**Proceedings Paper:**
Barbosa, Manuel and Farshim, Pooya (2018) Indifferentiable Authenticated Encryption. In: Shacham, H. and Boldyreva, A., (eds.) Advances in Cryptology – CRYPTO 2018. , pp. 187-220.

https://doi.org/10.1007/978-3-319-96884-1_7

# Indifferentiable Authenticated Encryption

Manuel Barbosa[1] and Pooya Farshim[2,3]

[1] INESC TEC and FC University of Porto, Porto, Portugal
mbb@dcc.fc.up.pt
[2] DI/ENS, CNRS, PSL University, Paris, France
[3] Inria, Paris, France
pooya.farshim@gmail.com

**Abstract.** We study Authenticated Encryption with Associated Data (AEAD) from the viewpoint of composition in arbitrary (single-stage) environments. We use the indifferentiability framework to formalize the intuition that a "good" AEAD scheme should have random ciphertexts subject to decryptability. Within this framework, we can then apply the indifferentiability composition theorem to show that such schemes offer extra safeguards wherever the relevant security properties are not known, or cannot be predicted in advance, as in general-purpose crypto libraries and standards.

We show, on the negative side, that generic composition (in many of its configurations) and well-known classical and recent schemes fail to achieve indifferentiability. On the positive side, we give a provably indifferentiable Feistel-based construction, which reduces the round complexity from at least 6, needed for blockciphers, to only 3 for encryption. This result is not too far off the theoretical optimum as we give a lower bound that rules out the indifferentiability of *any* construction with less than 2 rounds.

**Keywords.** Authenticated encryption, indifferentiability, composition, Feistel, lower bound, CAESAR.

# Table of Contents

## 1 Introduction

Authenticated-Encryption with Associated-Data (AEAD) [Rog02, BN00] is a fundamental building block in cryptographic protocols, notably those enabling secure communication over untrusted networks. The syntax, security, and constructions of AEAD have been studied in numerous works. Recent, ongoing standardization processes, such as the CAESAR competition [Ber14] and TLS 1.3, have revived interest in this direction. Security notions such as misuse-resilience [PS16, GL15, HRRV15, RS06], robustness [ADL17, AFL+16, HKR15], multi-user security [BT16], reforgeability [FLLW17], and unverified plaintext release [ABL+14], as well as syntactic variants such as online operation [HRRV15] and variable stretch [HKR15, RVV16] have been studied in recent works.

Building on these developments, and using the indifferentiability framework of Maurer, Renner, and Holenstein [MRH04], we propose new definitions that bring a new perspective to the design of AEAD schemes. In place of focusing on specific property-based definitions, we formalize when an AEAD behaves like a *random* one. A central property of indifferentiable schemes is that they offer security with respect to a wide class of games. This class includes all the games above plus many others, including new unforeseen ones. Indifferentiability has been used to study the security of hash functions [CDMP05, BDPV08] and blockciphers [CPS08, HKT11, ABD+13, DSSL16], where constructions have been shown to behave like random oracles or ideal ciphers respectively. We investigate this question for authenticated encryption and ask if, and how efficiently, can indifferentiable AEAD schemes be built. Our contributions are as follows.

**Definitions:** We define ideal authenticated-encryption as one that is indifferentiable from a *random keyed injection*. This definition gives rise to a new model that is intermediate between the random-oracle and the ideal-cipher models. Accordingly, the random-injection model offers new efficiency and security trade-offs when compared to the ideal-cipher model. Along the way, we also recall the composition theorem and give an extension that includes game-based properties with *multiple* adversaries.

**Constructions:** We study a number of AEAD schemes from the view of indifferentiability. We obtain both positive and negative results. For most well-known constructions our results are negative. However, our main positive result is a Feistel construction that reduces the number of rounds from eight for ideal ciphers to only *three* for ideal keyed injections. This result improves the concrete parameters involved as well. We also give a transformation from offline to online ideal AEADs.

**Efficiency lower bounds:** Three rounds of Feistel are necessary to build injections. However, we prove a stronger result that lower bounds the number of primitive queries as a function of message blocks in *any* construction. This, in turn, shows that the *rate* of our construction is not too far off the optimal solution. For this result we combine two lower bound techniques, one for collision resistance and the other for pseudorandomness, which may be of independent interest.

The central motivation for the study of indifferentiable encryption is the *composition theorem*. We give a brief overview of indifferentiability before discussing their implications for our work.

### 1.1 Background on Indifferentiability

A common paradigm in the design of symmetric schemes is to start from some simple primitive, such as a public permutation or a compression function, and through some "mode of operation" build

a more complex scheme, such as a blockcipher or a variable-length hash function. The provable-security of such constructions has been analyzed mainly through two approaches. One is to formulate specific game-based properties, and then show that the construction satisfies them if its underlying primitives are secure. This methodology has been successfully applied to AEAD schemes. (See works cited in the opening paragraph of the paper.) Following this approach, higher-level protocols need to choose from a catalog of explicit properties offered by various AEAD schemes. For example, one would use an MRAE scheme whenever nonce-reuse cannot be excluded [PS16, GL15, HRRV15, RS06] or a key-dependent message (KDM) secure one when the scheme is required to securely encrypt its own keys [BK11, BRS03].

The seminal work of Maurer, Renner, and Holenstein (MRH) on the indifferentiability of random systems [MRH04] provides an alternative path to study the security of symmetric schemes. In this framework, a public primitive $f$ is available. The goal is to build another primitive $F$ from $f$ via a construction $C^f$. Indifferentiability formalizes a set of necessary and sufficient conditions for the construction $C^f$ to securely replace its ideal counterpart $F$ in a wide range of environments: there exists a *simulator* $S$, such that the systems $(C^f, f)$ and $(F, S^F)$ are indistinguishable, even when the distinguisher has access to $f$. Indeed, the composition theorem proved by MRH states that, if $C^f$ is indifferentiable from $F$, then $C^f$ can securely replace $F$ in *arbitrary* (single-stage) contexts.[4] Thus, proving that a construction $C$ is indifferentiable from an ideal object $F$ amounts to proving that $C^f$ *retains essentially all security properties implicit in* $F$. This approach has been successfully applied to the analysis of many symmetric cryptographic constructions in various ideal-primitive models; see, e.g., [CDMP05, BDPV08, HKT11, DSSL16, DSST17]. Our work is motivated by this composition property.

### 1.2 Motivation

Maurer, Renner, and Holenstein proposed indifferentiability as an alternative to the Universal Composability (UC) framework [Can01] for compositional reasoning in idealized models of computation such as the random-oracle (RO) and the ideal-cipher (IC) models. Indifferentiability permits finding constructions that can safely replace ideal primitives (e.g., the random oracle) in various schemes.

The UC framework provides another general composition theorem, which has motivated the study of many UC-secure cryptographic protocols. Küsters and Tuengerthal [KT09] considered UC-secure symmetric encryption and defined an ideal functionality on par with standard notions of symmetric encryption security. This, however, resulted in an intricate functionality definition that adds complexity to the analysis of higher-level protocols.

By adopting the indifferentiability framework for the study of AEADs, we follow an approach that has been successfully applied to the study of other *symmetric* primitives. As random oracles formalize the intuition that well-designed hash functions have random-looking outputs, ideal encryption formalizes random-looking ciphertexts subject to decryptability. This results in a simple and easy-to-use definition. We discuss the benefits of this approach next and give limitations and open problems at the end of the section.

[4] We recall this theorem and its proof. Consider a game $G$ that allows an adversary $A$ to attack $C^f$ with direct oracle access to $f$. Game $G$ controls the attack surface via $G^{C^f, A^f}$. Viewing the composition of the game $G$ and the adversary $A$ as an indifferentiability attacker, we may use the indifferentiability of $C^f$ to show that the analysis can be performed in game $G^{F, A^{S^F}}$ instead. By assumption, the simulator $S$ is able to produce $f$-values as if the (monolithic) object $F$ was constructed through $C^{S^F}$. Combining algorithms $A$ and $S$ into $B := A^S$ we arrive at game $G^{F, B^F}$. Therefore, using indifferentiability, the $G$-security of $C^f$ has been reduced to the $G$-security of $F$.

Once a primitive is standardized for general and widespread use, it is hard to predict in which environments it will be deployed, and which security properties may be intuitively expected from it. For example, consider a setting where a protocol designer follows the intuition that an AEAD scheme "essentially behaves randomly" and, while not knowing that AE security does not cover key-dependent message attacks [BK11, HK07, BRS03] (KDM), uses a standardized general-purpose scheme for disk encryption. A similar problem could occur in a cloud storage scenario when deduplication schemes [BKR13, ABM+13] are used. In other settings, a designer might create correlations among keys (as in 3GPP confidentiality and authenticity mechanisms) or allow an adversary to *tamper* parts of the key (e.g., set them to zero), expecting the underlying scheme to offer security against related-key attacks [IK04, BK03] (RKAs). There are also *chosen-key* settings where the adversary is allowed to entirely choose the keys. For example, certain MPC protocols [CO15] rely on AE schemes that need to be committing against malicious adversaries, which can choose all inputs and thus also the keys. This has lead to the formalizations of committing [GLR17] and key-robust [FOR17] authenticated encryption. When there is leakage, parts of the key and/or randomness might be revealed [PSV15, BMOS17]. All of these lie beyond standard notions of AE security, so the question is how should one deal with such a multitude of security properties.

One approach would be to formulate a new "super" notion that encompasses all features of the models above. This is clearly not practical. The model (and analyses using it) will be error-prone and, moreover, properties that have not yet been formalized will not be accounted for. Instead, and as mentioned above, we consider the following approach: a good AEAD scheme should behave like a random oracle, except that its ciphertexts are invertible. We formulate this in the language of indifferentiability, which results in a simple, unified, and easy to use definition. In indifferentiability the adversarial interfaces are *not* restricted to uniformly or independently chosen keys. All inputs are under the control of the adversary. This means that the security guarantees offered extend to notions that allow for tampering with keys or creation of dependencies among the inputs. Once indifferentiability is proved, security with respect to *all* these games, combinations thereof, and new yet unforeseen notions simultaneously follows as a corollary of the composition theorem.

Therefore one use-case for indifferentiable schemes would be to provision additional safeguards against *primitive misuse* in various deployment scenarios, such as general-purpose crypto libraries or standards, where the relevant security properties for target applications are complex or not known. We discussed some of these in the paragraph above. Protocol designers can rely on the intuition given by an ideal view of AEADs when integrating schemes into higher-level protocols, keeping game-based formulations implicit. Other applications include symbolic protocol analysis, where such idealizations are intrinsic [MW04] and security models where proof techniques such as programmability may be required [Unr12].

A CONCRETE EXAMPLE. In Facebook's message-franking protocol, an adversary attempts to compute a ciphertext that it can later open in two ways by revealing different keys, messages and header information. (Facebook sees one (harmless) message, whereas the receiver gets another (possibly abusive) message.) Grubbs, Lu, and Ristenpart [GLR17] formalize the security of such protocols and show that a standard AEAD can be used here, provided that it satisfies an additional security property called r-BIND [GLR17, Fig. 17 (left)].

One important feature of this definition is that it relies on a single-stage game in the sense of [RSS11]. The single-stage property immediately implies that any indifferentiable scheme is r-BIND if the ideal encryption scheme itself satisfies the r-BIND property. In contrast, not every AE-secure scheme is r-BIND secure [GLR17]. Interestingly, it is easy to see that the ideal encryption

scheme (a keyed random injection) indeed satisfies r-BIND and this is what, intuitively, the protocol designers seem to have assumed: that ciphertexts *look random* and thus collisions are hard to find, even if keys are adversarially chosen.

Indifferentiable AEADs therefore allow designers to rely on the above (arguably pragmatic) random-behavior intuition much in the same way as they do when using hash functions as random oracles. As the practicality of random oracles stems from their random output behavior (beyond PRF security or collision resistance) indifferentiable AEAD offers similar benefits: instead of focusing on a specific game-based property, it considers a fairly wide *class* of games for which the random behavior provably holds. Thus an indifferentiable AE can be used as a safety net to ensure any existing or future single-stage assumptions one may later need is satisfied (with the caveat of possibly weaker bounds). However, we note that for RO indifferentiability there is the additional motivation that a fair number of security proofs involving hash functions rely on modeling the hash as RO. Our work also unlocks the possibility to use the *full* power of random injections in a similar way (see [KPS13] and footnote 5).

To summarize, in the context of Facebook's protocol, if an indifferentiable scheme was used from the start, it would have automatically met the required binding property. The same holds for RKA security (in 3GPP), KDM security (in disk encryption), and other single-stage AEAD applications.

### 1.3 Contributions

Definitions. The MRH framework has been formulated with respect to a general class of random systems. We make this definition explicit for AEAD schemes by formulating an adequate *ideal reference object*. This object has been gradually emerging through the notion of a pseudorandom injection (PRI) in a number of works [RS06, HKR15, HRRV15, BMM+15], and has been used to study the security of offline and online AEADs [HKR15, HRRV15]. (We refer the reader to the work of Hoang, Reyhanitabar, Rogaway, and Vizár [HRRV15] for a review of the rich body of work in this area.) We lift these notions to the indifferentiability setting by introducing offline and online *random injections*, which may be also keyed or tweaked. As a result, we obtain a new idealized model of computation: the ideal-encryption (or ideal-injection) model, which is intermediate between the RO and IC models.

Analysis of known schemes. We examine generic and specific constructions of AEADs that appear in the literature in light of our new definition. Since indifferentiability implies security in the presence of nonce-misuse (MRAE) as well as its recent strengthening to variable ciphertext stretch, RAE security,[5] we rule out the indifferentiability of a number of (classical) schemes that do not achieve these levels of security. This includes OCB [RBBK01], CCM, GCM, and EAX [BRW04], and all but two of the third-round CAESAR candidates [Ber14]. The remaining two candidates, AEZ [HKR17] and DEOXYS-II [JNPS16], are also ruled out, but only using specific indifferentiability attacks.

We then turn our attention to generic composition [BN00, NRS14]. We study the well-known Encrypt-then-MAC and MAC-then-Encrypt constructions via the favored (A1–A8) and modified

[5] The notion of RAE security that we use in this paper deviates from the original notion proposed in [HKR15] in the sense that we do not consider the benign leakage of partial information during decryption. This is because all indifferentiable constructions must guarantee that, like the ideal object, decryption gives the stronger guarantee that $\perp$ is returned for all invalid ciphertexts.

(B1–B8) composition patterns of Namprempre, Rogaway and Shrimpton [NRS14]. These include Synthetic Initialization Vector (SIV) [RS06] as A4 and EAX [BRW04] as B1. To simplify and generalize the analysis, we start by presenting a template for generic composition, consisting of a preprocessing and a post-processing phase, that encompasses a number of schemes that we have found in the literature. We show that if there is an insufficient flow of information in a scheme—a notion that we formalize—differentiating attacks exist. Our attacks render all of these constructions except A8 differentiable. We also identify *key reuse* as a mechanism to foil some of these attacks, leaving the modifications of A2, A6, and A8 with key reuse as potential indifferentiability candidates.

In short, contrarily to our expectations based on known results for hash functions and permutations, we could not find a well-known AEAD construction that meets the stronger notion of indifferentiability. We stress that these findings do *not* contradict existing security claims. However, an indifferentiability attack can guide the search to find environments in which the scheme will not offer the expected levels of security. For example, the lack of a successful simulator for some of our differentiators stems from the fact that ciphertexts do not depend on all keying material, giving way to related-key attacks. In others, the attacks target intermediate values in a computation, and are reminiscent of padding oracles [Vau02]. Indeed, a scheme might be vulnerable to different types of differentiating attacks, leading to different "bad" environments. On the other hand, proving indifferentiability sets a lower bound for the complexity of any attack in any (single-stage) environment. For these reasons, and even though our results do not single out any of the CAESAR candidates as being better or worse than the others, we pose that our results are aligned with the fundamental goal of CAESAR and prior competitions such as AES and SHA-3, to "boost to the cryptographic research community's understanding" of the primitive [Ber14].

Building injections. Looking for a simple and provably indifferentiable AEAD scheme, we revisit the classical Encode-then-Encipher (EtE) transform [BR00]. Given expansion $\tau$, which indicates the required level of authenticity, EtE pads the input message with $0^\tau$ and enciphers it with a variable-input-length (VIL) blockcipher. Decryption checks the consistency of the padding after recovering the message. We show that EtE is indifferentiable from a random injection in the VIL ideal-cipher model for any (possibly small) value of $\tau$. The ideal cipher underlying EtE can be instantiated via the Feistel construction [CHK+16] in the random-oracle model or via the confusion-diffusion construction [DSSL16] in the random-permutation model. In a series of works, the number of rounds needed for indifferentiability of Feistel has been gradually reduced from 14 [HKT11, CHK+16] to 10 [DS15, DKT16] and recently to 8 [DS16]. Due to the existence of differentiators [CPS08, CHK+16], the number of rounds must be at least 6. For confusion-diffusion, 7 rounds are needed for good security bounds [DSSL16]. This renders the above approach to the design of random injections somewhat suboptimal in terms of the number of queries per message block to their underlying ideal primitives (i.e., their *rate*).

