 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 $(c, [\mathsf{L}_\pi^{out}(y_{ch} \| s), \mathsf{L}_{\oplus}(y_{ch}, m^d)])$
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 $\mathbf{Adv}^{\mathsf{LORL}}(q_\pi, t)$ 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 $\mathsf{DuStr}[\pi]$. Formally, we have

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

$$\begin{aligned}\mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(q_\pi, t, \ell) \leq & \frac{(\ell+2)^2}{2^c} + \ell \cdot \mathbf{Adv}^{\mathsf{LORL}}(q_\pi, t^*) + 2\mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, t^*, 2q_\pi) \\ & + 2(\ell+1) \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi),\end{aligned} \tag{9}$$

where $t^* = O(t + \ell t_l)$*, and* t_l *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 $\ell \cdot \mathbf{Adv}^{\mathsf{LORL}}(q_\pi, t^*)$ reflects the reduction to the "minimal message manipulation" (as discussed earlier), and the factor ℓ reflects a (seemingly) unavoidable leakage security loss. The terms $2\mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, t^*, 2q_\pi) + 2(\ell+1) \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi)$ capture the hardness of side-channel secret recovery, and they are roughly of some birthday type

$$O\Big(\frac{q_\pi + \ell + t}{\mu_\kappa \cdot 2^\kappa}\Big) + O\Big(\ell \cdot \frac{q_\pi + \ell + t}{\mu_c \cdot 2^c}\Big),$$

for some parameters μ_κ and μ_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) μ_κ and μ_c can be made very small [DEM+17], so we "restore" the classical $2^{c/2}$ 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 $\mathsf{IdealS}(IV, A, M)$:

(1) Samples $B \xleftarrow{\$} \{0,1\}^{\kappa}$;

(2) Computes $S'_0 \leftarrow IV\|B$ and samples $S_1 \xleftarrow{\$} \{0,1\}^b$. **The leakages** of this step are $\mathsf{L}_\pi^{in}(S'_0)$ and $\mathsf{L}_\pi^{out}(S_1)$;

(3) For $i = 1, \dots, \nu$, $\nu = |A|/r$, computes $S'_i \leftarrow (A[i]\|0^c) \oplus S_i$ and samples $S_{i+1} \xleftarrow{\$} \{0,1\}^b$. **The leakages** are $\mathsf{L}_\pi^{in}(S'_i), \mathsf{L}_\pi^{out}(S_{i+1}), \mathsf{L}_\oplus(\mathsf{msb}_r(S_i), A[i])$;

(4) $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r\|\delta_1)$. **The leakages** are $\mathsf{L}_\oplus(\mathsf{lsb}_c(S_{\nu+1}), \delta_1)$;

(5) For $i = 1, \dots, \ell$, $\ell = |M|/r$, computes $j \leftarrow i + \nu$, $C[i] \leftarrow \mathsf{msb}_r(S_j) \oplus M[i]$, $S'_j \leftarrow C[i]\|\mathsf{lsb}_c(S_j)$, and samples $S_{j+1} \xleftarrow{\$} \{0,1\}^b$. **The leakages** are $\mathsf{L}_\oplus(\mathsf{msb}_r(S_j), M[i])$ and $\mathsf{L}_\pi^{in}(S'_j), \mathsf{L}_\pi^{out}(S_{j+1})$;

(6) Returns $\mathbf{c} = C[1]\|\dots\|C[\ell]$.

Figure 4: The ideal stream cipher 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 IdealS in Figure 4 for "relay". Briefly, IdealS is obtained via replacing all the internal actions $S_{i+1} \leftarrow \pi(S'_i)$ in $\mathsf{DuStr}[\pi]$ by sampling $S_{i+1} \xleftarrow{\$} \{0,1\}^b$. By our conventions, we have $\mathsf{LIdealS}(IV, A, M) := (\mathsf{IdealS}(IV, A, M), \mathsf{L}_{\mathsf{IdealS}}(IV, A, M))$, where $\mathsf{L}_{\mathsf{IdealS}}$ includes all the leakages in Figure 4.

Regarding IdealS, we have the following two lemmas, the proofs of which are deferred to subsections 3.3.1 and 3.3.2 respectively, for cleanness.

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

$$\begin{aligned}&\big|\Pr[\mathcal{D}^\pi(\mathsf{LDuStr}_B[\pi](\mathbf{tpl})) \Rightarrow 1] - \Pr[\mathcal{D}^\pi(\mathsf{LIdealS}(\mathbf{tpl})) \Rightarrow 1]\big| \\ \le&\frac{(\ell+2)^2}{2^{c+1}} + \mathbf{Adv}^{\mathsf{Inv}[\kappa]}\big(q_\pi, t^*, 2q_\pi\big) + (\ell+1)\cdot\mathbf{Adv}^{\mathsf{Inv}[c]}\big(q_\pi, t^*, 2q_\pi\big),\end{aligned} \tag{10}$$

where $t^* = O(t + \ell t_l)$, and t_l is the total time needed for evaluating L_π^{in}, L_π^{out}, $\mathsf{L}_\oplus$, and the xor of two r-bit values.

Lemma 3. *For every pair of ℓ-block messages M^0 and M^1 and (q_π, t)-bounded adversary $\mathcal{A}^\pi$, it holds*

$$\begin{aligned}&\big|\Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^1)) \Rightarrow 1]\big| \\ \le\ & \ell\cdot\mathbf{Adv}^{\mathsf{LORL}}(q_\pi, O(t+\ell t_l)),\end{aligned}$$

where t_l is as defined in Lemma 2.

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

$$\begin{aligned}&\big|\Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B[\pi](IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B[\pi](IV, A, M^1)) \Rightarrow 1]\big| \\ &\le \underbrace{\big|\Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^1)) \Rightarrow 1]\big|}_{\le \ell\cdot\mathbf{Adv}^{\mathsf{LORL}}(q_\pi, O(t+\ell t_l)) \text{ (by Lemma 3)}} \\ &\quad + \sum_{b=0,1}\big|\Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B[\pi](IV, A, M^b)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^b)) \Rightarrow 1]\big|.\end{aligned}$$

For each b, Lemma 2 indicates

$$\begin{aligned}&\big|\Pr[\mathcal{A}^\pi(\mathsf{LDuStr}_B[\pi](IV, A, M^b)) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, A, M^b)) \Rightarrow 1]\big| \\ \le&\mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, O(t+\ell t_l), 2q_\pi) + (\ell+1)\cdot\mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, O(t+\ell t_l), 2q_\pi) + \frac{(\ell+2)^2}{2^{c+1}}.\end{aligned}$$

Therefore,

$$\begin{aligned}&\big|\Pr[\mathcal{A}^{\pi}(\mathsf{LDuStr}_B[\pi](IV, A, M^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\pi}(\mathsf{LDuStr}_B[\pi](IV, A, M^1)) \Rightarrow 1]\big| \\ \leq &\frac{(\ell+2)^2}{2^c} + \ell \cdot \mathbf{Adv}^{\mathsf{LORL}}(q_\pi, O(t+\ell t_l)) + 2\mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, O(t+\ell t_l), 2q_\pi) \\ &+ 2(\ell+1)\cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, O(t+\ell t_l), 2q_\pi),\end{aligned}$$

as claimed. □

3.3.1 Proof of Lemma 2

The crux is to bound certain "bad events" during the execution of 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 $\mathsf{G}_1(\mathcal{D}, \mathsf{LDuStr}_B[\pi], \pi)$ and $\mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)$ the games capturing the interactions between $\mathcal{D}$ and the real $(\mathsf{LDuStr}_B[\pi], \pi)$ and the ideal $(\mathsf{LIdealS}, \pi)$ 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 τ_{le} and τ_π: the former includes the ciphertext as well as the leakages (its concrete representation won't be needed in this proof), while the latter $\tau_\pi = \big((S_1^{in}, S_1^{out}), \ldots, (S_{q_\pi}^{in}, S_{q_\pi}^{out})\big)$ includes the adversarial permutation queries and responses, and indicates the i-th query is either forward $\pi(S_i^{in}) \to S_i^{out}$ or backward $\pi^{-1}(S_i^{out}) \to S_i^{in}$. Besides, at the end of the interaction, we reveal the involved internal state values $\mathbf{S} = (S_0', S_1, S_1', \ldots)$ 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 $\mathbf{r}$, the random coins of the distinguisher $\mathcal{D}$ & the involved leakage functions. Yet, it can be seen that during two games $\mathsf{G}_1(\mathcal{D}, \mathsf{LDuStr}_B[\pi_1], \pi_1)$ and $\mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi_2)$, if the following conditions are fulfilled, then the queries and responses of $\mathcal{D}$ are the same, and thus $\mathcal{D}$ outputs the same:

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

It is because all the internal actions in $\mathsf{G}_1(\mathcal{D}, \mathsf{LDuStr}_B[\pi_1], \pi_1)$ and $\mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi_2)$ 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., $\tau = (\tau_\pi, \mathbf{S}, \mathbf{r})$. Note that τ_{le} is absent in τ, as it can be recovered from $\mathbf{r}$ and $\mathbf{S}$.

With respect to some fixed distinguisher $\mathcal{D}$, an extended transcript $\tau = (\tau_\pi, \mathbf{S}, \mathbf{r})$ is *attainable* if there exists randomness $(\mathbf{r}, \pi)$ such that using $\mathbf{r}$, the ideal execution of $\mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)$ yields $(\tau_\pi, \mathbf{S})$. We denote $\mathcal{T}$ the set of attainable transcripts.

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

$$\begin{aligned}\Pr[T_{id} = \tau] &= \Pr[\pi \vdash \tau_\pi] \cdot \Pr[\mathbf{r}] \cdot \Pr[\mathbf{S} \text{ is sampled in the ideal world}] \\ &= \Pr[\pi \vdash \tau_\pi] \cdot \Pr[\mathbf{r}] \cdot \frac{1}{2^\kappa} \cdot \frac{1}{2^{(\ell+1)b}}, \qquad (11)\end{aligned}$$

as $\Pr[S'_0] = \Pr[B] = \frac{1}{2^\kappa}$, while $\Pr[S_i] = \frac{1}{2^b}$ for $i = 1, \ldots, \ell + 1$. In the subsequent two paragraphs, we will define the partition $\mathcal{T} = \mathcal{T}_{good} \cup \mathcal{T}_{bad}$ and analyze $\Pr[T_{id} = \tau]$ for $\tau \in \mathcal{T}_{good}$ respectively.

Bad extended transcripts and $\Pr[T_{id} \in \mathcal{T}_{bad}]$. An attainable transcript $\tau = (\tau_\pi, \mathbf{S}, \mathbf{r})$ is bad, if any of following conditions is fulfilled:

- (B-1) *contradiction between state value*: there exists two distinct indices $i, j \in [0, \ldots, \ell]$ such that $S'_i = S'_j \wedge S_{i+1} \neq S_{j+1}$, or $S'_i \neq S'_j \wedge S_{i+1} = S_{j+1}$;
- (B-2) *exposure of secret state*: if any of the following is fulfilled:
  - there exists $i \in \{0, \ldots, \ell\}$ such that $(S'_i, \star) \in \tau_\pi$; or
  - there exists $i \in \{1, \ldots, \ell + 1\}$ such that $(\star, S_i) \in \tau_\pi$.

As long as the c-bit (uniformly distributed) capacity pieces of $S'_0, S_1, S_2, \ldots, S_{\ell+1}$ don't collide, the condition (B-1) cannot be fulfilled. Therefore,

$$\Pr[\text{(B-1)}] \leq \binom{\ell + 2}{2} \cdot \frac{1}{2^c} = \frac{(\ell + 2)^2}{2^{c+1}}. \tag{12}$$

To bound $\Pr[\text{(B-2)}]$, 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) $\mathsf{BadInner}$: that occurs when there exists an index $i \in \{1, \ldots, \ell\}$ such that $\big(\star, S_i\big) \in \tau_\pi$ or $\big(S'_i, \star\big) \in \tau_\pi$;
(2) $\mathsf{BadInit}$: that occurs when $(S'_0, \star) \in \tau_\pi$;
(3) $\mathsf{BadFinal}$: that occurs when $(\star, S_{\ell+1}) \in \tau_\pi$.

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

$$\mathbf{Adv}^{\mathsf{Inv}[c]}(\mathcal{A}^\pi) \leq \Pr_{\mathbf{r},\mathbf{S},\pi}[\mathsf{BadInner} \text{ in } \mathcal{D}^\pi(\mathsf{LIdealS}(IV, M))]. \tag{13}$$

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

(1) $\mathcal{A}^\pi$ guesses an index $i \xleftarrow{\$} [1, \ell]$, samples a key $B \xleftarrow{\$} \{0, 1\}^\kappa$, sets $S'_0 \leftarrow IV \| B$, and initializes a list leak with the leakages $\mathsf{L}^{in}_\pi(S'_0)$;
(2) For $j = 1, \ldots, i - 1$, $\mathcal{A}^\pi$ samples the new state $S_j \xleftarrow{\$} \{0, 1\}^b$, computes $C[j] \leftarrow \mathsf{msb}_r(S_j) \oplus M[j]$, and:
   - computes $S'_j \leftarrow C[j] \| (\mathsf{lsb}_c(S_j) \oplus \delta_1)$ and adds the leakages $\mathsf{L}_\oplus(\mathsf{lsb}_c(S_j), \delta_1)$ to leak when $j = 1$, and
   - computes $S'_j \leftarrow C[j] \| \mathsf{lsb}_c(S_j)$ otherwise.

   $\mathcal{A}^\pi$ further adds the leakages $\mathsf{L}^{out}_\pi(S_j), \mathsf{L}^{in}_\pi(S'_j)$, and $\mathsf{L}_\oplus(\mathsf{msb}_r(S_j), M[j])$ to leak;
(3) $\mathcal{A}^\pi$ samples $y \xleftarrow{\$} \{0, 1\}^r$ and computes $C[i] \leftarrow y \oplus M[i]$. $\mathcal{A}^\pi$ then submits $y, C[i]$, and $\delta = \delta_1$ (when $i = 1$) or $\delta = 0$ (when $i > 1$) to its $\mathsf{Inv}[c]$ challenger and obtains the leakages

$$\mathsf{leak}_{ch} = \big[\mathsf{L}^{out}_\pi(y \| s_{ch}), \mathsf{L}_\oplus(s_{ch}, \delta), \mathsf{L}^{in}_\pi\big(C[i] \| (s_{ch} \oplus \delta)\big)\big]$$

   for the challenge secret $s_{ch} \in \{0, 1\}^c$. $\mathcal{A}^\pi$ then adds the leakages $\mathsf{L}_\oplus(y, M[i])$ and leak_{ch} to leak. This means $y \| s_{ch}$ is taken as S_i—though $\mathcal{A}^\pi$ doesn't know its value.
(4) $\mathcal{A}^\pi$ then completes the remaining encrypting: for $j = i + 1, \ldots, \ell$, $\mathcal{A}^\pi$ samples $S_j \xleftarrow{\$} \{0, 1\}^b$, computes $C[j] \leftarrow \mathsf{msb}_r(S_j) \oplus M[j]$, $S'_j \leftarrow C[j] \| \mathsf{lsb}_c(S_j)$, and adds $\mathsf{L}^{out}_\pi(S_j), \mathsf{L}^{in}_\pi(S'_j)$, and $\mathsf{L}_\oplus(\mathsf{msb}_r(S_j), M[j])$ to leak.

(5) Finally, $\mathcal{A}^\pi$ samples $S_{\ell+1} \xleftarrow{\$} \{0,1\}^b$, adds $\mathsf{L}_\pi^{out}(S_{\ell+1})$ to the list leak, returns $(C[1]\|\ldots\|C[\ell], \mathsf{leak})$ to $\mathcal{D}$, and outputs $\mathcal{G} := \{\mathsf{lsb}_c(S^{in}), \mathsf{lsb}_c(S^{out}) : (S^{in}, S^{out}) \in \tau_\pi\}$.

The strategy of $\mathcal{A}^\pi$ is to make a uniform guess on the position of the first inner secret value that appears in τ_π, 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, $\mathcal{A}^\pi$ simulates $\mathsf{IdealS}(IV, M)$ and provides the leakages to $\mathcal{D}$, except for the i index, for which the leakages are replaced by those obtained from an Inv challenger. Now if the guess on the index i is correct, then all the inputs sent to $\mathcal{D}$ are distributed exactly as those in a normal execution of G_2. Therefore, when $\mathcal{D}$ halts, if $\mathcal{D}$ made a query with s_{ch}, then outputting the aforementioned set $\mathcal{G}$ (based on τ_π) would break the Inv game. So we have

$$\Pr[s_{ch} \in \mathcal{G} \mid \mathsf{BadInner} \text{ in } \mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)] \geq \frac{1}{\ell}.$$

Now, we observe that

$$\Pr[s_{ch} \in \mathcal{G} \mid \mathsf{BadInner} \text{ in } \mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)] \leq \frac{\Pr[s_{ch} \in \mathcal{G}]}{\Pr[\mathsf{BadInner} \text{ in } \mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)]}.$$

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

$$\begin{aligned}\Pr[\mathsf{BadInner} \text{ in } \mathsf{G}_2(\mathcal{D}, \mathsf{LIdealS}, \pi)] &\leq \ell \cdot \Pr[s_{ch} \in \mathcal{G}] \\ &\leq \ell \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(\mathcal{A}) \quad \text{(Eq. (3))} \\ &\leq \ell \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi). \quad \text{(Eq. (5))}\end{aligned}$$

These finish the analysis of $\mathsf{BadInner}$. For the events $\mathsf{BadInit}$ and $\mathsf{BadFinal}$, similar arguments could establish

$$\Pr[\mathsf{BadInit}] \leq \mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, t^*, 2q_\pi), \quad \Pr[\mathsf{BadFinal}] \leq \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi).$$

The three terms plus Eq. (12) yield

$$\Pr[T_{id} \in \mathcal{T}_{bad}] \leq (\ell+1) \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi) + \mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, t^*, 2q_\pi) + \frac{(\ell+2)^2}{2^{c+1}}. \quad (14)$$

Summarizing. For any good transcript $\tau = (\tau_\pi, \mathbf{S}, \mathbf{r}) \in \mathcal{T}_{good}$, we have

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

Conditioned on $\neg$(B-1) and $\neg$(B-2), it can be seen that, for all $i \in \{0, \ldots, \ell\}$, it holds $\Pr[\pi(S_i') = S_{i+1} \mid \pi \vdash \tau_\pi \wedge \forall j < i : \pi(S_j') = S_{j+1}] \geq \frac{1}{2^b}$. This means $\Pr[\forall i \in \{0, \ldots, \ell\} : \pi(S_i') = S_{i+1} \mid \pi \vdash \tau_\pi] \geq \frac{1}{2^{(\ell+1)b}}$. Therefore, $\Pr[T_{re} = \tau] \geq \Pr[T_{id} = \tau]$ by Eq. (11), and thus $\Pr[T_{id} \in \mathcal{T}_{bad}]$ constitutes the final bound, i.e.,

$$\begin{aligned}&\left|\Pr[\mathcal{D}^\pi(\mathsf{LDuStr}_B[\pi](IV, M)) \Rightarrow 1] - \Pr[\mathcal{D}^\pi(\mathsf{LIdealS}(IV, M)) \Rightarrow 1]\right| \\ \leq\ &\frac{(\ell+2)^2}{2^{c+1}} + (\ell+1) \cdot \mathbf{Adv}^{\mathsf{Inv}[c]}(q_\pi, t^*, 2q_\pi) + \mathbf{Adv}^{\mathsf{Inv}[\kappa]}(q_\pi, t^*, 2q_\pi).\end{aligned}$$

3.3.2 Proof of Lemma 3

Again we assume $|A| = 0$ for simplicity. Let $M^0 = M^0[1]\|\dots\|M^0[\ell]$ and $M^1 = M^1[1]\|\dots\|M^1[\ell]$. We start by building a sequence of $\ell+1$ messages $M_{h,0},\dots,M_{h,\ell}$ starting from M^0 and modifying its blocks one by one, to obtain M^1. That is, $M_{h,i} := M^1[1]\|\dots\|M^1[i]\|M^0[i+1]\|\dots\|M^0[\ell]$. For any i, assuming a (q_π, t)-bounded adversary $\mathcal{A}^\pi$ against $\mathsf{LIdealS}(IV, M_{h,i-1})$ and $\mathsf{LIdealS}(IV, M_{h,i})$, we build a $(q_\pi, O(t+\ell t_l))$-bounded adversary $\mathcal{A}_2^\pi$ against the distribution defined in Eq. (6). In detail, $\mathcal{A}_2^\pi$ proceeds in four steps:

(1) $\mathcal{A}_2^\pi$ samples $B \xleftarrow{\$} \{0,1\}^\kappa$, initializes an empty list leak, and sets $S_0' \leftarrow IV\|B$;

(2) for $j = 1,\dots,i-1$, $\mathcal{A}_2^\pi$ samples $S_j \xleftarrow{\$} \{0,1\}^b$, computes $C[j] \leftarrow \mathsf{msb}_r(S_j) \oplus M^1[j]$, $S_j' \leftarrow C[j]\|\mathsf{lsb}_c(S_j)$ ($S_j' \leftarrow C[j]\|(\mathsf{lsb}_c(S_j) \oplus \delta_1)$ when $j = 1$), and adds $\mathsf{L}_\pi^{in}(S_{j-1}')$, $\mathsf{L}_\pi^{out}(S_j)$, and $\mathsf{L}_\oplus(\mathsf{msb}_r(S_j), M^1[j])$ (and $\mathsf{L}_\oplus(\mathsf{lsb}_c(S_j), \delta_1)$) to leak;

(3) $\mathcal{A}_2^\pi$ samples $s \xleftarrow{\$} \{0,1\}^c$ and submits s to the LORL challenger. Assume that the outputs are (c^b, leak_b) with

$$\mathsf{leak}_b = \big[\mathsf{L}_\pi^{out}(y_{ch}\|s), \mathsf{L}_\oplus(y_{ch}, m^b)\big].$$

$\mathcal{A}_2^\pi$ then adds $\mathsf{L}_\pi^{in}((M^1[i-1]\|0^c) \oplus S_{i-1})$, $\mathsf{L}_\pi^{out}(y_{ch}\|s)$, and $\mathsf{L}_\oplus(y_{ch}, m^b)$ to leak;

(4) $\mathcal{A}_2^\pi$ starts from $c^b\|C$ to emulate the remaining actions of $\mathsf{LIdealS}$ encrypting the tail $M^0[i+1]\|\dots\|M^0[\ell]$ to obtain $C[i+1]\|\dots\|C[\ell]$. Eventually, $\mathcal{A}_2^\pi$ serves the ciphertext $C[1]\|\dots\|C[i-1]\|c^b\|C[i+1]\|\dots\|C[\ell]$ (and $\mathsf{lsb}_c(S_{\nu+\ell+1})$, when $= 1$) as well as all the generated simulated leakages to $\mathcal{A}^\pi$, and outputs whatever $\mathcal{A}^\pi$ outputs.