Our main positive result is the indifferentiability of *three*-round Feistel for large (but variable) expansion values $\tau$. Three rounds are also necessary, as we give a differentiator against the 2-round Feistel network for any $\tau$. In light of the above results, and state-of-the-art 2.5-round constructions such as AEZ, this is a surprisingly small price to pay to achieve indifferentiability. Our results, therefore, give some support to inclusion of redundancy for achieving authenticity (as opposed to generic composition). Furthermore, when using a blockcipher for encryption with redundancy, a significantly reduced number of rounds may suffice.

To simplify the analyses and focus on fundamental problem of constructing random injections, we first prove indifferentiability for a restricted class of differentiators that are bound to fixed (but arbitrary) key, nonce, associated data and expansion values. We then show how to extend these results to adversarially controlled values. We do this by proving a slightly more general result that identifies a set of sufficient conditions on the construction, a simulator, and the ideal objects that guarantee independence of executions, allowing a hybrid argument to go through. Along the way, we also formalize a folklore result on key extension via random oracles for constructions and reference objects that are structured.

THE SIMULATOR. Our main construction is an unbalanced 3-round Feistel network $\Phi_3$ with independent round functions where an input $X_1$ is encoded with redundancy as $(0^\tau, X_1)$ (see Figure 1).

The main task of our indifferentiability simulator is to consistently respond to round-function oracle queries that correspond to those that the construction makes for some (possibly unknown) input $X_1$. We show that with overwhelming probability the simulator can detect when consistency with the construction must be enforced; the remaining isolated queries can be simulated using random and independent values.

Take, for example, a differentiator that computes $(X_3', X_4') := \Phi_3(X_1)$ for some random $X_1$, then computes the corresponding round-function outputs $X_2 := F_1(X_1)$, $Y_2 := F_2(X_2)$, $Y_3 := F_3(X_1 \oplus Y_2)$, and finally checks if $(X_3', X_4') = (X_1 \oplus Y_2, X_2 \oplus Y_3)$.

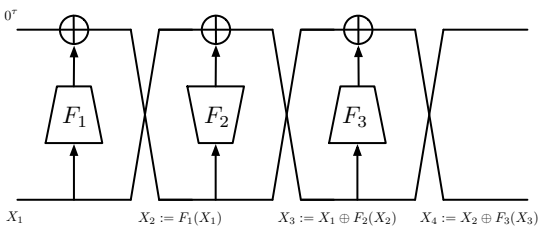


**Fig. 1.** Random injection from 3-round Feistel.

Note that these queries need *not* arrive in this particular order. Indeed, querying $F_1(X_1)$ first gives the simulator an advantage as it can preemptively complete this chain of queries and use its ideal injection to give consistent responses. A better (and essentially the only) alternative for the differentiator would be to check the consistency of outputs by going through the construction in the backward direction. We show, however, that whatever query strategy is adopted by the differentiator, the distribution of such chained queries in the real world takes the form

$$(X_1, X_2) \qquad (X_2, Y_2) \qquad (X_1 \oplus Y_2, Y_3) \qquad (X_1, (X_1 \oplus Y_2, X_2 \oplus Y_3))$$

$\Delta$corresponding, respectively, to $F_1$, $F_2$, $F_3$ and the construction. In the ideal world, the last component is fixed by the random injection. However, the simulator can still match the above by a change of variables, simulating them as

$$(X_1, Y_3 \oplus X_4) \qquad (Y_3 \oplus X_4, X_1 \oplus X_3) \qquad (X_3, Y_3) \qquad (X_1, (X_3, X_4))$$

for a random $Y_3$, and $(X_3, X_4)$ taken from the output of the random injection. For instance, in an inverse attack, once the $F_2$ and $F_3$ queries arrive, $(X_3, X_4)$ is fixed, and the simulator can use the inverse construction oracle to work out the corresponding $X_1$ and provide consistent answers.

A crucial part of this analysis hinges on the fact that the output of the first round function is directly fed as input to the second round function as a consequence of fixing parts of the input to $0^\tau$.[6] When launching an attack, the ability to cause collisions in internal wire values of Feistel typically allows an adversary to create input/output pairs that exhibit non-trivial relations in the

[6] Padding with $0^\tau$ has also been used by Kiltz, Pietrzak, and Szegedy [KPS13] in the context of building public-key signature schemes with message recovery and optimal overhead. The authors show that *4*-round Feistel is *publicly*

real world. From the simulator's perspective, such collisions would create ambiguity and prevent it from being able to check when consistency with the ideal object must be enforced. We show that for large $\tau$ the attacker would have no hope of finding collisions on the internal wires for distinct inputs in either direction.

As corollaries of our results (and the composition theorem) we obtain efficient and (simultaneously) RKA and KDM-secure offline (and, as we shall see, online) AEAD schemes in the random-permutation model under natural, yet practically relevant restrictions on these security models. For example, if the ideal AEAD $\mathcal{AE}$ is secure under encryptions of $\phi^{\mathcal{AE}}(K)$ for some oracle machine $\phi^{\mathcal{AE}}$, then so is an indifferentiable construction $C^{\pi}$ in presence of encryptions of $\phi^{C^{\pi}}(K)$, the restriction being that $\phi$ does not directly access $\pi$.

Bounds. Security bounds, including simulator query complexity, are important considerations for practice. Our bound for the Encode-then-Encipher construction is essentially tight. Our simulator for the 3-round Feistel construction has a quadratic query complexity and overall bounds are birthday-type. Improving these bounds, or proving lower bounds for them [DRST12], remain open for subsequent work.

Our construction of an ideal encryption scheme from a non-keyed ideal injection introduces an additional multiplicative factor related to the number of different ideal injection keys queried by the differentiator, resulting from a hybrid argument over keys. Furthermore, the number of ideal injection keys used in the construction is bound to the number of encryption and decryption operations that are carried out. This means that the overall bound for our authenticated encryption construction includes a multiplicative factor of $q^3$ (see Section 5.3).

We note that the concrete constructions that we analyze may satisfy (R)AE, RKA or KDM security with improved bounds (via game-specific security analyses), while remaining compatible with the single proof and bound that we present for all single-stage games.

Online AEADs. We give simple solutions to the problem of constructing an indifferentiable *segment-oriented* online AEAD scheme from an offline AEAD.

Following [HRRV15], we define ideal online AEAD scheme via initialization, next-segment encryption/decryption, and last-segment encryption/decryption procedures. The difference between next-segment and last-segment operations is that the former propagates *state values*, whereas the latter does not. Since a differentiator typically has access to *all* interfaces of a system, the state values become under its control/view. For this we restrict the state size of the ideal object to be finite and hence definitionally deviate from [HRRV15] in this aspect. Therefore our constructions have the extra security property that the state value hides all information about past segments.

The most natural way to construct an ideal online AEAD would be to *chain* encryptions of the segments by *tweaking* the underlying encryption primitive with the input history so far. This would ensure independence across segments for different keys, nonces and prefixes, which is also the intuition underlying the **CHAIN** transform of HRRV [HRRV15, Figure 8]. We show, however, that standard XOR-based tweaking techniques [HRRV15, BMM+15] are not sound in the indifferentiability setting and, in particular, we present a differentiating attack on **CHAIN**.

In building an indifferentiable online AEAD scheme, we put use to the parallel composition properties enjoyed by indifferentiable constructions [MRH04, DGHM13]. By decomposing the ideal

indifferentiable with tight security bounds. In public indifferentiability the simulator sees all the queries of the differentiator to the construction oracle. This level of indifferentiability, however, is not sufficient in the AEAD setting as it does not even imply CPA security. This is due to the fact that the adversary $B := A^S$ contains the simulator, and hence it gets to see the full inputs (including the key) to the construction.

object for online AEAD into simpler ones, we obtain a construction similar to **CHAIN** essentially for free, where a random oracle is used to prepare the state for the next segment. This hashing operation, however, comes at an extra cost proportional to the total size of key, nonce, associated data, and message. By unwrapping the underlying key computation for the offline AEAD, we can reduce the overhead to one that is proportional to the length of the message only. We call the resulting scheme **HashCHAIN**. Via optimizations specific to 3-round Feistel, we are able to reduce the overheads further to a *constant* number of hashes per segment. (We note, however, that other constructions might also be amenable to benefit from this technique.) Furthermore, our construction can also be optimized using a parallel and indifferentiable tree mode such as MD6 [DRRS09].

Lower bounds. The indifferentiability of Sponge [BDPV08] allows us to instantiate the round functions in 3-round Feistel with this construction and derive a random injection in the random-permutation model.[7] This construction requires roughly $3w$ calls to its underlying (one-block) permutation, where $w$ is the total number of input blocks. This is slightly higher than $2.5w$ for AEZ (which shares some of its design principles with us, but does not offer indifferentiability). This leads us to ask whether or not an indifferentiable construction with rate *less than* 3 is achievable. Our second main result is a lower bound showing the impossibility of *any* such construction with rate (strictly) less than 2. To prove this lower bound, we combine negative results for constructions of collision-resistant hash functions [Sta08, BCS05] and pseudorandom number generators by Gennaro and Trevisan [GT00], and put critical use to the existence of an indifferentiability simulator. To the best of our knowledge, this is the first impossibility result that exploits indifferentiability.

Roughly speaking, we prove the lower bound as follows. Take any indifferentiable random *function* $C^{\pi} : \{0,1\}^{wn} \longrightarrow \{0,1\}^{wn}$ from a permutation $\pi : \{0,1\}^{n} \longrightarrow \{0,1\}^{n}$ in $q$ queries. We "slice" the construction into $q$ components that process incoming inputs (from the previous stage), make a single $\pi$-query, and return an output. Since the input space of $C^{\pi}$ is of size $2^{wn}$, we can iteratively and in a *small* number of $\pi$-queries find a subset of inputs of size at least $2^{n}$ such that the first $w-1$ slices *always* query $\pi$ at a fixed set of points on these inputs. This means that the first $w-1$ queries to $\pi$ are essentially made redundant for inputs that are restricted to this subset. Let us now consider the rest of the slices consisting of $q-(w-1)$ further queries to $\pi$. By viewing the corresponding $\pi$-responses as parts of a seed, we can show that this truncated construction is a secure PRG that makes no $\pi$ calls and has a seed length that is shorter than its output length. However, no such PRG can information-theoretically exist. This means we must have that $n+(q-(w-1))n \geq wn$, which translates to $q \geq 2w-2$. To make this argument precise, we have to ensure that hardwiring of various values does not result in PRG insecurity. For this, we rely on the indifferentiability of the construction. Our lower bound immediately extends to random injections as we can indifferentiably reduce a random function to a random injection in a *single* query by simply chopping off a sufficiently large number of bits from the outputs of the injection. We do not have constructions or lower bounds for indifferentiable schemes with a rate $r \in [2,3)$. We leave bridging this gap for future work.

### 1.4 Limitations and future work

As clarified by Ristenpart, Shacham, and Shrimpton [RSS11], the indifferentiability composition theorem has limitations. It does not necessarily apply to multi-stage games where multiple ad-

[7] The round-function for the intermediate round can actually be fully parallelized to cost a single parallel call to the underlying primitive, as it is expanding a small seed to a large $w$-block random-looking string.

versaries cannot be collapsed into a single central adversary. Such games do exist, for example when an adversary's outputs are constrained in their format/lengths. Accordingly, indifferentiable AEAD schemes come with similar limitations. Moreover, other types of attacks, such as side-channel analysis, algorithm substitution, backdoors, etc. lie outside the model.

Indifferentiability typically operates in an ideal model of computation. Although this methodology has been successful in the analysis of practical schemes, it leaves open the question of standard-model security. However, it does not exclude a "best of the two worlds" construction, which is both indifferentiable and, for example, is RAE secure under a standard-model assumption. For example, chop-Merkle–Damgård [CDMP05] can be proven both indifferentiable from a random oracle *and* collision resistant in the standard model under standard assumptions. We leave exploring this for future work.

## 2 Basic Definitions

We let $\mathbb{N}$ denote the set of non-negative integers, including zero, and $\{0,1\}^*$ denote the set of all finite-length bit strings, including the empty string $\varepsilon$. (Note that $\{0,1\}^0 = \{\varepsilon\}$.) For two bit strings $X$ and $Y$, $X|Y$ denotes string concatenation and $(X,Y)$ denotes a uniquely decodable encoding of $X$ and $Y$. The length of a string $X$ is denoted by $|X|$.

### 2.1 Games

An $n$-adversary game $\mathbf{G}$ is a Turing machine $\mathbf{G}^{\Sigma,\mathscr{A}_1,\ldots,\mathscr{A}_n}$ where $\Sigma$ is a system (or functionality) and $\mathscr{A}_i$ are adversarial procedures that can keep full local state but may only communicate with each other through $\mathbf{G}$. We say an $n$-adversary game $\mathbf{G}_n$ is reducible to an $m$-adversary game if there is a $\mathbf{G}_m$ such that for any $(\mathscr{A}_1,\ldots,\mathscr{A}_n)$ there are $(\mathscr{A}'_1,\ldots,\mathscr{A}'_m)$ such that for all $\Sigma$ we have that $\mathbf{G}_n^{\Sigma,\mathscr{A}_1,\ldots,\mathscr{A}_n} = \mathbf{G}_m^{\Sigma,\mathscr{A}'_1,\ldots,\mathscr{A}'_m}$. Two games are equivalent if they are reducible in both directions. An $n$-adversary game is called $n$-stage [RSS11] if it is not equivalent to any $m$-adversary game with $m < n$. Any single-stage game $\mathbf{G}^{\Sigma,\mathscr{A}}$ can be also written as $\bar{\mathscr{A}}^{\overline{\mathbf{G}}^{\Sigma}}$ for some oracle machine $\overline{\mathbf{G}}$ and a class of adversarial procedures $\bar{\mathscr{A}}$ compatible with a modified syntax in which the game is called as an oracle.

### 2.2 Offline reference objects

Underlying the security definition for a cryptographic primitive there often lies an *ideal* primitive that is used as a *reference object* to formalize security. For instance, the security of PRFs is defined with respect to a random oracle, PRPs with respect to an ideal cipher, and as mentioned above, AEADs with respect to a random *injection*. Given the syntax and the correctness condition of a cryptographic primitive, we will define its ideal counterpart as the uniform distribution over the set of all functions that meet these syntactic and correctness requirements (but without any efficiency requirements). We start by formalizing a general class of ideal functions—that may be keyed, admit auxiliary data (such as nonces or authenticated data), or allow for variable-length outputs—and derive distributions of interest to us by imposing structural restrictions over the class of considered functions. This definitional approach has also been used, for example, in [BBT16, Section 6].

Ideal functions. A variable-output-length (VOL) function $\mathcal{F}$ with auxiliary input has signature

$$\mathcal{F} : \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R} ,$$

where $\mathcal{A}$ is the auxiliary-input space, $\mathcal{M}$ is the message space, $\mathcal{X} \subseteq \mathbb{N}$ is the expansion space, and $\mathcal{R}$ is the range. We let $\mathrm{Fun}[\mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}]$ be the set of all such functions satisfying

$$\forall (A, M, \tau) \in \mathcal{A} \times \mathcal{M} \times \mathcal{X} : |\mathcal{F}(A, M, \tau)| = \tau \ .$$

We endow the above set with the uniform distribution and denote the action of sampling a uniform function $\mathcal{F}$ via $\mathcal{F} \twoheadleftarrow \mathrm{Fun}[\mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}]$. To ease notation, given a function $\mathcal{F}$, we define $\mathrm{Fun}[\mathcal{F}]$ to be the set of all functions with signature identical to that of $\mathcal{F}$. Granting oracle access to $\mathcal{F}$ to all parties (honest or otherwise) results in an ideal model of computation.

Injections. We define $\mathrm{Inj}[\mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}]$ to be the set of all expanding functions that are injective on $\mathcal{M}$:

$$\forall (A, M, \tau), (A, M', \tau) \in \mathcal{A} \times \mathcal{M} \times \mathcal{X} : M \neq M' \implies \mathcal{F}(A, M, \tau) \neq \mathcal{F}(A, M', \tau) \ ,$$

and satisfy the length restriction

$$\forall (A, M, \tau) \in \mathcal{A} \times \mathcal{M} \times \mathcal{X} : |\mathcal{F}(A, M, \tau)| = |M| + \tau \ .$$

Each injective function defines a unique inverse function $\mathcal{F}^{-}$ that maps $(A, C, \tau)$ to either a unique $M$ if and only if $C$ is within the range of $\mathcal{F}(A, \cdot, \tau)$, or to $\perp$ otherwise. (Such functions are therefore *tidy* in the sense of [NRS14].) This gives rise to a *strong* induced model for injections where oracle access is extended to include $\mathcal{F}^{-}$, which we always assume to be the case when working with injections.

Observe that when $k = 0$ the key space contains the single $\varepsilon$ key and we recover unkeyed functions. For the sake of compactness we use the following abbreviations:

$$\mathrm{Fun}[n, m] := \mathrm{Fun}[\{0,1\}^0 \times \{0,1\}^n \times \{m\} \longrightarrow \{0,1\}^m] \ ,$$
$$\mathrm{Perm}[n] := \mathrm{Inj}[\{0,1\}^0 \times \{0,1\}^n \times \{0\} \longrightarrow \{0,1\}^n] \ .$$

Lazy samplers. Various ideal objects (such as random oracles) often appear as algorithmic procedures that lazily sample function values at each point. For instance, it is well known that lazy samplers for random functions and permutations with arbitrary domain and range exist. These procedures can be extended to admit auxiliary data and respect either of our length-expansion requirements above. Furthermore, given a list $L$ of input-output pairs, these samplers can be modified to sample a function that is also consistent with the points defined in $L$ (i.e., the conditional distribution given $L$ is also samplable). We denote the lazy sampler for random oracles with $(Y; L) \twoheadleftarrow \mathrm{LazyRO}(A, X, \tau; L)$ and that for ideal ciphers with $(Y; L) \twoheadleftarrow \mathrm{LazyIC}^{\pm}(A, X; L)$. The case of random injection is less well known, but such a procedure appears in [RS06, Figure 6]. We denote this sampler with $(Y; L) \twoheadleftarrow \mathrm{LazyRI}^{\pm}(A, X, \tau; L)$.

### 2.3 Authenticated-Encryption with Associated-Data

We follow [HRRV15] in formalizing the syntax of (offline) AEAD schemes.[8] We allow for arbitrary plaintexts and associated data, and also include an explicit expansion parameter $\tau$ specifying the level of authenticity that is required. Associated data may contain information that may be needed

[8] When referring to an AEAD without specifying its type, we mean an *offline* AEAD.

in the clear by a higher-level protocol (such as routing information) that nevertheless should be authentic. We also only allow for public nonces as the benefits of the AE5 syntax with a private nonce are unclear [NRS13].

SYNTAX AND CORRECTNESS. An AEAD scheme is a triple of algorithms $\Pi := (\mathcal{K}, \mathcal{AE}, \mathcal{AD})$ where: (1) $\mathcal{K}$ is the randomized key-generation algorithm which returns a key $K$. This algorithm defines a non-empty set, the support of $\mathcal{K}$, and an associated distribution on it. Slightly abusing notation, we denote all these by $\mathcal{K}$. (2) $\mathcal{AE}$ is the deterministic encryption algorithm with signature $\mathcal{AE} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \{0,1\}^* \times \mathcal{X} \longrightarrow \{0,1\}^*$. Here $\mathcal{N} \subseteq \{0,1\}^*$ is the nonce space, $\mathcal{H} \subseteq \{0,1\}^*$ is the associated data space, and $\mathcal{X} \subseteq \mathbb{N}$ is the set of allowed expansion values. We typically have that $\mathcal{K} = \{0,1\}^k$, $\mathcal{N} = \{0,1\}^n$ for $k, n \in \mathbb{N}$, $\mathcal{H} = \{0,1\}^*$, and the expansion space contains a single value. (3) $\mathcal{AD}$ is the deterministic decryption algorithm with signature $\mathcal{AD} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \{0,1\}^* \times \mathcal{X} \longrightarrow \{0,1\}^* \cup \{\perp\}$. As usual we demand that $\mathcal{AD}(K, N, A, \mathcal{AE}(K, N, A, M, \tau), \tau) = M$ for all inputs from the appropriate spaces. We also impose the ciphertext expansion restriction that for all inputs from the appropriate spaces $|\mathcal{AE}(K, N, A, M, \tau)| - |M| = \tau$.

IDEAL AEAD. An ideal AEAD is an injection with signature $(\mathcal{K} \times \mathcal{N} \times \mathcal{H}) \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C}$ and satisfying the ciphertext-expansion restriction. Therefore an ideal AEAD is a random injection in $\mathrm{Inj}[(\mathcal{K} \times \mathcal{N} \times \mathcal{H}) \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C}]$. Given a concrete AEAD scheme $\Pi$ with signature $\mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C}$ we associate the space $\mathrm{AE}[\Pi] := \mathrm{Inj}[(\mathcal{K} \times \mathcal{N} \times \mathcal{H}) \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C}]$ to it.[9]

NAMING CONVENTIONS. When referring to AEAD schemes we use $(\mathcal{AE}, \mathcal{AD})$ instead of $(\mathcal{F}, \mathcal{F}^-)$. When the associated-data space is empty, we use $(\mathcal{E}, \mathcal{D})$ for (encryption without associated data), when the nonce space is also empty we use $(\mathcal{F}, \mathcal{F}^-)$ (for keyed injection), when $\tau = 0$ as well we use $(\mathsf{E}, \mathsf{E}^-)$ (for blockcipher), and if these are also unkeyed we use $(\rho, \rho^-)$ and $(\pi, \pi^-)$ respectively. For a random function (without inverse) we use $\mathcal{H}$.

RAE SECURITY. Robust AE (RAE) security [HKR15, HRRV15, BMM+15] requires that an AEAD scheme behaves indistinguishably from an ideal AEAD under a random key. Formally, for scheme $\Pi = (\mathcal{K}, \mathcal{AE}, \mathcal{AD})$ and adversary $\mathscr{A}$ we define

$$\mathbf{Adv}^{\mathrm{rae}}_{\Pi}(\mathscr{A}) := \Pr\left[\mathbf{RAE\text{-}Real}^{\mathscr{A}}_{\Pi}\right] - \Pr\left[\mathbf{RAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}\right] ,$$

where games $\mathbf{RAE\text{-}Real}^{\mathscr{A}}_{\Pi}$ and $\mathbf{RAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}$ are defined in Figure 2. Informally, we say $\Pi$ is RAE secure if $\mathbf{Adv}^{\mathrm{rae}}_{\Pi}(\mathscr{A})$ is "small" for any "reasonable" $\mathscr{A}$. Misuse-resilient AE (MRAE) security [RS06] weakens RAE security by constraining the adversary to a fixed and sufficiently large value of expansion $\tau$. AE security [Rog02] weakens MRAE security and requires that the adversary does not repeat nonces in its queries to either oracle. These definitions lift to idealized models of computation where, for example, access to an ideal injection in both the forward and backward directions is provided.

The following proposition formalizes the intuition that the ideal AEAD, that is the trivial AEAD scheme in the ideal AEAD model, is indeed RAE secure. This fact will be used when studying the relation between indifferentiability and RAE security.

**Proposition 1 (Ideal AEAD is RAE secure).** *For any $q$-query adversary $\mathscr{A}$ attacking the trivial ideal AEAD $\Pi$ in the ideal AEAD model we have that $\mathbf{Adv}^{\mathrm{rae}}_{\Pi}(\mathscr{A}) \leq q/2^k$.*

[9] We do not idealize the key-generation procedure. This is compatible with the approach adopted in the ideal-cipher model where the key-generation algorithm simply returns a $k$-bit prefix of its random coins: the addition of oracle access to the identity function does not change the model.

| GAME **RAE-Real**$^{\mathscr{A}}_{\Pi}$ | PROC. ENC$(N, A, M, \tau)$ | GAME **RAE-Ideal**$^{\mathscr{A}}_{\Pi}$ | PROC. ENC$(N, A, M, \tau)$ |
|---|---|---|---|
| $K \twoheadleftarrow \mathcal{K}$ | return $\mathcal{AE}(K, N, A, M, \tau)$ | $(\mathcal{AE}', \mathcal{AD}') \twoheadleftarrow \mathrm{AE}[\Pi]$ | return $\mathcal{AE}'(K, N, A, M, \tau)$ |
| $b \twoheadleftarrow \mathscr{A}^{\mathrm{ENC},\mathrm{DEC}}$ | | $K \twoheadleftarrow \mathcal{K}$ | |
| return $b$ | PROC. DEC$(N, A, C, \tau)$ | $b \twoheadleftarrow \mathscr{A}^{\mathrm{ENC},\mathrm{DEC}}$ | PROC. DEC$(N, A, C, \tau)$ |
| | return $\mathcal{AD}(K, N, A, C, \tau)$ | return $b$ | return $\mathcal{AD}'(K, N, A, C, \tau)$ |

**Fig. 2.** Games defining RAE security. The adversary queries its oracles on inputs that belong to appropriate spaces.

*Proof.* The adversary in the real and ideal worlds has, respectively, access to

$$(\mathcal{AE}(\cdots), \mathcal{AD}(\cdots), \mathcal{AE}(K, \cdots), \mathcal{AD}(K, \cdots)) \quad \text{and} \quad (\mathcal{AE}(\cdots), \mathcal{AD}(\cdots), \mathcal{AE}'(K, \cdots), \mathcal{AD}'(K, \cdots)) .$$

Here $(\mathcal{AE}, \mathcal{AD})$ and $(\mathcal{AE}', \mathcal{AD}')$ are both ideal. These worlds are identical until $\mathscr{A}$ queries the second set of oracles with the key $K$. However, the probability of this event in the second environment over its $q$ queries is at most $q/2^k$ as the two oracles are independent and $(\mathcal{AE}', \mathcal{AD}')$ can be implemented independently of $K$. □

## 3 AEAD Indifferentiability

The indifferentiability framework of Maurer, Renner, and Holenstein (MRH) [MRH04] formalizes a set of necessary and sufficient conditions for one system to securely replace another in a wide class of environments. This framework has been successfully used to justify the structural soundness of a number of cryptographic constructions, including hash functions [CDMP05, DRS09], blockciphers [ABD+13, CHK+16, DSSL16], and domain extenders for them [CDMS10]. The indifferentiability framework is formulated with respect to general systems. When the ideal AEAD object defined in Section 2.3 is used, a notion of indifferentiability for AEAD schemes emerges. In this section, we recall indifferentiability of systems and make it explicit for AEAD schemes. We will then discuss some of its implications that motivate our work.

### 3.1 Definition

A random system or functionality $\Sigma := (\Sigma.\mathrm{hon}, \Sigma.\mathrm{adv})$ is accessible via two interfaces $\Sigma.\mathrm{hon}$ and $\Sigma.\mathrm{adv}$. Here, $\Sigma.\mathrm{hon}$ provides a public interface through which the system can be accessed. $\Sigma.\mathrm{adv}$ corresponds to a (possibly extended) interface that models adversarial access to the inner workings of the system, which may be exploited during an attack on constructions. A system typically implements some ideal object $\mathcal{F}$, or it is itself a construction $\mathrm{C}^{\mathcal{F}'}$ relying on some underlying (lower-level) ideal object $\mathcal{F}'$.

INDIFFERENTIABILITY [MRH04]. Let $\Sigma_1$ and $\Sigma_2$ be two systems and $\mathcal{S}$ be an algorithm called the simulator. The (strong) indifferentiability advantage of a (possibly unbounded) differentiator $\mathscr{D}$ against $(\Sigma_1, \Sigma_2)$ with respect to $\mathcal{S}$ is

$$\mathbf{Adv}^{\mathrm{indiff}}_{\Sigma_1,\Sigma_2,\mathcal{S}}(\mathscr{D}) := \Pr\left[\mathbf{Diff\text{-}Real}^{\mathscr{D}}_{\Sigma_1}\right] - \Pr\left[\mathbf{Diff\text{-}Ideal}^{\mathscr{D}}_{\Sigma_2,\mathcal{S}}\right] ,$$

where games $\mathbf{Diff\text{-}Real}^{\mathscr{D}}_{\Sigma_1}$ and $\mathbf{Diff\text{-}Ideal}^{\mathscr{D}}_{\Sigma_2,\mathcal{S}}$ are defined in Figure 3. Informally, we call $\Sigma_1$ indifferentiable from $\Sigma_2$ if, for an "efficient" $\mathcal{S}$, the advantage above is "small" for all "reasonable" $\mathscr{D}$.

| $\underline{\text{GAME } \textbf{Diff-Real}^{\mathscr{D}}_{\Sigma_1}}$ | $\underline{\text{PROC. CONST}(X)}$ | $\underline{\text{GAME } \textbf{Diff-Ideal}^{\mathscr{D}}_{\Sigma_2,\mathcal{S}}}$ | $\underline{\text{PROC. CONST}(X)}$ |
|---|---|---|---|
| $b \twoheadleftarrow \mathscr{D}^{\text{CONST},\text{PRIM}}$ | return $\Sigma_1.\text{hon}(X)$ | $b \twoheadleftarrow \mathscr{D}^{\text{CONST},\text{PRIM}}$ | return $\Sigma_2.\text{hon}(X)$ |
| return $b$ | $\underline{\text{PROC. PRIM}(X)}$ | return $b$ | $\underline{\text{PROC. PRIM}(X)}$ |
| | return $\Sigma_1.\text{adv}(X)$ | | return $\mathcal{S}^{\Sigma_2.\text{adv}}(X)$ |

**Fig. 3.** Games defining the indifferentiability of two systems.

In the rest of the paper we consider a specific application of this definition to two systems with interfaces

$$(\Sigma_1.\text{hon}(X), \Sigma_1.\text{adv}(x)) := (\text{C}^{\mathcal{F}_1}(X), \mathcal{F}_1(x)) \quad \text{and} \quad (\Sigma_2.\text{hon}(X), \Sigma_2.\text{adv}(x)) := (\mathcal{F}_2(X), \mathcal{F}_2(x)) \ ,$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two ideal cryptographic objects sampled from their associated distributions and $\text{C}^{\mathcal{F}_1}$ is a construction of $\mathcal{F}_2$ from $\mathcal{F}_1$. To ease notation, we denote the advantage function by $\mathbf{Adv}^{\text{indiff}}_{\text{C},\mathcal{S}}(\mathscr{D})$ when $\mathcal{F}_1$ and $\mathcal{F}_2$ are clear from context. Typically $\mathcal{F}_2$ will be an ideal AEAD and $\mathcal{F}_1$ a random oracle or an ideal cipher.

### 3.2 Consequences

MRH [MRH04] prove the following composition theorem for indifferentiable systems. Here we state a game-based formulation from [RSS11].

**Theorem 1 (Indifferentiability composition).** *Let $\Sigma_1 := (\text{C}^{\mathcal{F}_1}, \mathcal{F}_1)$ and $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ be two indifferentiable systems with simulator $\mathcal{S}$. Let $\mathbf{G}$ be a single-stage game. Then for any adversary $\mathscr{A}$ there exist an adversary $\mathscr{B}$ and a differentiator $\mathscr{D}$ such that*

$$\Pr\left[\mathbf{G}^{\text{C}^{\mathcal{F}_1},\mathscr{A}^{\mathcal{F}_1}}\right] \leq \Pr\left[\mathbf{G}^{\mathcal{F}_2,\mathscr{B}^{\mathcal{F}_2}}\right] + \mathbf{Adv}^{\text{indiff}}_{\text{C},\mathcal{S}}(\mathscr{D}) \ .$$

*Proof.* The proof is simple enough to be presented here:

$$\Pr\left[\mathbf{G}^{\text{C}^{\mathcal{F}_1},\mathscr{A}^{\mathcal{F}_1}}\right] \approx \Pr\left[\mathbf{G}^{\mathcal{F}_2,\mathscr{A}^{\mathcal{S}^{\mathcal{F}_2}}}\right] = \Pr\left[\mathbf{G}^{\mathcal{F}_2,\mathscr{B}^{\mathcal{F}_2}}\right] \ ,$$

where the first transition follows from indifferentiability by viewing the composition of $\mathbf{G}$ and $\mathscr{A}$ as a differentiator $\mathscr{D}$, and the second from absorption of $\mathcal{S}$ into $\mathscr{A}$ as adversary $\mathscr{B}$. □

As discussed in [RSS11], the above composition does not necessarily extend to multi-stage games since the simulator often needs to keep *local* state in order to guarantee consistency. However, some (seemingly) multi-stage games can be written as equivalent single-stage games (see Section 2 for a definition of game equivalence). Indeed, any $n$-adversary game where only one adversary can call the primitive directly and the rest call it indirectly *via the construction* can be written as a single-stage game as the game itself has access to the construction. We summarize this observation in the following theorem, which generalizes a result for related-key security in [FP15].