It can be seen depending on whether the input tuple received by $\mathcal{A}_2^\pi$ captures the LORL challenger encrypting $M^0[i]$ or $M^1[i]$, the inputs to $\mathcal{A}^\pi$ capture $\mathsf{LIdealS}$ encrypting $M_{h,i-1}$ or $M_{h,i}$. Moreover, $\mathcal{A}_2^\pi$ is $(q_\pi, O(t+\ell t_l))$-bounded if $\mathcal{A}^\pi$ is (q_π, t)-bounded. Therefore,

$$\begin{aligned} &\big|\Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, M_{h,i-1})) \Rightarrow 1] - \Pr[\mathcal{A}^\pi(\mathsf{LIdealS}(IV, M_{h,i})) \Rightarrow 1]\big| \\ \leq\ & \mathbf{Adv}^{\mathsf{LORL}}(q_\pi, O(t+\ell t_l)) \end{aligned}$$

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 cannnot 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 $\mathsf{BadFinal}$ in the proof of Lemma 2.

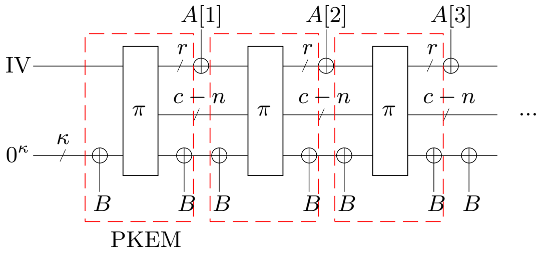

Figure 5: The PKEM-based representation of a keyed sponge with κ-bit key/initial seed B. The “partial-key” Even-Mansour cipher is inside the dashed rectangles. Internal actions of XORing B cancel. The construction turns basically the same as the keyed sponge.

4 Description of TETSponge

We design TETSponge 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. $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{E}}]$ is built on an (n, n, n)-TBC $\widetilde{\mathsf{E}}$ and a b-bit permutation π 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 $n_p \le r$, $n_N + n_p + n \le r + c$, and $2n \le r + c + 1$. 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 $\mathsf{muCIML2}$ (and black-box CCA) security up to $2^n/n^2$ queries.

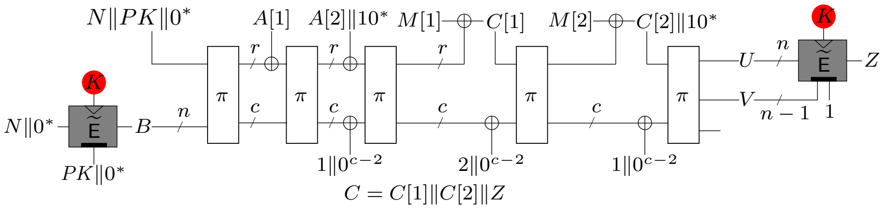

Figure 6: $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{E}}]_{K,PK}$ AEAD for $\nu = 2$ blocks of associated data and $\ell = 2$ message blocks. Dark squares indicate the "leak-free" TBC where the triangle denotes the key input and the small black rectangle denotes the tweak input. The value $1\|0^{c-2}$ 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 $\widetilde{\mathsf{E}}_K^{PK\|0^*}(N\|0^*)$. The seed B is then used as the key of the duplex to process A and $M = M[1]\|\ldots\|M[\ell]$ and produce $\mathbf{c} = C[1]\|\ldots\|C[\ell]$. 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 $Z = \widetilde{\mathsf{E}}_K^{V\|1}(U)$. The final ciphertext is $C[1]\|\ldots\|C[\ell]\|Z$.

The decryption. Upon decrypting (N, A, C), $C = C[1]\|\ldots\|C[\ell]\|Z$, the mode first recovers the initial seed B via $B = \widetilde{\mathsf{E}}_K^{PK\|0^*}(N\|0^*)$, and then runs the duplex on A and $C[1]\|\ldots\|C[\ell]$ to recover M and the $2n-1$ bit truncated state $U\|V$. Finally, it makes an inverse TBC call $U^* = (\widetilde{\mathsf{E}}_K^{V\|1})^{-1}(Z)$, 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 $\mathsf{CIML2}$).

The encryption and decryption are formally described in Figure 7. Note that when $\mathbf{c}$ 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 $\mathsf{TETSponge}$ in the subsequent two sections. Both of the two proofs will rely on an idealized $\mathsf{TETSponge}$ variant $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$, which internally makes calls to a secret ideal TBC $\widetilde{\mathsf{SIC}}$ instead of the public $\widetilde{\mathsf{IC}}$.[3] This means the key derivation and tag generation calls in $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$ will not be influenced by the adversarial queries (to $\widetilde{\mathsf{IC}}$).

We will consider a setting motivated by the unbounded leakage assumption: a distinguisher is interacting with either the real world $(\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}}, \pi, \widetilde{\mathsf{IC}})$ or the idealized world $(\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}, \pi, \widetilde{\mathsf{IC}})$, but all the intermediate values inside the mode $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}}/\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$ are completely given. Note that this means the adversary could observe inputs/outputs of the internal TBC-calls from the interaction with $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}}/\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$. 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 $\mathcal{D}$ cannot distinguish the two worlds up to $\approx 2^n/n^2$ queries. Formally,

Lemma 4. *Under the unbounded leakage assumption, it holds*

$$\begin{aligned}&\left|\Pr[\mathcal{D}^{\mathsf{TETSponge}[\pi,\widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}},\pi,\pi^{-1},\widetilde{\mathsf{IC}},\widetilde{\mathsf{IC}}^{-1}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{TETSponge}[\pi,\widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}},\pi,\pi^{-1},\widetilde{\mathsf{IC}},\widetilde{\mathsf{IC}}^{-1}} \Rightarrow 1]\right| \\ \leq & \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}. \end{aligned} \tag{15}$$

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.

[3] Regarding 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 $\frac{u^2}{2^n}$ emerges.

```
**algorithm** $\mathsf{Enc}_{K,PK}(N, A, M)$
  $\ell \leftarrow \lceil |M|/r \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
  parse $M$ as $M[1]\|\ldots\|M[\ell]$, with
    $|M[1]| = \ldots = |M[\ell-1]| = r$ and
    $1 \leq |M[\ell]| \leq r$
  parse $A$ as $A[1]\|\ldots\|A[\nu]$, with
    $|A[1]| = \ldots = |A[\nu-1]| = r$ and
    $1 \leq |A[\nu]| \leq r$
  $B \leftarrow \widetilde{\mathsf{E}}_K^{PK\|0^{n-n_P}}(N\|0^{n-n_N})$
  $IV \leftarrow N\|PK\|0^{b-n-n_N-n_P}$
  $S_0 \leftarrow IV\|B$, $S_1 \leftarrow \pi(S_0)$
  **if** $\nu \geq 1$ **then**
    **for** $i = 1$ **to** $\nu - 1$ **do**
      $S_i \leftarrow S_i \oplus (A[i]\|0^c)$
      $S_{i+1} \leftarrow \pi(S_i)$
    **if** $|A[\nu]| < r$ **then**
      $A[\nu] \leftarrow A[\nu]\|10^{r-|A[\nu]|-1}$
      $S_\nu \leftarrow S_\nu \oplus (0^r\|[1]_2\|0^{c-2})$
    $S_\nu \leftarrow S_\nu \oplus (A[\nu]\|0^c)$
    $S_{\nu+1} \leftarrow \pi(S_\nu)$
  **if** $\ell \geq 1$ **then**
    $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r\|[2]_2\|0^{c-2})$
    **for** $i = 1$ **to** $\ell - 1$ **do**
      $j \leftarrow i + \nu$
      $C[i] \leftarrow \mathsf{msb}_r(S_j) \oplus M[i]$
      $S_j \leftarrow C[i]\|\mathsf{lsb}_c(S_j)$
      $S_{j+1} \leftarrow \pi(S_j)$
    $C[\ell] \leftarrow \mathsf{msb}_{|M[\ell]|}(S_{\nu+\ell}) \oplus M[\ell]$
    **if** $|C[\ell]| < r$ **then**
      $S_{\nu+\ell} \leftarrow S_{\nu+\ell} \oplus (0^r\|[1]_2\|0^{c-2})$
      $S_{\nu+\ell} \leftarrow C[\ell]\|10^{r-|C[\ell]|-1}\|\mathsf{lsb}_c(S_{\nu+\ell})$
    **else** $S_{\nu+\ell} \leftarrow C[\ell]\|\mathsf{lsb}_c(S_{\nu+\ell})$
    $S_{\nu+\ell+1} \leftarrow \pi(S_{\nu+\ell})$
  $U\|V \leftarrow \mathsf{msb}_{2n-1}(S_{\nu+\ell+1})$
  $Z \leftarrow \widetilde{\mathsf{E}}_K^{V\|1}(U)$
  $\mathbf{c} \leftarrow C[1]\|\ldots\|C[\ell]$, $C \leftarrow \mathbf{c}\|Z$
  **return** $C$
```

```
**algorithm** $\mathsf{Dec}_{K,PK}(N, A, C)$
  $\ell \leftarrow \lceil \frac{|C|-n}{r} \rceil$, $\nu \leftarrow \lceil |A|/r \rceil$
  parse $C$ as $C[1]\|\ldots\|C[\ell]\|Z$, with
    $|C[1]| = \ldots = |C[\ell-1]| = r$, $1 \leq |C[\ell]| \leq r$, and $|Z| = n$
  parse $A$ as $A[1]\|\ldots\|A[\nu]$, with $|A[1]| =$
    $\ldots = |A[\nu-1]| = r$ and $1 \leq |A[\nu]| \leq r$
  $B \leftarrow \widetilde{\mathsf{E}}_K^{PK\|0^{n-n_P}}(N\|0^{n-n_N})$
  $IV \leftarrow N\|PK\|0^{b-n-n_N-n_P}$
  $S_0 \leftarrow IV\|B$, $S_1 \leftarrow \pi(S_0)$
  **if** $\nu \geq 1$ **then**
    **for** $i = 1$ **to** $\nu - 1$ **do**
      $S_i \leftarrow S_i \oplus (A[i]\|0^c)$
      $S_{i+1} \leftarrow \pi(S_i)$
    **if** $|A[\nu]| < r$ **then**
      $A[\nu] \leftarrow A[\nu]\|10^{r-|A[\nu]|-1}$
      $S_\nu \leftarrow S_\nu \oplus (0^r\|[1]_2\|0^{c-2})$
    $S_\nu \leftarrow S_\nu \oplus (A[\nu]\|0^c)$
    $S_{\nu+1} \leftarrow \pi(S_\nu)$
  **if** $\ell \geq 1$ **then**
    $S_{\nu+1} \leftarrow S_{\nu+1} \oplus (0^r\|[2]_2\|0^{c-2})$
    **for** $i = 1$ **to** $\ell - 1$ **do**
      $j \leftarrow i + \nu$
      $M[i] \leftarrow \mathsf{msb}_r(S_j) \oplus C[i]$
      $S_j \leftarrow C[i]\|\mathsf{lsb}_c(S_j)$
      $S_{j+1} \leftarrow \pi(S_j)$
    $M[\ell] \leftarrow \mathsf{msb}_{|C[\ell]|}(S_{\nu+\ell}) \oplus C[\ell]$
    **if** $|C[\ell]| < r$ **then**
      $S_{\nu+\ell} \leftarrow S_{\nu+\ell} \oplus (0^r\|[1]_2\|0^{c-2})$
      $S_{\nu+\ell} \leftarrow C[\ell]\|10^{r-|C[\ell]|-1}\|\mathsf{lsb}_c(S_{\nu+\ell})$
    **else** $S_{\nu+\ell} \leftarrow C[\ell]\|\mathsf{lsb}_c(S_{\nu+\ell})$
    $S_{\nu+\ell+1} \leftarrow \pi(S_{\nu+\ell})$
  $U\|V \leftarrow \mathsf{msb}_{2n-1}(S_{\nu+\ell+1})$
  $U^* \leftarrow (\widetilde{\mathsf{E}}_K^{V\|1})^{-1}(Z)$
  **if** $U \neq U^*$ **then return** $\perp$
  **else if** $\ell > 0$ **then return** $M[1]\|\ldots\|M[\ell]$
  **else return true**
```