**Theorem 2.** *Let $\Sigma_1 := (\text{C}^{\mathcal{F}_1}, \mathcal{F}_1)$ and $\Sigma_2 := (\mathcal{F}_2, \mathcal{F}_2)$ be two indifferentiable systems with simulator $\mathcal{S}$. Let $\mathbf{G}$ be an $n$-adversary game and $\mathscr{A} := (\mathscr{A}_1, \ldots, \mathscr{A}_n)$ be an $n$-tuple of adversaries where $\mathscr{A}_1$ can access $\mathcal{F}_1$ but $\mathscr{A}_i$ for $i > 1$ can only access $\text{C}^{\mathcal{F}_1}$. Then there is an $n$-adversary $\mathscr{B}$ and a differentiator $\mathscr{D}$ such that*

$$\Pr\left[\mathbf{G}^{\text{C}^{\mathcal{F}_1},\mathscr{A}^{\mathcal{F}_1},\mathscr{A}_2^{\text{C}^{\mathcal{F}_1}},\ldots,\mathscr{A}_n^{\text{C}^{\mathcal{F}_1}}}\right] \leq \Pr\left[\mathbf{G}^{\mathcal{F}_2,\mathscr{B}_1^{\mathcal{F}_2},\mathscr{B}_2^{\mathcal{F}_2},\ldots,\mathscr{B}_n^{\mathcal{F}_2}}\right] + \mathbf{Adv}^{\text{indiff}}_{\text{C},\mathcal{S}}(\mathscr{D}) \ .$$

Remark 1. There is a strong practical motivation for the restriction imposed on the class of games above. Consider, for example, security against related-key attacks (RKAs) where the related-key deriving (RKD) function $\phi^{\mathcal{F}_1}$ may depend on the ideal primitive [AFPW11]. The RKA game is *not* known to be equivalent to a single-stage game. The authors in [FP15] consider a restricted form of this game where dependence of $\phi$ on the ideal primitive $\mathcal{F}_1$ is constrained to be through the construction $\mathrm{C}^{\mathcal{F}_1}$ only. In other words, an RKD function takes the form $\phi^{\mathrm{C}^{\mathcal{F}_1}}$ rather than $\phi^{\mathcal{F}_1}$. When comparing the RKA security of a construction $\mathrm{C}^{\mathcal{F}_1}$ to the RKA security of its ideal counterpart, one would expect the set of RKD functions from which $\phi$ is drawn in two games to be syntactically fixed and hence comparable. Since no underlying ideal primitive for $\mathcal{F}_2$ exists, RKD functions take the form $\phi^{\mathcal{F}_2}$ and hence it is natural to consider RKD functions of the form $\phi^{\mathrm{C}^{\mathcal{F}_1}}$ with respect to $\mathrm{C}^{\mathcal{F}_1}$. The same line of reasoning shows that an indifferentiable construction would resist key-dependent message (KDM) attacks for key-dependent deriving functions that depend on the underlying ideal primitive via the construction only. Other (multi-stage) security notions that have a practically relevant single-stage formulation include security against bad-randomness attacks, where malicious random coins are computed using the construction, and leakage-resilient encryption where leakage functions may rely on the construction. Therefore, from a practical point of view, composition extends well beyond 1-adversary games.

Remark 2. Theorem 1 only reduces the security of one system to that of another, leaving an overall security statement conditional on the latter. For instance, using the theorem one can deduce the RKA (resp., KDM or leakage-resilient) security of an indifferentiable construction $\mathrm{C}^{\mathcal{F}_1}$ of $\mathcal{F}_2$ *if* $\mathcal{F}_2$ itself can be proven to be RKA (resp., KDM or leakage-resilient) secure. We have seen an example of the latter in Proposition 1, where the ideal AEAD scheme is shown to be RAE secure. Hence Theorem 1 and Proposition 1 immediately allow us to deduce that an indifferentiable AEAD construction $\mathrm{C}^{\mathcal{F}_1}$ will be RAE secure in the idealized model of computation induced by its underlying ideal primitive $\mathcal{F}_1$. Analogous propositions for RKA, KDM, leakage resilience of the ideal AEAD scheme (for quantified classes of related-key deriving functions, key-dependent deriving, and leakage functions) can be formulated. These in turn imply that an indifferentiable AEAD scheme will also resist strong forms of related-key, KDM, and leakage attacks. We leave a formal treatment to future work.

Public indifferentiability. Public indifferentiability is a (substantial) weakening of standard indifferentiability where the simulator gets to see all the queries that the differentiator makes to the construction oracle(s). This notion guarantees composition in games where all inputs to the construction are known to the adversary. Collision resistance would be an example of such a game. In the context of authenticated encryption, however, public indifferentiability is of limited use. In particular it does not imply AE security (or even one-wayness) as the encryption key is kept outside the view of the adversary in these security games.

## 4 Differentiators

Having defined AEAD indifferentiability, we ask whether or not (plausibly) indifferentiable constructions of AEAD schemes in the literature exist. In this section we present a number of generic and specific attacks that essentially rule out the indifferentiability of many constructions that we have found in the literature.[10] We emphasize that existing schemes were not designed with the goal

[10] Our treatment is not (and cannot be) exhaustive and hence we only focus on a selection of well-known schemes.

| Algo. $\mathcal{AE}(K, N, A, M, \tau)$ | Algo. $\mathcal{AD}(K, N, A, C, \tau)$ |
|---|---|
| $(est_0, est_1) \leftarrow \mathcal{I}_e(K, N, A, M, \tau)$ | $(dst_0, dst_1) \leftarrow \mathcal{I}_d(K, N, A, C, \tau)$ |
| $(K', N', M', \tau') \leftarrow \mathcal{E}_0^{\mathcal{H}}(est_0)$ | $(K', N', C', \tau') \leftarrow \mathcal{D}_0^{\mathcal{H}}(dst_0)$ |
| $C' \leftarrow \mathcal{E}(K', N', \varepsilon, M', \tau')$ | $M' \leftarrow \mathcal{D}(K', N', \varepsilon, C', \tau')$ |
| $C \leftarrow \mathcal{E}_1^{\mathcal{H}}(C', est_1)$ | $M \leftarrow \mathcal{D}_1^{\mathcal{H}}(M', dst_1)$ |
| return $C$ | return $M$ |

**Fig. 4.** A template for a typical generically composed AEAD scheme $(\mathcal{AE}, \mathcal{AD})$ from an encryption scheme (without associated data) $(\mathcal{E}, \mathcal{D})$ and a hash function $\mathcal{H}$.

of meeting indifferentiability, and our attacks do not contradict any security claims made under the standard RAE, MRAE, or AE models. Indeed, many AEAD schemes are designed with the goal of maximizing efficiency, forsaking stronger security goals such as misuse resilience or robustness.

### 4.1 Generic composition

Any construction that is not (M)RAE secure (in the sense of [RS06, HKR15]) can be immediately excluded as one that is indifferentiable: the ideal AEAD is RAE secure (Proposition 1), furthermore RAE is a single-stage game and hence implied by indifferentiability (Theorem 1). This simple observation rules out the indifferentiability of a number notable AEAD schemes such as OCB [RBBK01], CCM, GCM, EAX [BRW04], and many others. The MRAE insecurity of these schemes are discussed in the respective works.

RAE insecurity can be used to also rule out the indifferentiability of some generic AEAD constructions. In this section, we present a more general result by giving differentiators against a wide class of generically composed schemes, some of which have been proven to achieve RAE security. This class consists of schemes built from a hash function $\mathcal{H}$, which we treat as a random oracle, and an encryption scheme $(\mathcal{E}, \mathcal{D})$, which we consider to be an ideal AEAD *without* associated data. We assume that the encryption algorithm of the composed scheme operates as follows. An initialization procedure $\mathcal{I}_e$ is used to prepare the inputs to a preprocessing algorithm $\mathcal{E}_0^{\mathcal{H}}$ and a post-processing algorithm $\mathcal{E}_1^{\mathcal{H}}$. The preprocessing algorithm prepares the inputs to the underlying $\mathcal{E}$ algorithm. The post-processing algorithm gets the output ciphertext and completes encryption (e.g., by appending a tag value). The decryption algorithm operates analogously by reversing this process via an initialization procedure $\mathcal{I}_d$, a preprocessing algorithm $\mathcal{D}_0^{\mathcal{H}}$ and a post-processing algorithm $\mathcal{D}_1^{\mathcal{H}}$. See Figure 4 for the details.

The next theorem shows that this class of generically composed schemes are differentiable as long as certain conditions on information passed between the above sub-procedures are met.

**Theorem 3 (Differentiability of generic composition).** *Let* $\Pi$ *be a generically composed AEAD scheme from an encryption scheme (without associated data)* $(\mathcal{E}, \mathcal{D})$ *and a hash function* $\mathcal{H}$ *that follows the structure shown in Figure 4 for some algorithms* $(\mathcal{I}_e, \mathcal{E}_0, \mathcal{E}_1, \mathcal{I}_d, \mathcal{D}_0, \mathcal{D}_1)$*. Let* $\Delta_C := |C| - |C'|$ *denote the ciphertext overhead added by the transform. Suppose that one of the following conditions holds.*

**Type-I** *: Let* $est_1$ *be the state passed to* $\mathcal{E}_1$*. We require that for all inputs* $(K, N, A, M)$ *and for a sufficiently large* $\Delta_1$ *we have that* $|(K, N, A, M)| - |est_1| \geq \Delta_1$*.*[11] *Furthermore, there is a recovery*

[11] We do not count the length of $\tau$ as our attacks also work for fixed values of $\tau$.

*algorithm $\mathcal{R}_1$ (with no oracle access) that on input $C$ recovers $C'$, the internal ciphertext output by $\mathcal{E}$.*

**Type-II** *: Let $dst_0$ be the state passed to $\mathcal{D}_0$. We require that for all inputs $(K, N, A, C)$ and for a sufficiently large $\Delta_2$ we have that $|(K, N, A, C)| - |dst_0| \geq \Delta_2$. Furthermore, there is a recovery algorithm $\mathcal{R}_2$ (with no oracle access) that on input $M'$ recovers $M$, the output of $\mathcal{D}_1^{\mathcal{H}}(M', dst_1)$ in decryption.*

*Then $\Pi$ is differentiable. More precisely, for any type-I (resp., type-II) scheme $\Pi$ there exists a differentiator $\mathscr{D}_1$ (resp., $\mathscr{D}_2$) such that for any simulator $\mathcal{S}$ making at most $q$ queries in total to its ideal AEAD oracles*

$$\mathbf{Adv}^{\text{indiff}}_{\Pi,\mathcal{S}}(\mathscr{D}_1) \geq 1 - q/2^{\Delta_1} - (q+1)/2^{\Delta_C} \quad \textit{and} \quad \mathbf{Adv}^{\text{indiff}}_{\Pi,\mathcal{S}}(\mathscr{D}_2) \geq 1 - q/2^{\Delta_2} - (q+1)/2^{n} \ .$$

*Proof.* We start with type-I schemes. The differentiator computes a ciphertext for a random set of inputs using the construction in the forward direction and then checks if the result matches that computed via the generic composition using the provided primitive oracles. To rule out the existence of successful simulators the differentiator must ensure that it does not reveal information that allows the simulator to use its ideal construction oracles to compute a correct ciphertext. The restriction on the size of $est_1$ (and the ability to recompute the internal ciphertext $C'$ via $\mathcal{R}_1$) will be used to show this. The pseudocode for the differentiator, which we call $\mathscr{D}_1$, is shown in Figure 5 (left). The attack works for any given value of $\tau$, and to simplify the presentation, we have assumed all spaces consist of bit strings of length $n$.

Analysis of $\mathscr{D}_1$. It is easy to see that when $\mathscr{D}_1$ is run in the real world with respect to the generically composed construction and with its correctly implemented underlying primitives, its output will be always 1. This follows from the fact that $\mathcal{R}_1(C)$ will correctly recover the internal ciphertext $C'$ and hence $\mathcal{E}_1^{\text{PRIM}_2}(C', est_1)$, being run with respect to correct inputs and hash oracle, will also output $C$.

We now consider the ideal world. We first modify the ideal game so that the ideal object presented to the simulator is independent of that used to answer construction queries placed by the differentiator. This game is identical to the ideal world unless $\mathcal{S}$ queries the forward construction oracle on $(K, N, A, M, \tau)$ (call this event $E_1$) or the backward construction oracle on $(K, N, A, C, \tau)$ (call this event $E_2$). We will bound the probability of each of these events momentarily. In the modified game, we claim that no algorithm $\mathcal{S}$ can compute $C$ from $(C', est_1)$. This is the only information about $C$ that is revealed to a simulator and this claim in particular means that running $\mathcal{E}_1^{\text{PRIM}_2}(C', est_1)$ within $\mathscr{D}_1$ won't output the correct $C$ either. The answers to oracle queries placed by $\mathcal{S}$ can be computed independently of the ideal construction oracles. Furthermore, $(C', est_1)$ misses at least $\Delta_C$ bits of information about $C$ as $est_1$ is computed independently of $C$. The simulator therefore has at most a probability of $1/2^{\Delta_C}$ of outputting $C$ in this game.

We now bound the probability of events $E_1$ and $E_2$ in the modified game. The occurrence of Event $E_2$ is similar to the analysis above for guessing the value of $C$, but here we admit that $\mathcal{S}$ can make $q$ distinct guesses corresponding to its maximum number of queries; this results in an overall bound $q/2^{\Delta_C}$. To analyze event $E_1$, we observe that $(C', est_1)$ misses $\Delta_1$ bits of information about the uniformly chosen $(K, N, A, M)$ as $C'$ is simply a randomly chosen string. Hence, over $q$ distinct queries to the construction oracle, $\mathcal{S}$ can cause such an event with probability at most $q/2^{\Delta_1}$. We therefore obtain an overall upper bound of $(q+1)/2^{\Delta_C} + q/2^{\Delta_1}$ for guessing $C$.

Let us now consider type-II schemes. The differentiator will compute a ciphertext corresponding to a random input via the construction oracle and will check if it matches that computed via the

| $\underline{\text{ALGO. } \mathscr{D}_1^{\text{CONST}^+,\text{PRIM}_2}(\tau)}$ | $\underline{\text{ALGO. } \mathscr{D}_2^{\text{CONST}^+,\text{PRIM}_1^-,\text{PRIM}_2}(\tau)}$ |
|---|---|
| $(K,N,A,M) \twoheadleftarrow \{0,1\}^{4n}$ | $(K,N,A,M) \twoheadleftarrow \{0,1\}^{4n}$ |
| $C \leftarrow \text{CONST}^+(K,N,A,M,\tau)$ | $C \leftarrow \text{CONST}^+(K,N,A,M,\tau)$ |
| $(est_0, est_1) \leftarrow \mathcal{I}_e(K,N,A,M,\tau)$ | $(dst_0, dst_1) \leftarrow \mathcal{I}_d(K,N,A,C,\tau)$ |
| $C' \leftarrow \mathcal{R}_1(C)$ | $(K',N',C',\tau') \leftarrow \mathcal{D}_0^{\text{PRIM}_2}(dst_0)$ |
| $\tilde{C} \leftarrow \mathcal{E}_1^{\text{PRIM}_2}(C', est_1)$ | $M' \leftarrow \text{PRIM}_1^-(K',N',C',\tau')$ |
| return $(\tilde{C} = C)$ | $\tilde{M} \leftarrow \mathcal{R}_2(M')$ |
| | return $(\tilde{M} = M)$ |

**Fig. 5.** Differentiators $\mathscr{D}_1$ and $\mathscr{D}_2$ against type-I and type-II schemes respectively.

generically composed encryption algorithm. In this case, however, the differentiator will perform the check by attempting to decrypt the internal ciphertext passed to $\mathcal{E}$ and verifying if the result matches the *message* queried to the construction oracle. Once again, the differentiator has to ensure that the simulator does not learn sufficient information to recover the originally queried message. (For example, the simulator would be able to do this via the backward construction oracle if it learns $(K,N,A,C)$.) This will be guaranteed by the restriction imposed on the size of $dst_0$. Algorithm $\mathcal{R}_2$ will be used to recover the original message $M$ from the internal message $M'$ (e.g., by removing any tag values or masking). The pseudocode for this differentiator, which we call $\mathscr{D}_2$, is shown in Figure 5 (right). Again, the attack will work for any fixed value of $\tau$.

ANALYSIS OF $\mathscr{D}_2$. It is easy to see that when $\mathscr{D}_2$ is run in the real world with respect to the generically composed construction and its correctly implemented underlying primitives, the output will be always 1. This follows from the fact that $\text{PRIM}_1^-$ oracle will correctly recover the internal message $M'$ and hence $\mathcal{R}_2$, being run on the correct input, will recover the original message $M$.

We follow a similar strategy to that for type-I schemes to analyze the success probability of simulators. We first modify the ideal game so that the ideal object presented to the simulator is independent of that used to answer construction queries placed by the differentiator. This game is identical to the ideal world unless $\mathcal{S}$ queries the forward construction oracle on $(K,N,A,M,\tau)$ (call this event $E_1$) or the backward construction oracle on $(K,N,A,C,\tau)$ (call this event $E_2$). In this modified game, no algorithm $\mathcal{S}$ can compute $M$ from $dst_0$ alone (note that $(K',N',C',\tau')$ are determined based on $dst_0$). The answers to oracle queries placed by $\mathcal{S}$ can be computed independently of the ideal construction oracles and since $dst_0$ is independent of $M$—it is computed from $(K,N,A,C,\tau)$ for a random $C$—the probability of guessing $M$ is $1/2^n$. This in particular means that the sequence of statements in $\mathscr{D}_2$ consisting of $\mathcal{D}_0^{\text{PRIM}_2}$, $\text{PRIM}_1^-$ and $\mathcal{R}_2$ will output the correct $M$ with probability $1/2^n$ as the sole input to perform these statements is $dst_0$.

We now bound the probability of events $E_1$ and $E_2$ in the modified game. The occurrence of Event $E_2$ is similar to the analysis above for guessing the value of $M$, but here we admit that $\mathcal{S}$ can make $q$ distinct guesses corresponding to its maximum number of queries; this results in an overall bound $q/2^n$.

Finally, we consider the probability of event $E_2$ where the simulator queries $(K,N,A,C,\tau)$. Since the simulator only sees $dst_0$, at least $\Delta_2$ bits of information about the uniformly chosen $(K,N,A,C)$ are missing from its view. Hence, over $q$ distinct queries to the construction oracle, $\mathcal{S}$ can cause such an event with probability at most $q/2^{\Delta_2}$. We therefore obtain an over all upper bound of $(q+1)/2^n + q/2^{\Delta_2}$ for guessing $M$. □

Consequences for generic composition. Namprempre, Rogaway, and Shrimpton [NRS14] explore various methods to generically compose an AEAD scheme from a nonce-based AE scheme (without associated data) and a MAC. In their analysis the authors single out eight favored schemes A1–A8. Roughly speaking, schemes A1, A2, and A3 correspond to Encrypt-and-MAC where, respectively, $N$, $(N, A)$, and $(N, A, M)$ are used in the preparation of the input $IV$ to the base AE scheme. Scheme A4 is the Synthetic Initialization Vector (SIV) mode of operation [RS06, Figure 5], which is misuse resilient. Schemes A5 and A6 correspond to Encrypt-then-MAC, where $IV$ is computed using $N$ and $(N, A)$, respectively. Schemes A7 and A8 correspond to MAC-then-Encrypt, where $IV$ is computed using $N$ and $(N, A)$ respectively. The MAC component in all these schemes is computed over $(N, A, M)$. Key $L$ is used for $IV$ and MAC generation, and an independent key $K$ is used in encryption. We refer the reader to the original paper [NRS14, Figure 2] for further details. For convenience we reproduce this figure in Appendix A, Figure 20.

A-schemes. Indifferentiability of A1–A6 can be ruled out using the first differentiator $\mathscr{D}_1$ above: the computation of tag $T$ does not depend on the encryption key $K$, which means that $K$ can be omitted from $est_1$, and hence $|est_1|$ is sufficiently smaller than $|(K|L, N, A, M)|$. We can rule out the indifferentiability of A7 via our second differentiator $\mathscr{D}_2$: the computation of $IV$ does not depend on $A$, which means that $IV$ can be omitted from $dst_0$ and hence $|dst_0|$ is sufficiently smaller than $|(K|L, N, A, C)|$. (Note that the required recovery algorithms in both cases trivially exist and simply read off the correct outputs from the inputs.) This leaves A8 as the only favored scheme that could meet indifferentiability. Our construction in the next section is similar to A8 and simplifies it by providing authenticity via redundancy (rather than a MAC tag computation).

Key reuse. NRS [NRS14] consider independent keys $K$ and $L$ in their generically composed schemes, a choice that is needed for standard-model security analyses. With ideal components, however, one can consider *reuse of keys*: setting $K = L$ avoids the differentiator $\mathscr{D}_1$. With this modification, however, the second differentiator $\mathscr{D}_2$ can still be used to rule out A1, A3, A4, A5, and A7. Indeed, in all these schemes, $IV$ is either available in the clear as part of the ciphertext, or its computation does not depend on $A$ at all. This means that $A$ can be excluded from $dst_0$ and hence $|dst_0|$ is smaller than $|(K, N, A, C)|$. This leaves the modified A2, A6, and A8 schemes (i.e., with key reuse) as plausibly indifferentiable candidates. We note that these schemes, respectively, correspond to the (enhanced) Encrypt-and-MAC, Encrypt-then-MAC, and MAC-then-Encrypt transforms with key reuse. (Here both $N$ and $A$ are included in the computation of $IV$ and $(N, A, M)$ is included in the computation of tag $T$.) Proving the indifferentiability of these transforms would bring new and strong assurances for these classical constructions. We leave establishing these (or disproving them) to future work.[12]

Other schemes. NRS also identify eight schemes B1–B8 [NRS14, Figure 3] associated to the A-schemes that are more efficient as their MAC is instantiated via the "XOR3" construction. We observe the B-schemes are not indifferentiable either. Schemes B1, B3, B4, B5, and B7 (with or without key reuse) can be attacked identically to their corresponding A-schemes. For the remaining schemes (with or without key reuse), a differentiator can ensure $IV := f_{L1}(N) \oplus f_{L3}(A)$ is always zero by setting $L1 = L3$ and $N = A$. This then enables launching the same attack performed by $\mathscr{D}_2$ in Figure 5 with respect to $IV = 0$. Among the four N-schemes [NRS14, Figure 6], N1 and N2 follow the structure of A1 and A5 respectively, are type-I and hence differentiable. Schemes N3 and

[12] We note, however, that a positive result would not provide a satisfactory construction as the underlying assumption of an ideal AE (without associated data) is very close to the desired end goal of an ideal AEAD.

N4 follow that of A7, are therefore type-II and hence also differentiable. In the literature, we also found a recent scheme called Robust Initialization Vector (RIV) [AFL+16], that is MRAE secure and bears similarities to our construction. We show in Appendix C that RIV is type-I and hence differentiable. For the readers' convenience, we reproduce the original figures depicting B-schemes and N-schemes from [NRS14] in Appendix A.

### 4.2 A glimpse of CAESAR

We now take a brief look at the indifferentiability of the 15 third-stage candidates in CAESAR [Ber14]. All except two of these candidates are one-pass, preserve prefixes, and hence do not meet MRAE security [RS06, HKR15]. Therefore, by the observations at the beginning of this section, they are also differentiable. The remaining two candidates are DEOXYS-II, a misuse-resilient version of DEOXYS [JNPS16], and AEZ [HKR15, HKR17].

DEOXYS-II follows the Synthetic Counter-in-Tweak (SCT) composition pattern [PS16]. This is akin to scheme A4 (SIV) discussed above [NRS14] with two exceptions: (1) the key $K$ is reused in the MAC and encryption computations and (2) the nonce $N$ is both authenticated *and* directly fed into the encryption stage. However, as in scheme A4, the $IV$ is encoded explicitly in the ciphertext, which means that associated data need not be known when inverting the encryption layer (cf. [PS16, Figure 4]). According to the terminology introduced in the previous section, associated data $A$ does not need to be present in $dst_0$. For this reason, DEOXYS-II and SCT both fall prey to the same differentiating attack we described for the A4 construction with key reuse.

For AEZ, we consider the Encipher-AEZ-core algorithm of AEZ v5 [HKR17, Figure 3]. (See also Figure 5 there for a schematic diagram of the construction, which we reproduce for the reader's convenience in Appendix B.) The computation of the final two ciphertext blocks in this algorithm follows a 4-round Feistel network. For any input message of the form $M_1 M'_1 \cdots M_m M'_m M_{\mathsf{uv}} M_{\mathsf{x}} M_{\mathsf{y}}$, the final ciphertext block $C_{\mathsf{y}}$ is fully determined by $(K, M_{\mathsf{x}} \oplus X, \Delta, M_{\mathsf{y}})$. Here the message is broken into blocks of 128-bits, except for $M_{\mathsf{uv}}$ which is at most 256-bits long, $K$ is the secret key, $\Delta$ is a hash value that is independent of the message being enciphered, and $X$ is fully determined by $(K, M_1 M'_1 \cdots M_m M'_m M_{\mathsf{uv}})$, i.e., by the key and the initial message blocks.

This design is not a problem in the RAE security model, since $X$ and $\Delta$ are unpredictable when $K$ remains hidden from adversary. However, an indifferentiability attacker can learn these intermediate values. Indeed, for any message $\mathbf{M} M_{\mathsf{x}} M_{\mathsf{y}}$, an attacker can use the oracles for the underlying round functions to compute $X$ and then choose a distinct message $\mathbf{M}' M'_{\mathsf{x}} M_{\mathsf{y}}$ such that $M_{\mathsf{x}} \oplus X = M'_{\mathsf{x}} \oplus X'$. This means that the attacker can cause a collision in the final block of the construction. A successful simulator in the ideal world would have to provide responses for the round functions so that they also give rise to similar collisions with respect to ideal injection oracles. This, however, is infeasible for any simulator that places a polynomial number of oracle calls. The argument is similar to that used for a differentiating attack against the 5-round Feistel network presented in [CPS08]: With high probability the attacker in the real world is able to generate a set of construction input/output values that satisfy a non-trivial (XOR-based) relation. In the ideal world, however, the simulator is bound to a random object for which such relations only hold with negligible probability. This means that no simulation strategy can be successful.

### 4.3 Interpretations

The indifferentiability composition theorem, along with its generality and modularity implications as discussed in the introduction, provide a strong case in support of adopting indifferentiability

as a new design criteria for AEAD schemes. Put differently, a successful differentiator against an AEAD scheme, even an RAE-secure one, can be indicative of potential deficiencies that can manifest themselves as vulnerabilities when used in application scenarios that require strong levels of security. By proving a scheme indifferentiable we show that no such vulnerabilities exist for *any* single-stage security game.

On the other hand, the stronger indifferentiability guarantees could come with efficiency penalties. Indeed, for ideal permutations (which can be seen as special forms of expansion 0 AEAD schemes) the efficiency of the best known indifferentiable constructions is significantly worse than what can be achieved in the standard model: the best results for Feistel networks and Confusion-Diffusion networks yield rate 7 constructions (i.e., where the underlying primitive is called at least 7 times per input block) whereas rate 3 [DKS+17] and rate 4 [Luc96] constructions are known in the standard model.

The efficiency penalties are not necessarily so large when it comes to AEADs. In the following sections we will show how for large stretch $\tau$ a rate 3 indifferentiable scheme can be provably built. This is not too far from the efficiency of practical RAE-secure constructions (cf. rate 2.5 for AEZ). On the other hand, in Section 6 we give a lower bound showing that rate $\geq 2$ is *necessary*. Although extending this result to (M)RAE schemes remains open, we conjecture that this is indeed the case. (In particular all MRAE scheme we have found have rate at least 2.) This means a 20–30% increase in the number of primitives calls in our constructions, which is relatively a small price to pay for the strong levels of security enjoyed by indifferentiable schemes. Moreover schemes with a rate closer to that for RAE schemes are not (yet) ruled out. We leave finding such constructions (or proving their impossibility) as an open challenge for the community.

## 5 Ideal Offline AEAD

We now give two constructions of ideal AEAD from simpler ideal primitives. The first is based on a VIL blockcipher, it enjoys a simpler analysis and supports any expansion $\tau$. The second is based on the unbalanced 3-round Feistel network, where round functions are alternatively compressing and expanding random oracles. It achieves higher efficiency, but here $\tau$ must be sufficiently large.

We present our proofs in a modular way. We first build ideal AEADs that achieve indifferentiability in a restricted setting where all parameters except the input message are fixed. More precisely, we first show that there is a simulator $\mathcal{S}$ that for any *arbitrary but fixed* value of $K' := (K, N, A, \tau)$ is successful against all differentiators that are $K'$-bound in the sense that they only query the construction and primitive oracles on values specified by $K'$. To this end, we also begin with the simplifying assumption that the underlying ideal objects can be keyed with keys of arbitrary length. We then show how these restrictions and simplifying assumptions can be removed to obtain fully indifferentiable AEADs.

### 5.1 Indifferentiability of Encode-then-Encipher

Our first construction transforms a VIL ideal cipher with arbitrary key space into an ideal AEAD. It follows the Encode-then-Encipher (EtE) transform of Bellare and Rogaway [BR00]. In its most simple form, EtE fixes $\tau$ bits of the input to $0^\tau$ and checks the correctness of the included redundancy upon inversion (see Figure 6).[13] The domain of the underlying blockcipher should therefore be at

[13] In both the EtE construction and the Feistel construction in the next section, the $0^\tau$ constant can be replaced by any fixed constant $\Delta$ of the same length. For EtE the indifferentiability proof is the same. For the Feistel

least $\tau$ bits longer than that needed for the injection. This, in particular, is the case when both objects have variable input lengths.

| Algo. $\mathcal{AE}(K, N, A, M, \tau)$ | Algo. $\mathcal{AD}(K, N, A, C, \tau)$ |
|---|---|
| $K' \leftarrow (K, N, A, \tau)$ | $K' \leftarrow (K, N, A, \tau)$ |
| $C \leftarrow \mathsf{E}(K', 0^\tau \vert M)$ | $T \vert M \leftarrow \mathsf{E}^-(K', C)$, where $\vert T \vert = \tau$ |
| return $C$ | if $T \neq 0^\tau$ return $\perp$ else return $M$ |

**Fig. 6.** The (un-hashed) Encode-then-Encipher construction. In the full scheme we set $K' \leftarrow \mathcal{H}(K, N, A, \tau)$ for a random oracle $\mathcal{H}$.

The results of this section (in contrast to the attacks against other generic schemes) support the soundness of EtE-based schemes from an indifferentiability perspective.

**Theorem 4 (EtE is indifferentiable).** *The EtE construction in Figure 6 is indifferentiable from an ideal AEAD for any fixed $K' := (K, N, A, \tau)$ when instantiated with a VIL ideal cipher $(\mathsf{E}, \mathsf{E}^-)$. More precisely, there is an expected $4q$-query simulator $\mathcal{S}(\cdot\,; K')$ that presents a perfect simulation of the underlying permutation for any $K'$-bound $q/2$-query differentiator $\mathscr{D}$ for $q/2 \leq 2^{n+\tau}/8$.*

*Proof.* We start by observing that since $K'$ non-ambiguously encodes $(K, N, A, \tau)$, the simulator gets to the $\tau$ that the differentiator is bound to when its first (forward or backward) query arrives. Furthermore, for any such fixed $K'$, the ideal AEAD objects $(\mathsf{E}, \mathsf{E}^-)$ simply implement a random injection $(\rho, \rho^-)$ with expansion $\tau$ and the simulator only has to simulate a random permutation $(\pi, \pi^-)$ rather than an ideal cipher. Hence we will present a simulator for permutations and with respect to a random injection that receives a fixed $\tau$ at the onset as input. (The specific values of $(K, N, A)$ are not needed by the simulator; it is only important to recall that they are fixed throughout the attack.)

Intuitively, the simulator will simulate the permutation on inputs of the form $0^\tau \vert M$ via the ideal AEAD oracle $\rho$ and will use a lazily sampled injection *disjoint* from $\rho$ (i.e., one whose domain and range are disjoint from those of $\rho$) for inputs of the form $T \vert M$ with $T \neq 0^\tau$. The pseudocode for the simulator $\mathcal{S}^{\rho^\pm} := (\mathcal{S}^\rho_+, \mathcal{S}^{\rho^-}_-)$ is shown in Figure 7. In the description of the simulator we use procedures $\text{LazyIC}^\pm(P; L)$ for the lazy sampling of an ideal cipher in the forward and backward directions conditioned on a partial list of already assigned elements $L$; see Section 2.2.

Let $\mathscr{D}$ be a differentiator. We show the views of $\mathscr{D}$ in the real and ideal worlds with respect to the simulator $\mathcal{S}$ above are identical. The view of $\mathscr{D}$ consists of inputs to and outputs from the forward/backward directions of the (real or ideal) injection and forward/backward directions of the (real or simulated) permutation. More precisely, such a view (in either world) can be seen as a list containing entries of the form

$$(\textsc{Const}^+, M, C)\ , \quad (\textsc{Const}^-, M, C)\ , \quad (\textsc{Prim}^+, X, Y)\ , \quad (\textsc{Prim}^-, X, Y)\ ,$$

indicating the oracle, its input, and the output received (for the inverse queries the output is written first). By observing that the oracles in both the real and ideal worlds faithfully implement functions

construction the proof can be easily adapted. To see this, note that any round function $F_1(X)$ can be replaced with an indifferentiable one $F'_1(X) = \Delta \oplus F_1(X)$. The resulting construction becomes identical to the one using $0^\tau$ by cancellation.

| Algo. $\mathcal{S}^{\rho}_{+}(X;\tau)$ | Algo. $\mathcal{S}^{\rho^-}_{-}(Y;\tau)$ |
|---|---|
| if $X = (0^\tau \vert M)$<br>  $Y \leftarrow \rho(M,\tau)$<br>  return $Y$<br>$L' \leftarrow L$<br>do<br>  $(Y;L) \twoheadleftarrow \text{LazyIC}(X;L')$<br>while $\rho^-(Y,\tau) \neq \perp$<br>return $Y$ | $M \leftarrow \rho^-(Y,\tau)$<br>if $M \neq \perp$<br>  $X \leftarrow 0^\tau \vert M$<br>  return $X$<br>$L' \leftarrow L$<br>do<br>  $(X;L) \twoheadleftarrow \text{LazyIC}^-(Y;L')$<br>while $X = 0^\tau \vert M$<br>return $X$ |

**Fig. 7.** The simulator for un-hashed EtE and with respect to a fixed $\tau$. List $L$ is initialized to empty.

and their inverses we can simplify $\mathscr{D}$'s view to a list containing entries of the form

$$(\textsc{Const}, M, C) \ , \quad (\textsc{Prim}, X, Y) \ .$$

We assume, without loss of generality and at the expense of doubling the query complexity of the differentiator to $q$, that before termination $\mathscr{D}$ places the following queries.[14] (1) For all entries $(\textsc{Const}, M \neq \perp, C)$ it queries $X := 0^\tau \vert M$ to $\textsc{Prim}^+$. (2) For all entries $(\textsc{Const}, \perp, C)$ it queries $C$ to $\textsc{Prim}^-$. Note that in both the real and ideal worlds $\textsc{Prim}^-$ returns a value $T \vert M$ where $T \neq 0^\tau$. (3) For all entries of the form $(\textsc{Prim}, 0^\tau \vert M, Y)$ it queries $M$ to $\textsc{Const}^+$. (4) For all entries of the form $(\textsc{Prim}, T \vert M, Y)$ with $T \neq 0^\tau$ it queries $Y$ to $\textsc{Const}^-$. This allows us to match $\textsc{Const}$ and $\textsc{Prim}$ queries in a one-to-one manner and partition $\mathscr{D}$'s view in both the real and ideal worlds as follows.