Figure 7: Specification of $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{E}}]$ using a b-bit permutation π and an (n, n, n)-TBC $\widetilde{\mathsf{E}}$.

Permutation query transcripts. We summarize the adversarial queries to the random permutation π in

$$\tau_\pi = \big((S_1^{in}, S_1^{out}), \ldots, (S_{q_\pi}^{in}, S_{q_\pi}^{out})\big).$$

From our unbounded-leakage assumption, all the (encryption and decryption) queries made to $\mathsf{TETSponge}$ give all the ephemeral intermediate values to $\mathcal{D}$. Therefore, these queries result in records of the form (S^{in}, S^{out}). To make a distinction, we denote by τ_π^* the union of these leakage records and the adversarial query transcript τ_π. It can be seen that:

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

Therefore, when interacting with the idealized scheme $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$, the number of internal π-calls is at most $\sigma + q_e + q_d$, and we have

$$Q := \left|\tau_\pi^*\right| \le \sigma + q_e + q_d + q_\pi. \tag{16}$$

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 $\mathbf{c}$, the duplex behaves as evaluating a keyless hash function on $(N, PK, A, \mathbf{c})$, and all the underlying permutation calls are known by the adversary. Note that $\mathbf{c}$ 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 τ_h^*, which is defined upon τ_π^*, as the list of the inputs and outputs of this duplex hashing. Formally, a tuple $((N, PK, A, \mathbf{c}), S)$, $N \in \{0,1\}^{n_N}$, $PK \in \{0,1\}^{n_p}$, $A, \mathbf{c} \in \{0,1\}^*$, and $S \in \{0,1\}^b$, is in τ_h^*, if and only if there exists a sequence of π queries (S_0^{in}, S_0^{out}), (S_1^{in}, S_1^{out}),..., $(S_\omega^{in}, S_\omega^{out})$ in τ_π^* such that:

- $S_0^{in} = N \| PK \| 0^* \| B$ for some $B \in \{0,1\}^n$, and
- With $\nu = \lceil |A|/r \rceil$ and $\ell = \lceil |M|/r \rceil$, it holds $\omega = \nu + \ell$, and
  - For $i = 1, \ldots, \nu - 1$, $\mathsf{lsb}_c(S_i^{in}) = \mathsf{lsb}_c(S_{i-1}^{out})$, $\mathsf{msb}_r(S_i^{in}) \oplus \mathsf{msb}_r(S_{i-1}^{out}) = A[i]$;
  - When $|A[\nu]| < r$, it holds $\mathsf{lsb}_c(S_\nu^{in}) = \mathsf{lsb}_c(S_{\nu-1}^{out}) \oplus (0^r \| [1]_2 \| 0^{c-2})$ and $\mathsf{msb}_r(S_\nu^{in}) \oplus \mathsf{msb}_r(S_{\nu-1}^{out}) = A[\nu] \| 10^*$; when $|A[\nu]| = r$, it holds $\mathsf{lsb}_c(S_\nu^{in}) = \mathsf{lsb}_c(S_{\nu-1}^{out})$ and $\mathsf{msb}_r(S_\nu^{in}) \oplus \mathsf{msb}_r(S_{\nu-1}^{out}) = A[\nu]$;
  - If $\ell > 1$:
    - $\mathsf{lsb}_c(S_{\nu+1}^{in}) = \mathsf{lsb}_c(S_\nu^{out}) \oplus (0^r \| [2]_2 \| 0^{c-2})$, and $\mathsf{msb}_r(S_{\nu+1}^{in}) = C[1]$;
    - For $i = \nu + 2, \ldots, \omega - 1$, $\mathsf{lsb}_c(S_i^{in}) = \mathsf{lsb}_c(S_{i-1}^{out})$, $\mathsf{msb}_r(S_i^{in}) = C[i - \nu]$;
    - When $|C[\ell]| < r$, $\mathsf{lsb}_c(S_\omega^{in}) = \mathsf{lsb}_c(S_{\omega-1}^{out}) \oplus (0^r \| [1]_2 \| 0^{c-2})$, $\mathsf{msb}_r(S_\omega^{in}) = C[\ell] \| 10^*$; when $|C[\ell]| = r$, $\mathsf{lsb}_c(S_\omega^{in}) = \mathsf{lsb}_c(S_{\omega-1}^{out})$, and $\mathsf{msb}_r(S_\omega^{in}) = C[\ell]$.
  - If $\ell = 1$:
    - When $|C[\ell]| < r$, $\mathsf{lsb}_c(S_\omega^{in}) = \mathsf{lsb}_c(S_{\omega-1}^{out}) \oplus (0^r \| [3]_2 \| 0^{c-2})$, $\mathsf{msb}_r(S_\omega^{in}) = C[\ell] \| 10^*$; when $|C[\ell]| = r$, $\mathsf{lsb}_c(S_\omega^{in}) = \mathsf{lsb}_c(S_{\omega-1}^{out}) \oplus (0^r \| [2]_2 \| 0^{c-2})$, and $\mathsf{msb}_r(S_\omega^{in}) = C[\ell]$.
- $S = S_\omega^{out}$.

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

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

Records for key derivation and tag generation. In some sense, the adversarial goal is to distinguish $\widetilde{\mathsf{IC}}_{K_1}, \ldots, \widetilde{\mathsf{IC}}_{K_u}$ from $\widetilde{\mathsf{SIC}}_{K_1}, \ldots, \widetilde{\mathsf{SIC}}_{K_u}$. In this respect, at the end of the interaction, we reveal all the internal calls to $\widetilde{\mathsf{IC}}$ (in the real world) and $\widetilde{\mathsf{SIC}}$ (in the ideal world) to $\mathcal{D}$. We summarize these calls in a list

$$\tau_{\widetilde{\mathsf{SIC}}} = \big((K_1, T_1, X_1, Y_1), (K_2, T_2, X_2, Y_2), \ldots\big).$$

In this list, the j-th tuple (K_j, T_j, X_j, Y_j) indicates that:

- interacting with the real scheme $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}}$, the j-th query is either $\widetilde{\mathsf{IC}}_{K_j}^{T_j}(X_j) \to Y_j$ or $(\widetilde{\mathsf{IC}}_{K_j}^{T_j})^{-1}(Y_j) \to X_j$; and,
- interacting with the idealized scheme $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}$, the j-th query is either $\widetilde{\mathsf{SIC}}_{K_j}^{T_j}(X_j) \to Y_j$ or $(\widetilde{\mathsf{SIC}}_{K_j}^{T_j})^{-1}(Y_j) \to X_j$.

Note that:

- these calls and their responses are secret in the black-box setting, but are leaked in our unbound leakage setting.
- yet, since we assume leak-freeness of internal TBC-calls (i.e., TBC-calls made by $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]$), the secret user keys cannot be seen by the distinguisher, and do not appear in the true adversarial transcripts. The transcript $\tau_{\widetilde{\mathsf{SIC}}}$, in some sense, is a merge of the secret keys (augmented by us, as mentioned) and the information really leaked to $\mathcal{D}$.

Summary. With all the above, we consider the four tuples $\tau_h^*, \tau_\pi^*, \tau_{\widetilde{\mathsf{IC}}}, \tau_{\widetilde{\mathsf{SIC}}}$ appended by the keys $\mathbf{PK}, \mathbf{K}$. This yields what we call the *transcript*

$$\tau = (\tau_h^*, \tau_\pi^*, \tau_{\widetilde{\mathsf{IC}}}, \tau_{\widetilde{\mathsf{SIC}}}, \mathbf{PK}, \mathbf{K}).$$

5.2 Establishing Eq. (15)

First of all, for a transcript τ, we define two quantities μ_{PK} and μ_V, the *maximum multiplicity of PK and V*, as

$$\mu_{PK} := \max_{pk \in \{0,1\}^{n_p}} \big|\{i \in \{1, \ldots, u\} : PK_i = pk\}\big|,$$
$$\mu_V := \max_{v \in \{0,1\}^{n-1}} \Big|\Big\{((N, PK, A, \mathbf{c}), S) \in \tau_h^* : V = v\Big\}\Big|. \qquad (17)$$

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 τ 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 $(K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}}$ such that $(K, T, \star, \star) \in \tau_{\widetilde{\mathsf{IC}}}$.