(1) Entries $(\textsc{Const}, M, C)$ and $(\textsc{Prim}, 0^\tau \vert M, Y)$ with $M \neq \perp$. We call these invertible chains.
(2) Entries $(\textsc{Const}, \perp, C)$ and $(\textsc{Prim}, T \vert M, C)$ with a $T \neq 0^\tau$. We call these non-invertible chains.

Let us now look at the views of $\mathscr{D}$ in the two worlds.

Real world. In the real world, the distribution of $\mathscr{D}$'s view is induced by application of the random permutation $\pi$ to inputs. Hence the invertible and non-invertible chains are distributed as show below for a random permutation $\pi$.

$$\begin{aligned} &(\textsc{Const}, M, \pi(T \vert M)) \ , \quad (\textsc{Prim}, T \vert M, \pi(T \vert M)) \quad \text{where} \quad T = 0^\tau \ , \\ &(\textsc{Const}, \perp, \pi(T \vert M)) \ , \quad (\textsc{Prim}, T \vert M, \pi(T \vert M)) \quad \text{where} \quad T \neq 0^\tau \ . \end{aligned}$$

Ideal world. In the ideal world, since the simulator *knows* $\tau$ and has access to $(\rho, \rho^-)$, it can detect queries that belong to an invertible chain. If invertible, it simulates the output of the permutation via the random injection. Hence the $\mathscr{D}$'s view of invertible chains is

$$(\textsc{Const}, M, \rho(M)) \ , \quad (\textsc{Prim}, 0^\tau \vert M, \rho(M)) \ .$$

This is identically distributed to invertible chains in the real world as the outputs of $\rho(\cdot)$ and $\pi(0^\tau \vert \cdot)$ are identically distributed. If the simulator detects that a value belongs to a non-invertible chain,

[14] This is indeed without loss of generality because for any $\mathscr{D}$ that does not have this behavior, we can construct another one that runs $\mathscr{D}$, waits for it to decide on its output, and then performs the missing queries.

it uses lazy sampling of a random permutation subject to the condition that sampled range values are not in the range of $\rho$ and sampled domain vales do not takes the form $0^\tau|M$. Hence these values are distributed as

$$(\textsc{Const}, \perp, \tilde{\pi}(T|M)) \ , \quad (\textsc{Prim}, T|M, \tilde{\pi}(T|M)) \quad \text{where} \quad T \neq 0^\tau$$

for a random permutation $\tilde{\pi}$ that is disjoint from the from the permutation induced by $\rho$. By construction, the simulator glues these two permutations together to get an overall random permutation, and hence we can deduce that the overall view of $\mathscr{D}$ is distributed as in the real world.

Runtime of $\mathcal{S}$. We now analyze the number of queries of the simulator to $\rho$ or $\rho^-$. For queries corresponding to invertible chains the simulator places a single call to the forward/backward ideal object. This, in particular, is always the case when $\tau = 0$. To analyze non-invertible chains, we assume $\tau \geq 1$. On each such query, the simulator places an initial call to the ideal object to detect that the query is non-invertible. $\mathcal{S}_+$ then samples a random $Y$, subject to permutativity, such that it does not have a preimage under $\rho^-$. Value $Y$ is sampled uniformly from a set of size at least $2^{n+\tau} - q$. Furthermore there are at most $2^n$ points that have a preimage under $\rho^-$. Hence the probability of rejection in each iteration is $\Pr[\rho^-(Y, \tau) \neq \perp] \leq \frac{2^n}{2^{n+\tau}-q}$ . Similarly, $\mathcal{S}_-$ looks for a random $T|M$ such that $T \neq 0^\tau$. There are at most $2^n$ values of $T|M$ that start with $0^\tau$ and the sampling is performed from a set of size at most $2^{n+\tau} - q$. Hence the probability of rejection in each iteration is again $\Pr[T \neq 0^\tau] = \frac{2^n}{2^{n+\tau}-q}$ . Assuming $q \leq 2^{n+\tau}/4$ (and $\tau \geq 1$), this probability is at most $2/3$. This in turn leads to $1/(1 - 2/3) = 3$ expected samples, and in total an expected 4 queries to the random injection for each query of $\mathscr{D}$. □

Remark. Our simulator runs in expected polynomial time and provides a perfect simulation of the permutation oracles. This simulator can be converted into one that runs in strict polynomial-time in the standard way by capping the number of samples to $t$ tries. With $q \leq 2^{n+\tau}/4$, this simulator fails with probability at most $(2/3)^t$ for each query of the differentiator, and hence introduces an overall statistical distance of $q(2/3)^t$.

### 5.2 Indifferentiability of 3-round Feistel

A variable-input-length (VIL) permutation can be constructed via the Feistel construction [CHK+16] from a VIL/VOL random oracle, or via the confusion-diffusion construction [DSSL16] from a fixed-input-length (FIL) random permutation.[15] By indifferentiability composition, the VIL/VOL hash function in Feistel can be instantiated with the Sponge construction [BDPV08] in the FIL random-permutation model.[16] The number of rounds needed for indifferentiability of Feistel from an ideal cipher has been gradually reduced from 14 [HKT11, CHK+16] to 10 [DS15, DKT16] and recently to 8 [DS16]. Due to the existence of differentiators [CPS08, CHK+16], the number of rounds must be at least 6.[17] For confusion-diffusion, 7 rounds are needed for good security bounds [DSSL16]. This

[15] We note that using a hybrid argument (and the fact that input lengths can be read from the inputs) the indifferentiability of the Feistel and confusion-diffusion constructions carry over to variable input lengths, provided that the underlying ideal objects also support variable-length inputs and outputs.

[16] As noted in [DGHM13], when dealing with domain and range extension for Sponge one needs to take care of encoding the lengths of inputs and outputs as part of the inputs fed to the random oracle.

[17] And provable indifferentiability for 6 rounds seems to be difficult [CPS08].

state of affairs leaves the above approach to the design of random injections somewhat suboptimal in terms of the number of queries per message block to a random permutation.

We ask weather or not this rate can be improved for random *injections*. Our starting point is the observation that indifferentiability attacks against 5-round Feistel do not necessarily translate to those that fix parts of the input to $0^\tau$. Despite this, we show that differentiating attacks against 2-round Feistel still exist. For concreteness, we refer to the first two rounds of the construction shown in Figure 1.

**Proposition 2 (Differentiability of 2-round Feistel).** *The 2-round unbalanced Feistel construction $\Phi_2$ with the left part of the input fixed to $0^\tau$ is differentiable from an ideal injection. More precisely, there is a differentiator 2-query differentiator $\mathscr{D}$ such for any $q$-query simulator $\mathcal{S}$*

$$\mathbf{Adv}^{\text{indiff}}_{\Phi_2,\mathcal{S}}(\mathscr{D}) \geq 1 - (2q+1)/2^n \ .$$

*Proof.* Consider the differentiator $\mathscr{D}$ in Figure 8 that checks the consistency of simulated output against the construction on a random input $X$. When $\mathscr{D}$'s oracles implement the real construction and the primitives $F_1$ and $F_2$, we have that $(X_2, X_3) = (F_1(X_1), X_1 \oplus F_2(F_1(X_1)))$. The second primitive oracle is then queried on $X_2 = F_1(X_1)$ which means that $Y_2 = F_2(F_1(X_1))$. Hence in this case $X_1 \oplus Y_2 = X_3$ with probability 1.

Let us now analyze the ideal game execution. We first modify the ideal game so that the ideal oracle available to the simulator is independent of that used to answer construction queries placed by the differentiator. This game is identical to the ideal game, unless the simulator queries the ideal injection oracle in the forward direction on $X_1$ (call this event $E_1$) or in the backward direction on $(X_2, X_3)$ (call this event $E_2$). We will bound the probability of each of these events occurring momentarily. In this modified game, however, $\mathcal{S}$ can only guess the random value $Y_2 = X_1 \oplus X_3$ with probability at most $1/2^n$, since the answers to all of its queries can be computed independently of the ideal construction oracle.

ALGO. $\mathscr{D}^{\text{CONST}^+,\text{PRIM}_2}$
$X_1 \twoheadleftarrow \{0,1\}^n$
$(X_2, X_3) \leftarrow \text{CONST}^+(X_1)$
$Y_2 \leftarrow \text{PRIM}_2(X_2)$
if $(X_1 \oplus Y_2 = X_3)$ return 1
else return 0

**Fig. 8.** Differentiator against 2-round Feistel.

The probability of event $E_1$ occurring is essentially that of the simulator outputting $q$ distinct values, over $q$ queries, and hitting the value $X_1$ of which it has no information; it is therefore at most $q/2^n$. Similarly, for event $E_2$, the probability of occurrence is that of guessing a length $n+\tau$ random string over $q$ distinct queries, and this is therefore at most $q/2^{n+\tau}$. The proposition follows from the combination of the previous bounds. □

The simplicity of the above attack and the necessity for large number of rounds in building indifferentiable permutations raise the undesirable possibility that many rounds would also be needed for building random injections. We show, perhaps surprisingly, that this is not the case and adding only one extra round results in indifferentiability as long as $\tau$ and the input size are sufficiently large.[18] This means, somewhat counter-intuitively, that the efficiency of constructions of ideal injections can be increased when a *higher* level of security is required. The 3-round Feistel construction and variable names are shown in Figure 1.

We present the more intricate part of the proof of the following theorem in the code-based game-playing framework of Bellare and Rogaway [BR06] to help its readability and verifiability.

[18] We note, on the other hand, that the previous construction applied to possibly small values of $\tau$.

**Theorem 5 (Indifferentiability of 3-round Feistel).** *Take the 3-round Feistel construction $\Phi_3$ shown in Figure 1 when it is instantiated with three independent keyed random oracles (the round functions are all keyed with the same key and they are VIL/FOL in rounds 1 and 3 and FIL/VOL in round 2). This construction is indifferentiable from an ideal AEAD scheme for any fixed key of the form $K' := (K, N, A, \tau)$. More precisely, there is a simulator $\mathcal{S}$ such that for all $(q_e, q_d, q_1, q_2, q_3)$-query $K'$-bound differentiators $\mathscr{D}$ with $q_e + q_d + 2q_1 + q_2 + q_3 \leq q$ we have*

$$\mathbf{Adv}^{\text{indiff}}_{\Phi_3,\mathcal{S}}(\mathscr{D}) \leq 9q^2/2^\tau \ ,$$

*as long as $q_2(q_1 + q_2 + q_3) \leq 2^{n+\tau}/2$ and $q_e + q_1 \leq 2^n/2$. The simulator places at most $q^2$ queries to its oracles.*

*Proof.* To make the notation lighter we omit the key input to the various ideal objects (as we are dealing with $K'$-bound differentiators) and indicate forward/backward queries to the construction or ideal AEAD by $C/C^-$, and queries to the real or simulated round functions by $F_1$, $F_2$, and $F_3$. To simplify the analysis, we consider a *restricted* class of differentiators that (1) query $C(X_1)$ before any query $F_1(X_1)$, and (2) never query $C^-$. We also call a simulator $C$-respecting if it calls $C$ only when simulating $F_1(X_1)$, in which case it places a single query $C(X_1)$. The following lemma shows that we can focus on restricted differentiators for $C$-respecting simulators.

**Lemma 1 (Restricting $\mathscr{D}$).** *For any $(q_e, q_d, q_1, q_2, q_3)$-query differentiator $\mathscr{D}$ there is a restricted $(q_e + q_1, 0, q_1, q_2, q_3)$-query differentiator $\mathscr{D}'$ such that for any $C$-respecting simulator $\mathcal{S}$*

$$|\mathbf{Adv}^{\text{indiff}}_{\Phi_3,\mathcal{S}}(\mathscr{D}') - \mathbf{Adv}^{\text{indiff}}_{\Phi_3,\mathcal{S}}(\mathscr{D})| \leq 3q_d/2^\tau \ ,$$

*as long as $q_e + q_1 \leq 2^n/2$.*

*Proof.* Given an unrestricted $\mathscr{D}$ consider the restricted $\mathscr{D}'$ that runs $\mathscr{D}$, answers its $C$, $F_1$, $F_2$ and $F_3$ queries using its own equivalent oracles, always queries $C(X_1)$ before a query $F_1(X_1)$ by $\mathscr{D}$, and answers $C^-(Y)$ queries with $\perp$, unless $Y$ was the output one of its $C$ queries (which include those of $\mathscr{D}$ to $F_1$). In this case $\mathscr{D}'$ returns the correct preimage using a list that it maintains. It is easily seen that $\mathscr{D}'$ is restricted. We analyze the distance between the outputs of $\mathscr{D}$ and $\mathscr{D}'$ in the two worlds.

Real world. The two outputs with respect to the real game are identical unless $\mathscr{D}$ queries $C^-$ on a $Y$ that results in $X_1 \neq \perp$ in the real game and in $\perp$ when run by $\mathscr{D}'$. The latter happens when $X_1$ was not queried to either $C$ or $F_1$. (Recall $\mathscr{D}'$ forwards both the $C$ and $F_1$ queries of $\mathscr{D}$ to its $C$ oracle.) Let us call this event $E$. We bound the probability of $E$ when $\mathscr{D}$ is run by $\mathscr{D}'$. First note that $\mathscr{D}'$ can be modified to a new $\mathscr{D}''$ that uses $F_1$, $F_2$, and $F_3$ to perfectly emulate $C$ via the construction. This $\mathscr{D}''$ can also convert a valid $C^-$ query $Y = (X_3, X_4)$ to a valid pair $(X_1, X_2 := F_1(X_1))$ by computing $X_2 := F_3(X_3) \oplus X_4$ and $X_1 := F_2(X_2) \oplus X_3$ *without* querying $F_1(X_1)$ (see Figure 1). Moreover, procedures $F_2$ and $F_3$ are independent of $F_1$, and $\mathscr{D}''$ can lazily sample them. We have thus built an algorithm $\mathscr{D}''$ that places at most $q_1 + q_e$ queries to a random oracle $F_1$ and outputs a valid pair $(X_1, F_1(X_1))$ in a list of $q_d$ possible candidates (corresponding to the inverse queries). By the union bound, the success probability of any such $\mathscr{D}''$ is upper bounded by $q_d/2^\tau$ (independently of $q_1 + q_e$).

Ideal world. We now turn to the outputs of $\mathscr{D}$ and $\mathscr{D}'$ in the ideal game. Since the outputs of the simulator are not affected by the extra query of $\mathscr{D}'$ to $C$, the two outputs are identical

unless a query $Y$ of $\mathscr{D}$ to $C^-$ results in $X_1 \neq \perp$ in the ideal world that is answered with $\perp$ by $\mathscr{D}'$. This happens if $X_1$ was not queried to either $C$ or $F_1$. Let us call this event $E$. We bound the probability of $E$ in the former setting. We view the composition of algorithms $\mathscr{D}$ and $\mathcal{S}$ as a single algorithm $\overline{\mathscr{D}}$ with access to $C$ and $C^-$. We now look at the probability that $\overline{\mathscr{D}}$ places a query $Y$ to $C^-$ and receives $X_1 \neq \perp$ but $\overline{\mathscr{D}}$ (note the over-line) never queried $C$ on $X_1$. Since the simulator is $C$-respecting, requiring that $\mathscr{D}$ does not query $C$ or $F_1$ on $X_1$ is equivalent to requiring that $\overline{\mathscr{D}}$ does not querying $C$ on $X_1$. Moreover, since $C$ is a *truly* random injection, after at most $q_e + q_1$ queries to $C$, the probability that a (fresh) candidate for $Y$ inverts successfully under $C^-$ is at most $(2^n - q_e - q_1)/(2^{n+\tau} - q_e - q_1)$. Assuming $q_e + q_1 \leq 2^n/2$, maximizing the numerator and minimizing the denominator, this translates to an overall upper bound of $q_d/(2^\tau - 1/2) \leq 2q_d/2^\tau$ for the $q_d$ queries that $\mathscr{D}$ places to $C^-$. (Note that queries of $\mathcal{S}$ to $C^-$ are not counted as $C^-$ is not replaced by $\perp$ for $\mathcal{S}$.)

The lemma will follow from putting together the bounds established above. □

We prove indifferentiability with respect to restricted differentiators via a sequence of games as follows. We start with the real game, which includes oracles for the construction and the round functions, and gradually modify the implementations of these oracles until: (1) the construction no longer places any queries to the round functions and is implemented as an ideal injection; and (2) the round functions use this (ideal) construction oracle. We start with a high-level description of these games, which are detailed in Figures 9–13.

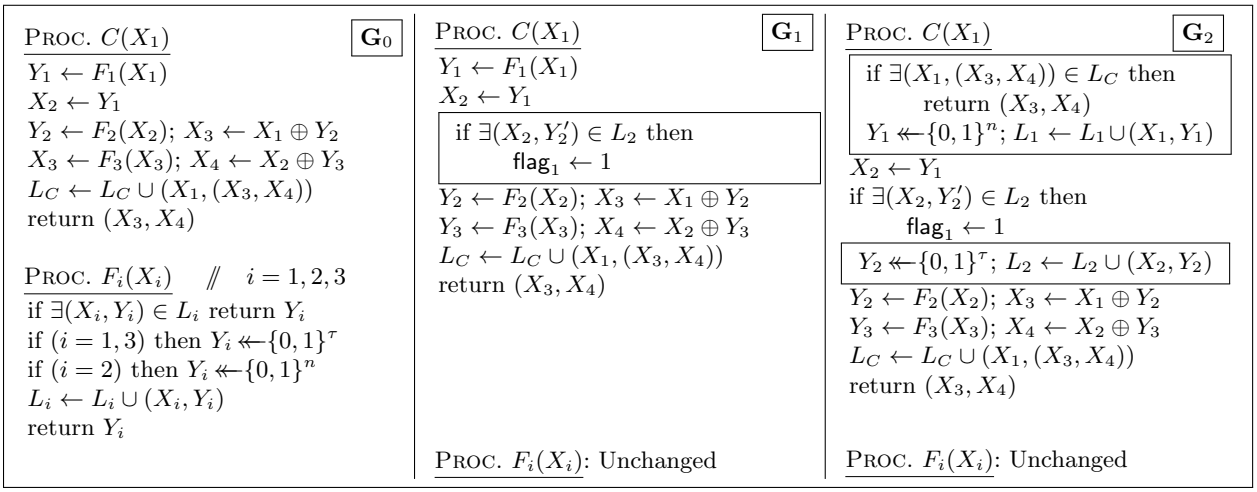


**Fig. 9.** Games $\mathbf{G}_0$, $\mathbf{G}_1$, and $\mathbf{G}_2$.

$\mathbf{G}_0$ : This game is identical to the (restricted) real game as shown in Figure 9 (left). Therefore in this game the construction oracle $C$ calls $F_1$, $F_2$ and $F_3$ and adds entries to their corresponding lists $L_1$, $L_2$, and $L_3$. Note that list $L_C$ is not used here.

$\mathbf{G}_1$ : This game introduces $\mathsf{flag}_1$. The game sets $\mathsf{flag}_1$ if $F_1$ chooses an output value that was already queried to $F_2$. As we will see, we can easily bound the probability of this flag getting set via the birthday bound.[19]

[19] As usual, once a flag is set, nothing matters. For example, we can assume that the game is set to return 0.

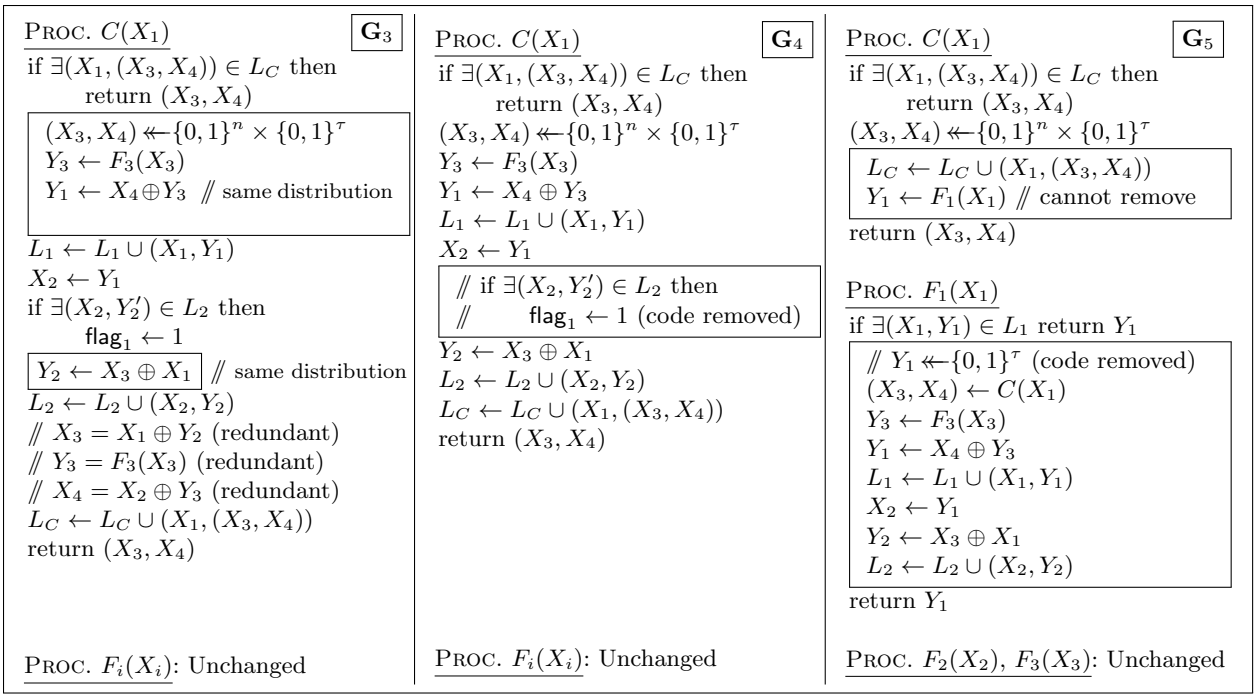


**Fig. 10.** Games $\mathbf{G}_3$, $\mathbf{G}_4$, and $\mathbf{G}_5$.

$\mathbf{G}_2$ : This game explicitly samples fresh values that are added to $L_1$ and $L_2$ as a result of a non-repeat query $X_1$ to $C$ within the code of $C$ rather than under the corresponding round functions. This is a conceptual modification and the game is identical to $\mathbf{G}_1$. Indeed, the sampled $L_1$ entry is always guaranteed to be fresh assuming a non-repeat value $X_1$, and the $L_2$ entry will be also non-repeat or $\mathsf{flag}_1$ is set. List $L_C$ is used to deal with repeat queries and avoid spurious samplings.

$\mathbf{G}_3$ : This game introduces a (conceptual) *change of random variables*. Instead of choosing $Y_1$ and $Y_2$ (i.e., the outputs of $F_1$ and $F_2$) randomly and computing the outputs $(X_3, X_4)$ of the construction, it first chooses $(X_3, X_4)$ and sets $Y_1$ and $Y_2$ based on these, the input, and $Y_3$. This is done via a linear change of variables that will not affect the distributions of $Y_1$ and $Y_2$, as we show below. This game constitutes our first step in constructing the simulator by defining the outputs of $F_1$ and $F_2$ in terms of those for $C$. The proof, however, is not yet complete: although $C$ is implemented independently of the round functions, $F_2$ and $F_3$ need access to the list of queries made to $C$. This is not permitted in the (full) indifferentiability setting. In the rest of the proof we remove this dependency.

$\mathbf{G}_4$ : This game removes $\mathsf{flag}_1$ (which allowed the previous transitions to be carried out in a conservative way) as we wish to gradually construct the code of the simulator, and this code is not needed in the final simulation.[20]

$\mathbf{G}_5$ : This game shifts most of the code from the $C$ oracle to the $F_1$ oracle. In particular, the manipulations of $L_1$ and $L_2$ are now done within $F_1$. The outputs of $C$ are still sampled within the construction procedure and $C$ makes a call to $F_1$. Procedure $F_1$ retrieves the necessary

[20] We need not introduce additional terms as a results of this change. Indeed, suppose games $\mathbf{G}$ and $\mathbf{G}''$ never set $\mathsf{flag}$, but game $\mathbf{G}'$ does. If these three games are identical until $\mathsf{flag}$ is set, then the distance between $\mathbf{G}$ and $\mathbf{G}''$ is still bounded by the probability of $\mathsf{flag}$ getting set in any game.

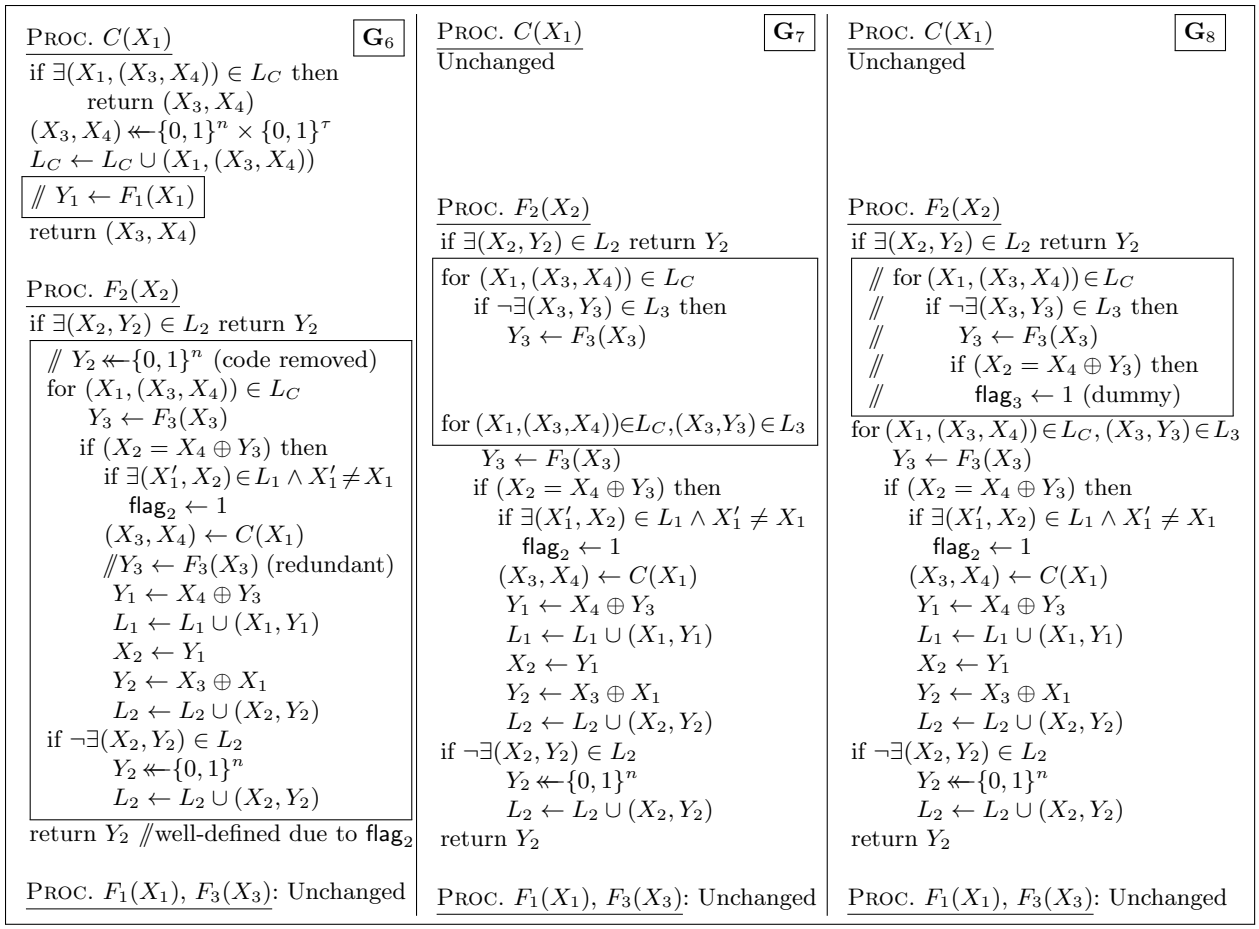


**Fig. 11.** Games $\mathbf{G}_6$, $\mathbf{G}_7$, and $\mathbf{G}_8$.

$(X_3, X_4)$ values by calling back the construction (note these are now added to $L_C$ prior to calling $F_1$). This modification is conceptual since (1) restricted differentiators *always* call the construction oracle before calling $F_1$ and hence the entry for $X_1$ will already be in the list $L_C$, and (2) although some queries to $F_2$ and $F_3$ may no longer be done, these oracles behave as random oracles and hence performing such queries earlier or later does not affect the view of the adversary in any way.

$\mathbf{G}_6$ : This game removes the query to $F_1$ from $C$ and adds a bad event based on $\mathsf{flag}_2$ to $F_2$ that guarantees that this game is identical to $\mathbf{G}_5$ until $\mathsf{flag}_2$. Removing the call to $F_1$ from $C$ has implications for $F_2$, since the operation of this oracle depends on entries that were added to $L_2$ whenever a call to $C$ (and therefore a call to $F_1$) occurred. For each $F_2$ query, we therefore need to ensure that processing left undone in this modified construction oracle (which may influence the view of the adversary) is carried out as before. To this end, we go through the entries in $L_C$ and check if an entry $(X_1, (X_3, X_4))$ occurred that might have set the value of $Y_2$. If more than one such entry exists, then this is detected as a collision at the output of $F_1$ and $\mathsf{flag}_2$ is set. If only one candidate is found, this corresponds exactly to the query that would have been made by the removed $F_1$ call. If no candidate is found, then the oracle simply samples a fresh value

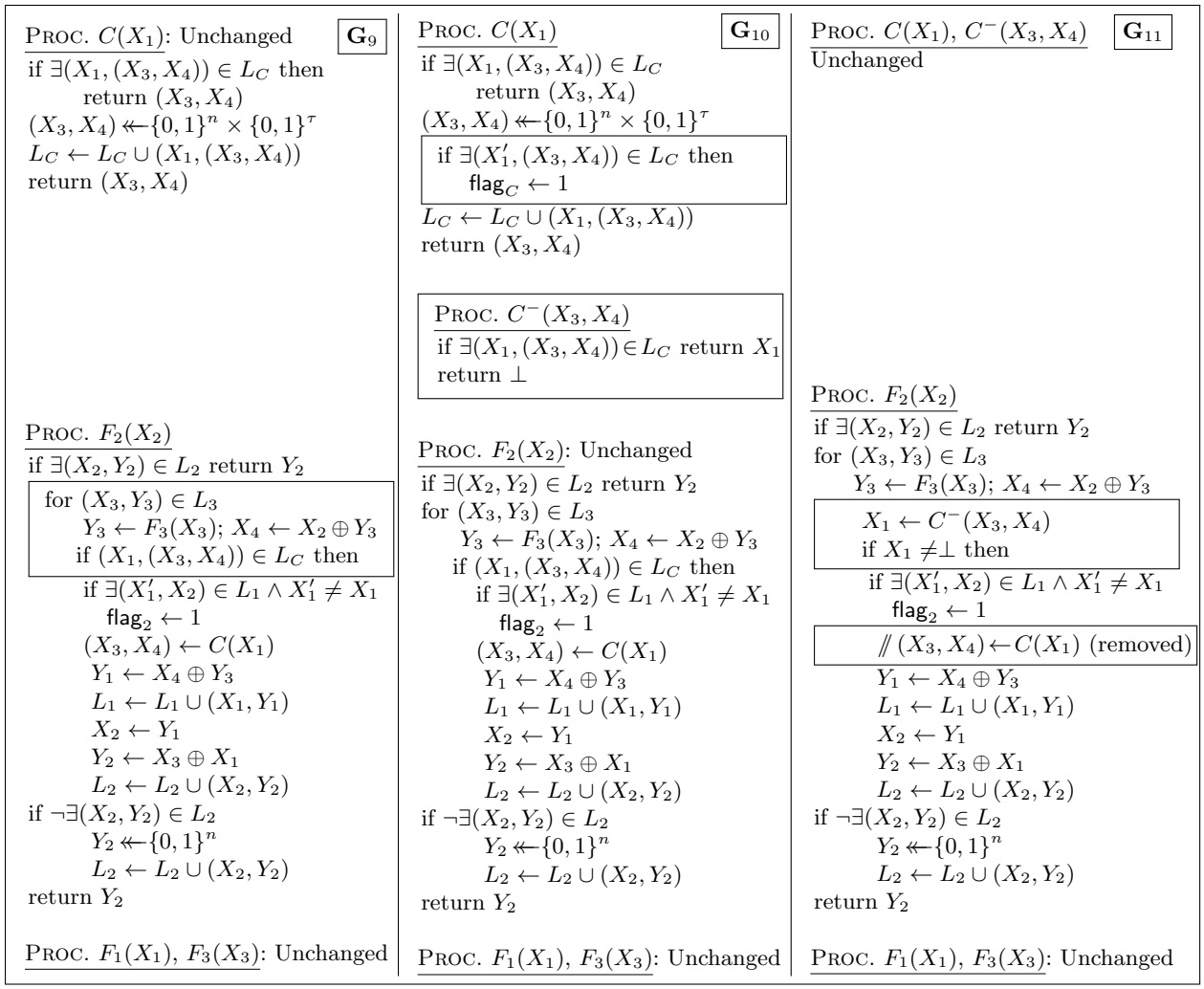


**Fig. 12.** Games $\mathbf{G}_9$, $\mathbf{G}_{10}$, and $\mathbf{G}_{11}$.

as before. The games are therefore identical until $\mathsf{flag}_2$ is set, the probability of which we bound via a birthday bound below.