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

Intuitions of the conditions. G_2 deviates from G_1 in the use of the secret ideal TBC $\widetilde{\mathsf{SIC}}$. The difference thus stems from the contradictions between $\tau_{\widetilde{\mathsf{IC}}}$ and $\tau_{\widetilde{\mathsf{SIC}}}$, which is captured by (B-2). Though, the straightforward approach to bound Pr[(B-2)] gives rise to a poorer bound $\Pr[(\text{B-2})] = O(uq_{\widetilde{\mathsf{IC}}}/2^n)$. To rescue, we note that a query $(K, T, \star, \star) \in \tau_{\widetilde{\mathsf{IC}}}$ can only "hit" queries in $\tau_{\widetilde{\mathsf{SIC}}}$ with the specific tweak T. Thus precisely, the probability that each adversarial query to $\widetilde{\mathsf{IC}}$ is $O(\mu_T/2^n)$ rather than $O(|\tau_{\widetilde{\mathsf{SIC}}}|/2^n)$, where μ_T is the maximum number of times a specific tweak value appears in $\tau_{\widetilde{\mathsf{SIC}}}$. 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

$$\Pr[\mu_{PK} \geq n+1] \leq \binom{u}{n+1} \cdot \frac{1}{(2^{n_p})^n} \leq \Big(\frac{u}{2^{n_p}}\Big)^{n+1} \cdot \frac{2^{n_p}}{(n+1)!} \leq \Big(\frac{u}{2^{n_p}}\Big)^{n+1},$$

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

$$\Pr[\mu_{PK} \geq n+1] \leq \Big(\frac{u}{2^{n_p}}\Big)^{n+1} \leq \frac{u}{2^{n_p}}. \tag{18}$$

To reason about μ_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 $\mathcal{D}$ with the ideal world $(\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}}, \pi, \widetilde{\mathsf{IC}})$. We define several simple bad events during this interaction:

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

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

$$\Pr\left[\mathsf{lsb}_{c-2}(S^{out}) = \mathsf{lsb}_{c-2}(S^{in'})\right] \leq \frac{2^{r+2}}{2^b - Q} \leq \frac{2^{r+3}}{2^b} = \frac{8}{2^c}.$$

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

$$\Pr[(\text{B-11})] \leq q_1 \cdot Q \cdot \Big(\frac{8}{2^c} + \frac{8}{2^c}\Big) \leq \frac{16 q_1 Q}{2^c}.$$

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

$$\Pr[(\text{B-12})] \leq q_2 \cdot Q \cdot \frac{8}{2^c} + q_2 \cdot (q_e + q_d) \cdot \frac{2}{2^b} \leq \frac{10 q_2 Q}{2^c}, \text{ and}$$
$$\Pr[(\text{B-13})] \leq \binom{q_1}{n+1} \cdot \Big(\frac{2}{2^{n-1}}\Big)^n \leq \Big(\frac{8Q}{2^n}\Big)^{n+1} \cdot \frac{1}{8(n+1)!} \leq \Big(\frac{4Q}{2^n}\Big)^{n+1} \cdot \frac{1}{8} \leq \frac{Q}{2^n},$$

since $4Q \leq 2^n$.

To bound Pr[(B-14)], we define a set

$$\tau^*_\pi\big[N, PK\big] := \big\{B \in \{0,1\}^n : (N\|PK\|0^*\|B, \star) \in \tau^*_\pi\big\}. \tag{19}$$

Then, for a certain $\widetilde{\mathsf{SIC}}$ query $\widetilde{\mathsf{SIC}}_K^{PK\|0^*}(N\|0^*) \to B$, we have

$$\Pr\big[B \in \tau^*_\pi[N, PK]\big] \le \frac{\big|\tau^*_\pi[N, PK_i]\big|}{2^n - 2q_e - 2q_d} \le \frac{2\big|\tau^*_\pi[N, PK_i]\big|}{2^n}.$$

Summing over all the $\widetilde{\mathsf{SIC}}$ queries, we find

$$\begin{aligned}\Pr[\text{(B-14)}] \le & \sum_{(K, PK\|0^*, N\|0^*, B) \in \tau_{\widetilde{\mathsf{SIC}}}} \frac{2\big|\tau^*_\pi\big[N, PK\big]\big|}{2^n} \\ \le & \sum_{i=1}^{u} \frac{2\sum_{N\in\{0,1\}^{n_N} : (K_i, PK_i\|0^*, N\|0^*, \star) \in \tau_{\widetilde{\mathsf{SIC}}}} \big|\tau^*_\pi\big[N, PK_i\big]\big|}{2^n} \\ \le & \mu_{PK} \cdot \sum_{N\in\{0,1\}^{n_N}, PK\in\{0,1\}^{n_p}} \frac{2\big|\tau^*_\pi\big[N, PK\big]\big|}{2^n} \le \frac{2nQ}{2^n},\end{aligned}$$

since $\sum_{N\in\{0,1\}^{n_N}, PK\in\{0,1\}^{n_p}} \big|\tau^*_\pi\big[N, PK\big]\big| = |\tau^*_\pi| \le Q$.

Let $\mathsf{Bad}_1 :=$ (B-11) $\vee$ (B-12) $\vee$ (B-13) $\vee$ (B-14). We now show that $\mu_V \le n$ given $\neg\mathsf{Bad}_1$, so that (using $q_1 + q_2 \le Q$)

$$\Pr[\mu_V \ge n+1] \le \Pr[\mathsf{Bad}_1] \le \frac{16q_1Q}{2^c} + \frac{10q_2Q}{2^c} + \frac{Q}{2^n} + \frac{2nQ}{2^n} \le \frac{16Q^2}{2^c} + \frac{(2n+1)Q}{2^n}.$$

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

Lemma 5. *Conditioned on $\neg\mathsf{Bad}_1$, we show that two distinct tuples $((N, PK, A, \mathbf{c}), S)$ and $((N', PK', A', \mathbf{c}'), S')$ in τ^*_h necessarily have $S \ne S'$. Moreover, both S and S' were the outputs of forward π queries during the interaction.*

The proof is deferred to subsection 5.3 for cleanness. By Lemma 5, the "final" π queries of the $|\tau^*_h|$ hash records are $|\tau^*_h|$ distinct forward queries. Conditioned on $\neg$(B-13), the number of semi-collisions on V within these final π queries is at most n. Therefore, the claim $\mu_V \le n$ follows.

Analyzing (B-2) conditioned on ¬(B-1). Note that in the ideal world, for any query $(K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}}$, the key K is from the dummy key-tuple $\mathbf{K}$, and is uniformly distributed. Then, using an auxiliary set

$$\tau_{\widetilde{\mathsf{IC}}}[T] := \big\{K \in \{0,1\}^n : (K, T, \star, \star) \in \tau_{\widetilde{\mathsf{IC}}}\big\},$$

we can bound

$$\begin{aligned}\Pr[\text{(B-2)}] \le & \sum_{(K,T,\star,\star)\in\tau_{\widetilde{\mathsf{SIC}}}} \Pr\big[K \in \tau_{\widetilde{\mathsf{IC}}}[T]\big] \\ \le & \underbrace{\sum_{t\in\{0,1\}^{n-1} : (K, t\|0, \star, \star)\in\tau_{\widetilde{\mathsf{SIC}}}} \frac{\big|\tau_{\widetilde{\mathsf{IC}}}[t\|0]\big|}{2^n}}_{C_1} + \underbrace{\sum_{V\in\{0,1\}^{n-1} : (K, V\|1, \star, \star)\in\tau_{\widetilde{\mathsf{SIC}}}} \frac{\big|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\big|}{2^n}}_{C_2}.\end{aligned}$$

By the construction, the $\widetilde{\mathsf{SIC}}$ queries $(K, t\|0, \star, \star)$ are necessarily key derivation queries, for which $K = K_i$ and $t\|0 = PK_i\|0^*$ for some user index i. Since $\mu_{PK} \le n$, we have

$$C_1 = \sum_{i=1}^{u} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[PK_i\|0^*]\right|}{2^n} \le \mu_{PK} \cdot \sum_{PK\in\{0,1\}^{n_p}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[PK\|0^*]\right|}{2^n} \le n \cdot \sum_{PK\in\{0,1\}^{n_p}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[PK\|0^*]\right|}{2^n}.$$

Moreover, for any $\widetilde{\mathsf{SIC}}$ query $(K_i, V\|1, \star, \star)$, i.e., tag generation query, there necessarily exists at least one hash record $((N, PK, A, \mathbf{c}), S) \in \tau_h^*$ such that $PK = PK_i$. By this,

$$\begin{aligned}
C_2 =& \sum_{i=1}^{u} \sum_{V:((\star,PK_i,\star,\star),S)\in\tau_h^*:\mathsf{mid}(S)=V} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\right|}{2^n} \\
\le& \mu_{PK} \cdot \sum_{PK\in\{0,1\}^{n_p}} \sum_{V:((\star,PK,\star,\star),S)\in\tau_h^*:\mathsf{mid}(S)=V} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\right|}{2^n} \\
\le& \mu_{PK} \cdot \mu_V \cdot \sum_{V\in\{0,1\}^{n-1}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\right|}{2^n} \;\le\; n^2 \cdot \sum_{V\in\{0,1\}^{n-1}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\right|}{2^n}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Pr[\text{(B-2)} \mid \neg\text{(B-1)}] \le& n \cdot \sum_{PK\in\{0,1\}^{n_p}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[PK\|0^*]\right|}{2^n} + n^2 \cdot \sum_{V\in\{0,1\}^{n-1}} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[V\|1]\right|}{2^n} \\
\le& n^2 \cdot \sum_{t\in\{0,1\}^n} \frac{\left|\tau_{\widetilde{\mathsf{IC}}}[t]\right|}{2^n} \le \frac{n^2 q_{\widetilde{\mathsf{IC}}}}{2^n},
\end{aligned}$$

which allows us to conclude

$$\Pr[T_{id} \in \mathcal{T}_{bad}] \le \Pr[\text{(B-1)}] + \Pr[\text{(B-2)} \mid \neg\text{(B-1)}] \le \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}.$$

5.2.2 Finishing the application of the H-coefficient technique

Now consider a good transcript $\tau = (\tau_h^*, \tau_\pi^*, \tau_{\widetilde{\mathsf{IC}}}, \tau_{\widetilde{\mathsf{SIC}}}, \mathbf{PK}, \mathbf{K})$. Let

$$\tau_{\widetilde{\mathsf{SIC}}}[K, T] := \left\{(X, Y) \in (\{0,1\}^n)^2 : (K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}}\right\}.$$

With this notation, it is clear that

$$\Pr[T_{id} = \tau] = \Pr[\mathbf{K}, \mathbf{PK}] \cdot \Pr[\pi \vdash \tau_\pi^*] \cdot \Pr[\widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}] \cdot \prod_{(K,T)} \frac{1}{(2^n)_{|\tau_{\widetilde{\mathsf{SIC}}}[K,T]|}}.$$

Furthermore,

$$\begin{aligned}
\Pr[T_{re} = \tau] =& \Pr[\mathbf{K}, \mathbf{PK}] \cdot \Pr[\pi \vdash \tau_\pi^*] \cdot \Pr[\widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{SIC}}} \mid \widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}] \cdot \Pr[\widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}] \\
=& \Pr[\mathbf{K}, \mathbf{PK}] \cdot \Pr[\pi \vdash \tau_\pi^*] \\
& \cdot \Pr[\widetilde{\mathsf{IC}}_K^T(X) = Y \text{ for all } (K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}} \mid \widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}] \cdot \Pr[\widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}],
\end{aligned}$$

Since τ is good,

$$\begin{aligned}
&\Pr[\widetilde{\mathsf{IC}}_K^T(X) = Y \text{ for all } (K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}} \mid \widetilde{\mathsf{IC}} \vdash \tau_{\widetilde{\mathsf{IC}}}] \\
=& \Pr[\widetilde{\mathsf{IC}}_K^T(X) = Y \text{ for all } (K, T, X, Y) \in \tau_{\widetilde{\mathsf{SIC}}}] = \prod_{(K,T)} \frac{1}{(2^n)_{|\tau_{\widetilde{\mathsf{SIC}}}[K,T]|}}.
\end{aligned}$$

In conclusion, for any good transcript τ we have $\Pr[T_{re} = \tau] = \Pr[T_{id} = \tau]$, and thus

$$\begin{aligned}&\left|\Pr[\mathcal{D}^{\mathsf{TETSponge}[\pi,\widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}},\pi,\pi^{-1},\widetilde{\mathsf{IC}},\widetilde{\mathsf{IC}}^{-1}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{TETSponge}[\pi,\widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}},\pi,\pi^{-1},\widetilde{\mathsf{IC}},\widetilde{\mathsf{IC}}^{-1}} \Rightarrow 1]\right| \\ \le &\frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}.\end{aligned}$$

5.3 Proof of Lemma 5

Assume that the query chains underlying the two hashing $((N, PK, A, \mathbf{c}), S)$ and $((N', PK', A', \mathbf{c}'), S')$ are (S_0^{in}, S_0^{out}),..., $(S_\omega^{in}, S_\omega^{out})$ and $(S_0^{in'}, S_0^{out'})$,..., $(S_{\omega'}^{in'}, S_{\omega'}^{out'})$ respectively, and let $\lceil |A|/r \rceil = \nu$, $\lceil |M|/r \rceil = \ell$, $\lceil |A'|/r \rceil = \nu'$, and $\lceil |M'|/r \rceil = \ell'$. Then proving $S \neq S'$ means proving $S_\omega^{in} \neq S_{\omega'}^{in'}$. Note that, conditioned on $\neg$(B-12) and $\neg$(B-14), the first records (S_0^{in}, S_0^{out}) and $(S_0^{in'}, S_0^{out'})$ were necessarily resulted from forward π 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 $S_\omega^{in} \neq S_{\omega'}^{in'}$, we distinguish several cases as follows.

5.3.1 Case 1: $(N, PK) \neq (N', PK')$.

Then $S_0^{in} \neq S_0^{in'}$, i.e., the two chains are already distinct at the first π queries. By $\neg$(B-11), we have $S_1^{in} \neq S_1^{in'}$; similarly, iteratively applying $\neg$(B-11) eventually results in the desired conclusion $S_\omega^{in} \neq S_{\omega'}^{in'}$.

5.3.2 Case 2: $(N, PK) = (N', PK')$.

This means $(A, \mathbf{c}) \neq (A', \mathbf{c}')$. We define $\overline{X}$ as:

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

Then we have to further consider several subcases.

Subcase 2.1: $\overline{A}\|\overline{\mathbf{c}} \neq \overline{A'}\|\overline{\mathbf{c}'}$. It is clear that there exists an index i such that $S_i^{in} \neq S_i^{in'}$. By $\neg$(B-11), $S_j^{in} \neq S_j^{in'}$ for any $j > i$, and thus $S_\omega^{in} \neq S_{\omega'}^{in'}$.

Subcase 2.2: $\overline{A}\|\overline{\mathbf{c}} = \overline{A'}\|\overline{\mathbf{c}'}$ and $\nu = \nu'$. Since $(A, \mathbf{c}) \neq (A', \mathbf{c}')$, it has to be $|A[\nu]| < r$ or $|C[\ell]| < r$ or $|A'[\nu']| < r$ or $|C'[\ell']| < r$. Now,

- If $|A[\nu]| < r \wedge |A'[\nu]| = r$ or $|A[\nu]| = r \wedge |A'[\nu]| < r$, then $S_\nu^{in'} \neq S_\nu^{in}$ due to the separation constant $[1]_2\|0^{c-2}$. Thus by $\neg$(B-11), $S_j^{in} \neq S_j^{in'}$ for any $j > \nu$ and further $S_\omega^{in} \neq S_{\omega'}^{in'}$.
- Else, then either $|\mathbf{c}[\ell]| < r \wedge |\mathbf{c}'[\ell]| = r$ or $|\mathbf{c}[\ell]| = r \wedge |\mathbf{c}'[\ell]| < r$ since $(A, \mathbf{c}) \neq (A', \mathbf{c}')$. Then $S_\omega^{in'} \neq S_{\omega'}^{in}$ due to the separation constant $[2]_2\|0^{c-2}$.

Subcase 2.3: $\overline{A}\|\overline{\mathbf{c}} = \overline{A'}\|\overline{\mathbf{c}'}$, and $\nu \neq \nu'$. Wlog $\nu > \nu'$: then it has to be $\ell' \geq 1$. Now,

- If $|A[\nu']| < r$, then $S_{\nu'}^{in'} \neq S_{\nu'}^{in}$ since the separation constant $[1]_2\|0^{c-2}$ is only XORed into $S_{\nu'-1}^{out'}$, and thus all the subsequent calls are distinct.
- Else, if $\ell' = 1$, then $S_{\nu'+1}^{in'} \neq S_{\nu'+1}^{in}$ since $S_{\nu'+1}^{in'}$ is obtained by XORing $[3]_2\|0^{c-2}$ with $S_{\nu'}^{out'}$ while $S_{\nu'+1}^{in}$ is obtained by XORing either $[1]_2\|0^{c-2}$ or 0^c (depending on whether $|A[\nu'+1]| < r$).
- Else, i.e., $\ell' > 1$, then $S_{\nu'+1}^{in'} \neq S_{\nu'+1}^{in}$ since $S_{\nu'+1}^{in'}$ is obtained by XORing $[2]_2\|0^{c-2}$ with $S_{\nu'}^{out'}$ while $S_{\nu'+1}^{in}$ is obtained by XORing either $[1]_2\|0^{c-2}$ or 0^c.

In all, in all cases, we will have $S_\omega^{in} \neq S_{\omega'}^{in'}$. 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., DPAs 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 $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Enc}}, \mathsf{L}^*_{\mathsf{Dec}})$, where:

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

We further write $\overrightarrow{q} = (q_e, q_d, q_{\widetilde{\mathsf{IC}}}, q_\pi)$, and denote by $(\overrightarrow{q}, \sigma)$*-adversaries* the adversaries that respectively make q_e, q_d, $q_{\widetilde{\mathsf{IC}}}$, and q_π queries to $\mathsf{LEnc}_{\mathbf{K}}$, $\mathsf{LDec}_{\mathbf{K}}$, $\widetilde{\mathsf{IC}}$, and π, and have at most σ 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 $u \le 2^{n_p}$, $n_p \le n$, $n \ge 5$, $Q = \sigma + q_e + q_d + q_\pi \le \min\{2^n/4, 2^b/2\}$, and leakage L^* is "unbounded" as above. Then in the ideal TBC and permutation model, for any $(\overrightarrow{q}, \sigma)$-adversary $\mathcal{A}$ it holds*

$$\mathbf{Adv}^{\mathsf{muCIML2}}_{\mathcal{A},\mathsf{TETSponge},\mathsf{L}^*,u} \le \frac{3u}{2^{n_p}} + \frac{32Q^2}{2^c} + \frac{7nQ + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}. \tag{20}$$

Inside the bounds, $q_\pi, q_{\widetilde{\mathsf{IC}}} = t$ reflect the time complexity. If $n = 128$, and with our chosen parameters $n_p = n - 1$ and $c = 2n$, the bound simplifies to $\frac{5u}{2^{127}} + \frac{2^{14}t + 2^7\sigma}{2^{128}}$, 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 $\overrightarrow{q}$-bounded adversary $\mathcal{A}$ against the real construction $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]$ is close to that against the ideal $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{SIC}}]$, i.e.,

$$\begin{aligned} &\mathbf{Adv}^{\mathsf{muCIML2}}_{\mathcal{D},\mathsf{TETSponge}[\pi,\widetilde{\mathsf{IC}}]_{\mathbf{K},\mathbf{PK}},\mathsf{L}^*,u} - \mathbf{Adv}^{\mathsf{muCIML2}}_{\mathcal{D},\mathsf{TETSponge}[\pi,\widetilde{\mathsf{SIC}}]_{\mathbf{K},\mathbf{PK}},\mathsf{L}^*,u} \\ \le\ & \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}. \end{aligned} \tag{21}$$

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

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

- (C-1) There exist two user indices j, ℓ such that $K_j \| PK_j = K_\ell \| PK_\ell$, or $\mu_{PK} \ge n+1$.

- (C-2) Right after a forward π query $\pi(S^{in}) \to S^{out}$ is made,
  - (C-21) there exists an (earlier) π query $(S^{in'}, S^{out'})$ such that $\mathsf{lsb}_{c-2}(S^{out}) = \mathsf{lsb}_{c-2}(S^{in'})$, $\mathsf{lsb}_{c-2}(S^{out}) = \mathsf{lsb}_{c-2}(S^{out'})$, or $\mathsf{msb}_{2n-1}(S^{out}) = \mathsf{msb}_{2n-1}(S^{out'})$; or
  - (C-22) there exists a $\widetilde{\mathsf{SIC}}$ query $(K, V\|1, U, Z)$ (that corresponds to a tag generation) such that $\mathsf{msb}_{2n-1}(S^{out}) = U\|V$; or
- (C-3) At any time, there exist $n+1$ forward π queries $(S_1^{in}, S_1^{out}), \ldots, (S_{n+1}^{in}, S_{n+1}^{out})$ such that $\mathsf{mid}(S_1^{out}) = \ldots = \mathsf{mid}(S_{n+1}^{out})$, where $\mathsf{mid}(S_i^{out}) = \mathsf{msb}_{2n-1}(\mathsf{lsb}_{n-1}(S_i^{out}))$.
- (C-4) Right after a backward π query $\pi^{-1}(S^{out}) \to S^{in}$ is made, if:
  - (C-41) there exists another π query $(S^{in'}, S^{out'})$ such that $\mathsf{lsb}_{c-2}(S^{in}) = \mathsf{lsb}_{c-2}(S^{out'})$, or
  - (C-42) there exists a $\widetilde{\mathsf{SIC}}$ 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 $\widetilde{\mathsf{SIC}}_K^{PK\|0^*}(N\|0^*) \to B$ happens, there exists a π query (S^{in}, S^{out}) such that $S^{in} = N\|PK\|0^*\|B$.
- (C-6) Right after an inverse $\widetilde{\mathsf{SIC}}$ query/tag generation query $(\widetilde{\mathsf{SIC}}_K^{V\|1})^{-1}(Z) \to U$, there exists a π query (S^{in}, S^{out}) such that $\mathsf{msb}_{2n-1}(S^{out}) = U\|V$.

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

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

Second, (C-21) is the condition (B-11) in Section 5.2 enhanced with $\mathsf{msb}_{2n-1}(S^{out}) = \mathsf{msb}_{2n-1}(S^{out'})$, thus $\Pr[\text{(C-21)}] \le \frac{16q_1Q}{2^c} + \frac{2Q^2}{2^{2n-1}}$ (q_1 being the number of forward π queries). And it is clear that $\Pr[\text{(C-22)}] \le \frac{2(q_e+q_d)Q}{2^{2n-1}}$. Thus (using $q_e + q_d \le Q$)

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

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

The condition (C-3) is the same as the condition (B-13) in Section 5.2, thus $\Pr[\text{(C-3)}] \le \frac{Q}{2^n}$. The condition (C-4) is the condition (B-12) in Section 5.2, thus $\Pr[\text{(C-4)}] \le \frac{10q_2Q}{2^c}$. (C-5) is the condition (B-14) in in Section 5.2, thus

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

Finally consider (C-6). Conditioned on $\neg$(C-2) and $\neg$(C-3), an analysis similar to the previous one for μ_V (also in Section 5.2) indicates that the number of distinct records $((N_1, PK_1, A_1, \mathbf{c}_1), S_1), ((N_2, PK_2, A_2, \mathbf{c}_2), S_2), \ldots$ in τ_h^* with $\mathsf{mid}(S_1) = \mathsf{mid}(S_2) = \ldots$ is at most n. Therefore, for each inverse query $(\widetilde{\mathsf{SIC}}_K^{V\|1})^{-1}(Z) \to U$, there are at most n "target" U values, and thus

$$\Pr[\text{(C-6)}] \le \frac{nq_d}{2^n - q_{\widetilde{\mathsf{IC}}}} \le \frac{2nq_d}{2^n}. \tag{23}$$

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

$$\Pr[\mathsf{Bad}] \le \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+3)Q + 2nq_d}{2^n}. \tag{24}$$

Below we show $\Pr[\mathsf{CHAIN} \mid \neg\mathsf{Bad}] = 0$, so that $\Pr[\mathsf{CHAIN}] \le \Pr[\mathsf{Bad}]$. Assume otherwise, then consider the last adversarial action before CHAIN happens:

6.2.1 Case 1: $\mathcal{A}$ makes a π query

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

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

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

- Subcase 2.1: a subsequent (forward) π query causes CHAIN event. This again contradicts ¬(C-2).
- Subcase 2.2: the subsequent (new) tag generation query $\widetilde{\mathsf{SIC}}_{K_i}^{V\|1}(U) \rightarrow Z$ causes the event CHAIN. Assume that the involved hash record is $((N, PK_i, A, \mathbf{c}), S)$ which corresponds to the i-th user. The assumption means there exists another hash record of the j-th user $((N', PK_j, A', \mathbf{c}'), S')$ such that $\mathsf{msb}_{2n-1}(S) = \mathsf{msb}_{2n-1}(S')$. To reach a contradiction, we further distinguish two cases:
  - Subcase 2.3.1: $(N, A, \mathbf{c}) \neq (N', A', \mathbf{c}')$. Then as argued before (in Section 5.2), conditioned on ¬(C-21), the two involved hash chains are different, and thus $\mathsf{msb}_{2n-1}(S) = \mathsf{msb}_{2n-1}(S')$ would contradict ¬(C-21);
  - Subcase 2.3.2: $(N, A, \mathbf{c}) = (N', A', \mathbf{c}')$. Then it has to be $i \neq j$. Now, if $K_i \neq K_j$, then the new $\widetilde{\mathsf{SIC}}$ query $\widetilde{\mathsf{SIC}}_{K_i}^{V\|1}(U) \rightarrow Z$ has nothing to do with the j-th user. Otherwise, it holds $PK_i \neq PK_j$ by ¬(C-1), which means the two involved hash chains are different, and thus $\mathsf{msb}_{2n-1}(S) = \mathsf{msb}_{2n-1}(S')$ would contradict ¬(C-21).

6.2.3 Case 3: $\mathcal{A}$ makes a decryption query

We also consider two subcases:

- Subcase 3.1: a subsequent (forward) π query causes CHAIN event. Then it again contradicts ¬(C-2).
- Subcase 3.2: the subsequent (new) tag generation query $(\widetilde{\mathsf{SIC}}_{K_i}^{V\|1})^{-1}(Z) \rightarrow U$ causes CHAIN event. This contradicts ¬(C-6).

As conclusion, we have

$$\mathbf{Adv}^{\mathsf{muCIML2}}_{\mathcal{D},\mathsf{TETSponge}[\pi,\widetilde{\mathsf{SIC}}]_{\mathbf{PK}},\mathsf{L}^*,u} \leq \Pr[\mathsf{Bad}] \leq \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+3)Q + 2nq_d}{2^n}. \tag{25}$$

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

$$\begin{aligned} &\frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n} + \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+3)Q + 2nq_d}{2^n} \\ \leq\ &\frac{3u}{2^{n_p}} + \frac{32Q^2}{2^c} + \frac{5nQ + 2nq_d + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}. \end{aligned} \tag{26}$$

7 Leakage confidentiality of TETSponge

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 $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$ of our implementation as follows:

- $\mathsf{L}_{\mathsf{Enc}}$ consists of the leakages that are generated during the encryption:
  - — the leakages $\mathsf{L}_{\pi}^{in}(S^{in})$ and $\mathsf{L}_{\pi}^{out}(S^{out})$ generated by all the internal calls to $\pi(S^{in}) \to S^{out}$, and
  - — the leakages $\mathsf{L}_{\oplus}(a, b)$ generated by all the internal actions $a \oplus b$.
- $\mathsf{L}_{\mathsf{Dec}} = \emptyset$ since muCCAmL1 is CCA with encryption leakages only.

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

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

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

$$\mathbf{Adv}^{\mathsf{muCCAmL1}}_{\mathcal{A},\mathsf{TETSponge},\mathsf{L},u} \le \frac{6u}{2^{n_p}} + \frac{65Q^2}{2^c} + \frac{8(n+1)Q + 4nq_d + 2n^2 q_{\widetilde{\mathsf{IC}}}}{2^n} + \sigma \mathbf{Adv}^{\mathsf{LORL}}(Q, t^*) + 2q_e \mathbf{Adv}^{\mathsf{Inv}[n]}(Q, t^*, 2Q) + 2(\sigma + q_e)\mathbf{Adv}^{\mathsf{Inv}[c]}(Q, t^*, 2Q), \quad (27)$$

where $\mathbf{Adv}^{\mathsf{LORL}}$ and $\mathbf{Adv}^{\mathsf{Inv}[\omega]}$ are defined in Eqs. (8) and (5) resp, $Q = \sigma + q_e + q_d + q_m + q_\pi$, $t^* = O(t + \sigma t_l)$, and t_l 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 $2q_e \mathbf{Adv}^{\mathsf{Inv}[n]}(Q, t^*, 2Q) + 2(\sigma + q_e)\mathbf{Adv}^{\mathsf{Inv}[c]}(Q, t^*, 2Q)$ are in $O\Big(q_e \cdot \frac{q_\pi + \sigma + t}{\mu_n \cdot 2^n}\Big) + O\Big(\sigma \cdot \frac{q_\pi + \sigma + t}{\mu_c \cdot 2^c}\Big)$ for some specific parameters μ_n and μ_c. Though, the influence of the number u of users on the security remains negligible: once $u \le 2^{n_p}/5$, TETSponge 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 $M_1^0, M_2^0, \ldots$ is close to encrypting $M_1^1, M_2^1, \ldots$, we will replace the encryption of M_i^0 by M_i^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 $\mathsf{PrivK}^{\mathsf{muCCAmL1},0}_{\mathcal{A},\mathsf{TETSponge},\mathsf{L},u}$, and G_0^* as the game $\mathsf{PrivK}^{\mathsf{muCCAmL1},1}_{\mathcal{A},\mathsf{TETSponge},\mathsf{L},u}$. And we say a decryption query $\mathsf{Dec}_{\mathbf{K},\mathbf{PK}}(i, N, A, C)$ is *trivial* if the encryption action $\mathsf{Enc}_{\mathbf{K},\mathbf{PK}}(i, N, A, \star) \to C$ happens before. The subsequent proof consists of several small steps presented in the following paragraphs.

7.2.1 Idealizing TETSponge

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 $\widetilde{\mathsf{IC}}$-calls (made by $\mathsf{TETSponge}[\pi, \widetilde{\mathsf{IC}}]$) by $\widetilde{\mathsf{SIC}}$-calls. By Lemma 4, we have

$$\big|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]\big| \le \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}, \quad (28)$$

and (similarly)

$$\big|\Pr[\mathsf{G}_1^* \Rightarrow 1] - \Pr[\mathsf{G}_0^* \Rightarrow 1]\big| \le \frac{u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+1)Q + n^2 q_{\widetilde{\mathsf{IC}}}}{2^n}, \tag{29}$$

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

The gaps are actually the probability of forgery against the idealized $\mathsf{TETSponge}$ construction, which could be found in Eq. (24), Section 6. Therefore,

$$\big|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]\big| \le \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+3)Q + 2nq_d}{2^n} \tag{30}$$

$$\big|\Pr[\mathsf{G}_2^* \Rightarrow 1] - \Pr[\mathsf{G}_1^* \Rightarrow 1]\big| \le \frac{2u}{2^{n_p}} + \frac{16Q^2}{2^c} + \frac{(2n+3)Q + 2nq_d}{2^n}. \tag{31}$$

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 $\mathsf{G}_{3,0} = \mathsf{G}_2$, and define a sequence of games

$$\mathsf{G}_{3,1}, \mathsf{G}_{3,2}, \ldots, \mathsf{G}_{3,q_e},$$

such that in the j-th system $\mathsf{G}_{3,j}$, the first j messages processed by the challenge encryption oracle are $M_1^0, \ldots, M_j^0$, while the remaining $q_e - j$ messages being processed are $M_{j+1}^1, \ldots, M_{q_e}^1$. It can be seen that $\mathsf{G}_{3,q_e} = \mathsf{G}_2^*$.

We then show that for $j = 1, \ldots, q_e$, $\mathsf{G}_{3,j-1}$ and $\mathsf{G}_{3,j}$ are indistinguishable for $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$. For this, from $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$ we build an adversary $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$, such that $|\Pr[\mathsf{G}_{3,j-1} \Rightarrow 1] - \Pr[\mathsf{G}_{3,j} \Rightarrow 1]|$ is related to the bound proved in Theorem 1. In detail, initially, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ samples u distinct keys $K_1\|T_1, \ldots, K_u\|T_u$ for subsequent simulations. It also keeps a pair of tables $(\textsc{STable}, \textsc{STable}^{-1})$, which have entries of the form $\textsc{STable}(K, T, X) = Y$ and $\textsc{STable}^{-1}(K, T, Y) = X$, to simulate the secret ideal TBC $\widetilde{\mathsf{SIC}}$ via lazy sampling. $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ then runs $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$. Upon $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$'s action, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ reacts as follows.

Upon $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$ making a query to $\widetilde{\mathsf{IC}}$ or π: $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ simply relays the query to its corresponding oracle and relays the response.

Upon $\mathcal{A}$ making a (non-challenge) encryption query (i^*, N^*, A^*, M^*): $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ distinguishes two cases.

- If $(K_{i^*}, PK_{i^*}\|0^*, N^*\|0^*) \notin$ STable, then $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ samples an initial key B^* such that $(K_{i^*}, PK_{i^*}\|0^*, B^*) \notin$ STable^{-1}, defines STable$(K_{i^*}, PK_{i^*}\|0^*, N^*\|0^*) \leftarrow B^*$ and STable$^{-1}(K_{i^*}, PK_{i^*}\|0^*, B^*) \leftarrow N^*\|0^*$, and then runs the encryption $\mathsf{DuStr}_{B^*}[\pi](N^*\|PK_{i^*}\|0^*, A^*, M^*)$ to get the ciphertext $\mathbf{c}^*\|U^*\|V^*$ and leakages. $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ then computes $Z^* \leftarrow$ STable$(K_{i^*}, V^*\|1, U^*)$ (if $(K_{i^*}, V^*\|1, U^*) \notin$ STable then $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ defines the entry STable$(K_{i^*}, V^*\|1, U^*)$ to a newly sampled value Z^*). For this entire process $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ has to make $\ell_i^* + 1$ queries to π and consume $O(\ell_i t_l)$ time. Finally, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ returns the outputs $\mathbf{c}^*\|Z^*$ and the leakages to $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$;
- If $(K_{i^*}, PK_{i^*}\|0^*, N^*\|0^*) \in$ STable, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ simply runs the encryption process $\mathsf{DuStr}_{B^*}[\pi](N^*\|PK_{i^*}\|0^*, A^*, M^*)$ with $B^* =$ STable$(K_{i^*}, PK_{i^*}\|0^*, N^*\|0^*)$, computes $Z^* \leftarrow$ STable$(K_{i^*}, V^*\|1, U^*)$ on the obtained U^* and V^*, and returns $\mathbf{c}^*\|Z^*$ and the leakages to $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$. The cost is similar to the above case.

Upon $\mathcal{A}$ making a non-trivial decryption query (i^*, N^*, A^*, C^*): $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ parses $C^* = \mathbf{c}^*\|Z^*$, and computes the hashing digest $U^*\|V^*$ accordingly. Then,

- if $(K_{i^*}, V^*\|1, Z^*) \notin$ STable, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ samples U^{**} such that $(K_{i^*}, V^*\|1, U^{**}) \notin$ STable, and defines the two table entries as STable$(K_{i^*}, V^*\|1, U^{**}) \leftarrow Z^*$, STable$^{-1}(K_{i^*}, V^*\|1, Z^*) \leftarrow U^{**}$;
- if $(K_{i^*}, V^*\|1, U^{**}) \in$ STable, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ just sets $U^{**} \leftarrow$ STable$^{-1}(K_{i^*}, V^*\|1, Z^*)$.

Now $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ returns $(\perp, U^{**})$ to $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$ (regardless of $U^{**} = U^*$ or not). This requires $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ to make $\ell_i^* + 1$ queries to π.

Upon $\mathcal{A}$ submitting the ℓ-th challenge tuple $(i_\ell, N_\ell, A_\ell, M_\ell^0, M_\ell^1)$: it necessarily holds $(K_{i_\ell}, PK_{i_\ell}\|0^*, N_\ell\|0^*) \notin$ STable, since

- Previous encryption queries to the i_ℓ-th user cannot define STable$(K_{i_\ell}, T_{i_\ell}, C_0(N_\ell))$ due to the challenge nonce-respecting restriction on $\mathcal{A}$;
- Previous encryption queries to the $i_{\ell'}$-th users cannot define STable$(K_{i_\ell}, T_{i_\ell}, C_0(N_\ell))$ since $K_{i_\ell}\|T_{i_\ell} \neq K_{i_{\ell'}}\|T_{i_{\ell'}}$.

With these in mind, depending on ℓ, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ reacts as follows:

- When $\ell < j$, it encrypts M_ℓ^0 and returns. In detail, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ samples B_ℓ, defines the entries STable$(K_{i_\ell}, PK_{i_\ell}\|0^*, N_\ell\|0^*) \leftarrow B_\ell$ and STable$^{-1}(K_{i_\ell}, PK_{i_\ell}\|0^*, B_\ell) \leftarrow N_\ell\|0^*$, and then runs $\mathsf{DuStr}_{B_\ell}[\pi](M_\ell^0) \rightarrow \mathbf{c}_\ell\|U_\ell\|V_\ell$, generates the tag Z_ℓ accordingly and returns $\mathbf{c}_\ell\|Z_\ell$ and the leakages to $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$. The cost is similar to the non-challenge encryption queries.
- When $\ell = j$, it relays M_ℓ^0 and M_ℓ^1 to the eavesdropper EavL challenger of DuStr to obtain $\mathbf{c}_\ell^b\|U_\ell\|V_\ell$ and leakages leak_{enc}, then generates the tag Z_ℓ accordingly and returns $\mathbf{c}_\ell^b\|Z_\ell$ to $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$.
- When $\ell > j$, it simply encrypts M_ℓ^1 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 $\mathsf{G}_{3,j-1}$ or $\mathsf{G}_{3,j}$. As remarked before, besides running $\mathcal{A}^{\pi,\widetilde{\mathsf{IC}}}$, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ internally processes at most $q_m + q_e + q_d$ queries. Therefore, for $\mathsf{G}_{2,j}$ and $\mathsf{G}_{2,j-1}$, $\mathcal{A}_2^{\pi,\widetilde{\mathsf{IC}}}$ makes at most $\sigma + q_m + q_e + q_d + q_\pi = Q$ queries to π and spends $t^* = O(t + \sigma t_l)$ computations (mainly on evaluating the leakage functions). By all the above, we have

$$\left|\Pr[\mathsf{G}_{3,j} \Rightarrow 1] - \Pr[\mathsf{G}_{3,j-1} \Rightarrow 1]\right| \le \mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(Q, t^*, \ell_j),$$

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

$$\begin{aligned}\left|\Pr[\mathsf{G}_2^* \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]\right| &\le \sum_{j=1}^{q_e} \left|\Pr[\mathsf{G}_{3,j} \Rightarrow 1] - \Pr[\mathsf{G}_{3,j-1} \Rightarrow 1]\right| \\ &\le \sum_{j=1}^{q_e} \mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(Q, t^*, \ell_j). \qquad (32)\end{aligned}$$

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

$$\begin{aligned}&\left|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_0^* \Rightarrow 1]\right| \\ \le\ & \frac{6u}{2^{n_p}} + \frac{64Q^2}{2^c} + \frac{(8n+8)Q + 4nq_d + 2n^2 q_{\widetilde{\mathsf{IC}}}}{2^n} + \sum_{j=1}^{q_e} \mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(Q, t^*, \ell_j). \qquad (33)\end{aligned}$$

The term $\sum_{j=1}^{q_e} \mathbf{Adv}^{\mathsf{EavL}}_{\mathsf{LDuStr}}(Q, t^*, \ell_j)$ is bounded by Theorem 1 as

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

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 Dobraunig 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

[ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting Authenticated Encryption Robustness with Minimal Modifications. In *CRYPTO 2017, Part III*, pages 3–33, 2017.

[ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In *FSE 2015*, pages 364–384, 2015.

[AFM19] Alexandre Adomnicai, Jacques J.A. Fournier, and Laurent Masson. Masking the Lightweight Authenticated Ciphers ACORN and Ascon in Software. Cryptology ePrint Archive, Report 2018/708, 2019. Appeared at BalkanCryptSec 2018.

[BDF$^+$17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In *EUROCRYPT 2017*, pages 535–566, 2017.

[BDH$^+$19] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. The Keyak authenticated encryption scheme, 2019. https://competitions.cr.yp.to/caesar.html.

[BDPV10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-Based Pseudo-Random Number Generators. In *CHES 2010*, pages 33–47, 2010.

[BDPV11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *SAC 2011*, pages 320–337, 2011.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a leakage-resilient AEAD mode for high (physical) security applications. Cryptology ePrint Archive, Report 2019/137, 2019. https://eprint.iacr.org/2019/137.

[BGS15] Sonia Belaïd, Vincent Grosso, and François-Xavier Standaert. Masking and leakage-resilient primitives: One, the other(s) or both? *Cryptography and Communications*, 7(1):163–184, 2015.

[BHT18] Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: Multi-user Security, Faster Key Derivation, and Better Bounds. In *EUROCRYPT 2018, Part I*, pages 468–499, 2018.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives. In *AsiaCCS 2018*, pages 37–50, 2018.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In *ASIACRYPT 2017, Part I*, pages 693–723, 2017.

[BP17] Alex Biryukov and Leo Perrin. State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive, Report 2017/511, 2017. https://eprint.iacr.org/2017/511.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On Leakage-Resilient Authenticated Encryption with Decryption Leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[BT16] Mihir Bellare and Björn Tackmann. The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In *CRYPTO 2016, Part I*, pages 247–276, 2016.

[cae19] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2019. https://competitions.cr.yp.to/caesar.html.

[CS14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In *EUROCRYPT 2014*, pages 327–350, 2014.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - Towards Side-Channel Secure Authenticated Encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.

[DEMS19] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to NIST, 2019. Available: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/ascon-spec.pdf.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage Resilience of the Duplex Construction. Cryptology ePrint Archive, Report 2019/225, 2019.

[DMV17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In *ASIACRYPT 2017, Part II*, pages 606–637, 2017.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *FOCS 2008*, pages 293–302, 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In *CRYPTO 2010*, pages 21–40, 2010.

[FH15] Benjamin Fuller and Ariel Hamlin. Unifying Leakage Classes: Simulatable Leakage and Pseudoentropy. In *ICITS*, pages 69–86, 2015.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical Leakage-Resilient Symmetric Cryptography. In *CHES 2012*, pages 213–232, 2012.

[GL17] Shay Gueron and Yehuda Lindell. Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation. In *CCS 2017*, pages 1019–1036, 2017.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In *CT-RSA 2017*, pages 95–112, 2017.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In *EUROCRYPT 2017, Part I*, pages 567–597, 2017.

[HLWW16] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-Resilient Cryptography from Minimal Assumptions. *J. Cryptology*, 29(3):514–551, 2016.

[MR04] Silvio Micali and Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). In *TCC 2004*, pages 278–296, 2004.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In *ASIACRYPT 2015, Part II*, pages 465–489, 2015.

[oST18] National Institute of Standards and Technology. Lightweight Cryptography, 2018. https://csrc.nist.gov/projects/lightweight-cryptography.

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

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In *CCS 2015*, pages 96–108, 2015.

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

[RSWO17] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Security & Privacy*, 16(1):54–62, 2017.

[Sta19] François-Xavier Standaert. EUROCRYPT 2019. pages xv, https://www.youtube.com/watch?v=KdhrsuJT1sE, 2019.

[SY15] Yu Sasaki and Kan Yasuda. How to Incorporate Associated Data in Sponge-Based Authenticated Encryption. In *CT-RSA 2015*, pages 353–370, 2015.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical Leakage-Resilient Pseudorandom Generators. In *CCS 2010*, pages 141–151, 2010.