$\mathbf{G}_7$ : This game introduces a conceptual change in the way the loops in $F_2$ are executed. First, all $X_3$ values corresponding to entries in $L_C$ are queried to $F_3$ if they were not previously done so. This means that the subsequent search for a good $Y_3$ can be equivalently made by going through those entries in $L_C$ whose $X_3$ value is *already* present in $L_3$. This change sets the ground for the next game where we drop the first loop completely.

$\mathbf{G}_8$ : We now remove the code that corresponds to the first loop in $F_2$ completely and argue that there is a rare event that allows us to prove the games identical until bad and bound the statistical distance between the two. This rare event is explicitly shown, for convenience, as a dummy $\mathsf{flag}_3$: it is activated whenever the first loop was adding to list $L_3$ a freshly sampled

entry $(X_3, Y_3)$, which is used by the second loop.[21] Again we can bound the probability of this event easily, as $F_3$ implements a random oracle.

$\mathbf{G}_9$ : This game rewrites the loops in $F_2$ and only looks in $L_C$ for values that will be used by $F_2$, i.e., only those entries with $X_4 = X_2 \oplus Y_3$ will be searched over. This is a conceptual change.

$\mathbf{G}_{10}$ : This game introduces $\mathsf{flag}_C$, which is set if collisions in the outputs of $C$ are found. This prepares us to modify the implementation of $C$ from a random function to a random injection. We bound this via a standard RF/RI switching lemma. This game also introduces a (partial and so far unused) inverse $C^-$ to $C$ that returns the preimage to $(X_3, X_4)$ if this value was queried to $C$. This will allow us to remove the dependency on the $L_C$ next. (Recall that the differentiator is restricted and it cannot call $C^-$ at all.)

$\mathbf{G}_{11}$ : In this game $F_2$ no longer uses $L_C$; instead it uses $C^-$ to check if a value was queried to $C$. Since this partial inverse oracle always returns $\perp$ for inputs that are not on $L_C$, this game is identical to the previous game. (Note also that we may also omit the re-computation of $(X_3, X_4)$.)

$\mathbf{G}_{12}$ : This game modifies $C$ to the forward direction of a random injection oracle and $C^-$ to its backward direction (which could return a non-$\perp$ value even if an inverse is not found in $L_C$). This modification can be bounded via an argument similar to the analysis of the ideal world under Lemma 1. Note that in Lemma 1 we were looking at inverse queries placed by the differentiator. In contrast, here we are analyzing the probability that the simulator places an inverse query that was not previously obtained from the forward construction oracle.

| Proc. $C(X_1)$ | Proc. $C^-(X_3, X_4)$ | Proc. $F_i(X_i)$: Unchanged $\boxed{\mathbf{G}_{12}}$ |
|---|---|---|
| $(X_3, X_4) \twoheadleftarrow \text{LazyRI}(X_1; L_C)$<br>Return $(X_3, X_4)$ | $X_1 \twoheadleftarrow \text{LazyRI}^-((X_3, X_4); L_C)$<br>Return $X_1$ | |

**Fig. 13.** Game $\mathbf{G}_{12}$.

Now observe that $\mathbf{G}_{12}$ is the ideal game where procedures $F_1$, $F_2$ and $F_3$ are implemented in a way that make use of random injection oracles $(C, C^-)$ but *not* its internal list $L_C$. By viewing the implementations of these procedures as three (sub-)simulators $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ as shown in Figure 14, we arrive at our simulator. We note that $\mathcal{S}_2$ omits $\mathsf{flag}_2$ in $F_2$ with no loss in advantage (cf. remark in the conservative jump to $\mathbf{G}_4$ above). We also note that this simulator is $C$-respecting as needed in Lemma 1 above.

We now formally bound the probabilities of setting the four flags in the game sequence above. Let $q_e, q_1, q_2$ and $q_3$ be, respectively, the number of queries of the restricted differentiator to $C, F_1, F_2$ and $F_3$.

$\mathsf{flag}_1$ **in game** $\mathbf{G}_1$ : This flag is set whenever in a non-repeat query to the construction oracle a value of $X_2$ is freshly sampled from the uniform distribution over $\{0,1\}^\tau$ and it happens that such a value already exists in $L_2$, a list of length at most $q_e + q_2$. The probability of this event occurring on each query to the construction is at most $(q_e + q_2)/2^\tau$, and $(q_e^2 + q_e q_2)/2^\tau$ overall.

[21] Alternatively, we can explicitly introduce $\mathsf{flag}_3$ in $\mathbf{G}_8$ and then immediately remove it (in a game $\mathbf{G}_{8.5}$); cf. the footnote in the conservative jump to $\mathbf{G}_4$ above.

| Algo. $\mathcal{S}_1^C(X_1)$ | Algo. $\mathcal{S}_2^{C^-}(X_2)$ | Algo. $\mathcal{S}_3(X_3)$ |
|---|---|---|
| if $\exists(X_1, Y_1) \in L_1$ return $Y_1$<br>$(X_3, X_4) \leftarrow C(X_1)$<br>$Y_3 \leftarrow \mathcal{S}_3(X_3)$<br>$Y_1 \leftarrow X_4 \oplus Y_3$<br>$L_1 \leftarrow L_1 \cup (X_1, Y_1)$<br>$X_2 \leftarrow Y_1$<br>$Y_2 \leftarrow X_3 \oplus X_1$<br>$L_2 \leftarrow L_2 \cup (X_2, Y_2)$<br>return $Y_1$ | if $\exists(X_2, Y_2) \in L_2$ return $Y_2$<br>for $(X_3, Y_3) \in L_3$<br>$X_4 \leftarrow X_2 \oplus Y_3$<br>$X_1 \leftarrow C^-(X_3, X_4)$<br>if $X_1 \neq \perp$ then<br>$L_1 \leftarrow L_1 \cup (X_1, Y_1)$<br>$X_2 \leftarrow Y_1$<br>$Y_2 \leftarrow X_3 \oplus X_1$<br>$L_2 \leftarrow L_2 \cup (X_2, Y_2)$<br>if $\neg\exists(X_2, Y_2) \in L_2$<br>$Y_2 \twoheadleftarrow \{0,1\}^n$<br>$L_2 \leftarrow L_2 \cup (X_2, Y_2)$<br>return $Y_2$ | if $\exists(X_3, Y_3) \in L_3$ return $Y_3$<br>$Y_3 \twoheadleftarrow \{0,1\}^\tau$<br>$L_3 \leftarrow L_3 \cup (X_3, Y_3)$<br>return $Y_3$ |

**Fig. 14.** Simulator for the 3-round Feistel construction in terms of three sub-simulators corresponding to the three round functions. The sub-simulators share lists $L_1$, $L_2$, and $L_3$ as joint state (which are initialized to empty). The for-loop and "if $\exists(X_i, Y_i) \in L_i$" commands go through the list in some well-defined order.

$\mathsf{flag}_2$ **in** $\mathbf{G}_6$ : This flag is set whenever two different values of $X_1$ queried to the construction, corresponding to distinct $L_C$ entries $(X_1, (X_3, X_4))$ and $(X_1', (X_3', X_4'))$ and corresponding $L_3$ entries $(X_3, Y_3)$ and $(X_3', Y_3')$ give rise to the same value $X_2 = X_4 \oplus Y_3 = X_4' \oplus Y_3'$. Since $X_1 \neq X_1'$ we know that $X_4$ and $X_4'$ are sampled independently of each other. This means that, even fixing $Y_3$ and $Y_3'$, the probability of this event occurring for two arbitrary entries in $L_1$ is exactly $1/2^\tau$. Since list $L_1$ is of size at most $q_1 + q_2$, we have that the overall probability of this event occurring is at most $(q_1 + q_2)^2/2^\tau$.

$\mathsf{flag}_3$ **in** $\mathbf{G}_8$ : This flag corresponds to the bad event that could arise when one removes explicit samplings from $F_2$ in $\mathbf{G}_8$. In particular, it corresponds to the probability that a freshly sampled $Y_3$ value happens to satisfy an equation $X_2 = X_4 \oplus Y_3$, where $X_2$ and $X_4$ are fixed. Since this event can occur at most once for each entry in $L_C$, the overall probability of this occurring is upper bounded by $q_e/2^\tau$.

$\mathsf{flag}_C$ **in** $\mathbf{G}_{10}$ : This event corresponds to collisions at the output of a random oracle of output size $n + \tau$ emulated inside the construction oracle in $\mathbf{G}_{10}$. The probability of it occurring can therefore be bounded by $q_e^2/2^{n+\tau}$.

$\mathbf{G}_{11}$ **to** $\mathbf{G}_{12}$ : We need to upper-bound the probability of a backward query to $C^-$ that inverts successfully, but was not returned by $C$. The maximum number of queries to $C^-$ is $q_2|L_3|$. Guessing a fresh and valid $(X_3, X_4)$ without obtaining it from $C$ can be done with probability at most $1/(2^{n+\tau} - q_2|L_3|)$; which brings the overall probability to at most $q_2|L_3|/(2^{n+\tau} - q_2|L_3|)$. Assuming $q_2|L_3| \leq 2^{n+\tau}/2$, this can be upper bounded by $2q_2|L_3|/2^{n+\tau}$. Note also that $|L_3| \leq q_1 + q_2 + q_3$.

**Variable change in** $\mathbf{G}_3$ : It remains to show that the (variable) change introduced in game $\mathbf{G}_3$ preserves the distributions on $Y_1$, $Y_2$ and $Y_3$ and $(X_3, X_4)$ in $\mathbf{G}_2$. In both games $F_3$ is an independent random oracle and both games use $F_3$ to compute $Y_3$. On a non-repeat query $X_1$ to $C$, uniform and independent $Y_1$ and $Y_2$ are generated in $\mathbf{G}_2$. The distributions of $Y_1$ and $Y_2$ in $\mathbf{G}_3$ are also uniform and independent as $X_4$ and $X_3$ are uniform and independent (and XORing with a constant acts as a permutation). Finally $\mathbf{G}_2$ sets $X_3 := X_1 \oplus Y_2$ and $X_4 := Y_1 \oplus Y_3$. This is also the case in $\mathbf{G}_3$ as

$$X_1 \oplus Y_2 = X_1 \oplus X_3 \oplus X_1 = X_3 \quad \text{and} \quad Y_1 \oplus Y_3 = X_4 \oplus Y_3 \oplus Y_3 = X_4 \ .$$

Collecting the terms above, we obtain an overall bound of

$$\frac{(q_e^2 + q_e q_2) + (q_1 + q_2)^2 + q_e}{2^\tau} + \frac{q_e^2 + 2q_2(q_1 + q_2 + q_3)}{2^{n+\tau}} \ .$$

Since $q_e + 2q_1 + q_2 + q_3 \leq q$, this bound simplifies to $6q^2/2^\tau$. We note that the simulator places at most $q^2$ oracle queries (it is quadratic due to the loop in $\mathcal{S}_2^{C^-}$). □

### 5.3 Removing restrictions and simplifications

Our AEAD schemes were analyzed with respect to differentiators that were bound to a fixed $(K, N, A, \tau)$. We deal with arbitrary $(K, N, A, \tau)$ by applying a hybrid argument over $(K, N, A, \tau)$ and their respective simulators. For this argument to hold, it is important to ensure that the simulators do not "interfere" with each other: not only should they be run on independent coins, but also their ideal AEAD oracles should be independent. We formalize this argument in a more general form.

From key-wise to full indifferentiability. We call a keyed ideal object $\mathcal{F}$ *uniformly keyed* if $\mathcal{F}(K, X)$ and $\mathcal{F}(K', X)$ are identically and independently distributed for any $X$ and distinct keys $K$ and $K'$. Let $\mathrm{C}^{\mathcal{F}_1}$ be a construction of a uniformly keyed object $\mathcal{F}_2$ from a uniformly keyed object $\mathcal{F}_1$. We call the construction key-respecting if for all inputs $(K, X)$ it queries $\mathcal{F}_1$ on $K$ only. (Note this means that the key spaces of the two objects are identical.) We call a simulator (for $\mathcal{F}_1$) key-respecting if for all inputs $(K, X)$ it queries $\mathcal{F}_2$ on $K$ only. We call a differentiator key-respecting if it always queries both the construction and the primitive oracles on $K$ only. We call the construction key-wise indifferentiable if it is indifferentiable with a key-respecting simulator against all key-respecting differentiators.

**Lemma 2 (Hybrid over keys).** *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two uniformly keyed objects and $\mathrm{C}^{\mathcal{F}_1}$ be a key-respecting construction of $\mathcal{F}_2$ from $\mathcal{F}_1$. Then if $\mathrm{C}^{\mathcal{F}_1}$ is key-wise indifferentiable, it is also (fully) indifferentiable. More precisely, for any key-respecting simulator $\mathcal{S}$ and any $q$-query (unrestricted) differentiator $\mathscr{D}$ there is a key-respecting differentiator $\mathscr{D}'$ such that*

$$\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D}) \leq q \cdot \mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D}') \ .$$

*Proof (Sketch).* By key-wise indifferentiability, for each value of $K$ the distributions $(\mathcal{F}_2(\cdot), \mathcal{S}^{\mathcal{F}_2}(K, \cdot))$ and $(\mathrm{C}_1^{\mathcal{F}}(K, \cdot), \mathcal{F}_1(K, \cdot))$ are close. Since $\mathcal{S}^{\mathcal{F}_2}(K, \cdot)$ and $\mathrm{C}_1^{\mathcal{F}}(K, \cdot)$ are both key-respecting and the ideal objects $\mathcal{F}_1(K, \cdot)$ and $\mathcal{F}_2(K, \cdot)$ are independently distributed for distinct keys $K$, the outputs of the real (resp., ideal) worlds for different values of $K$ are independent. This allows us to apply a hybrid argument over $K$ to conclude. Appendix D gives the details of the hybrid argument. □

In order to apply this result to the EtE transformation, it suffices to syntacticly express a variable input/key length ideal cipher (in both its forward and backward directions) as a *single* keyed primitive $\mathcal{F}_1$, the ideal AEAD as a keyed primitive $\mathcal{F}_2$, and then show they are key-respecting. To this end, we map the keys of the ideal cipher to the keys of $\mathcal{F}_1$ and treat the AEAD input components $(K, N, A, \tau)$ as the key to $\mathcal{F}_2$ (note that this is consistent with our presentation in Figure 6). To deal with inverse oracles, we append a bit to the inputs indicating the direction of the oracle that is being queried. Note that this bit cannot be appended to the key as otherwise the resulting ideal objects would *not* be independent for different keys; indeed, inverse and forward

oracles for the same object would be accessible via different keys, which would prevent applying the lemma. With respect to these (syntactic) implementations of the ideal cipher and the ideal AEAD, the EtE constructions and its associated simulator can be easily seen to be key-respecting.

For 3-round Feistel, we define $\mathcal{F}_1$ as having the three round functions inside it, and attaching 2 bits to the inputs to identify which one is being called. Similarly to EtE, we will treat the AEAD input components $(K, N, A, \tau)$ as the key to $\mathcal{F}_2$. Since the lemma requires the construction to be key-respecting, with these implementations of $F_i$ and the ideal AEAD, the *same* key $(K, N, A, \tau)$ has to be used in the three round functions (which are nevertheless independent). We note that imposing the key-respecting restriction across all round functions is not merely a technicality arising from our proof presentation. Indeed, without observing this requirement, differentiating attacks similar to those in Section 4 arise.

Dealing with keys of arbitrary size. Objects with an arbitrarily large key space can be indifferentiably built from those with a smaller key space in the standard way by hashing the key using a random oracle. This means we can remove the assumption of variable key lengths on the VIL ideal cipher in our construction. Although establishing this for the ideal cipher would suffice for our purposes, we prove a slightly more general result for any keyed ideal object. Once again, we will indicate the forward or backward (or other) directions of an oracle via bits attached to the inputs.

**Proposition 3 (Key extension via hashing).** *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two uniformly keyed ideal objects with key spaces $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. Let $\mathcal{H} : \mathcal{K}_2 \longrightarrow \mathcal{K}_1$ be a random oracle. Suppose further that for some (and hence any) $K_1 \in \mathcal{K}_1$ and $K_2 \in \mathcal{K}_2$ we have that $\mathcal{F}_1(K_1, X)$ is identically distributed to $\mathcal{F}_2(K_2, X)$. Then $\mathrm{C}^{\mathcal{F}_1,\mathcal{H}}(K, X) := \mathcal{F}_1(\mathcal{H}(K), X)$ is indifferentiable from $\mathcal{F}_2$. More precisely, there is a simulator $\mathcal{S}$ such that for any $q/3$-query differentiator $\mathscr{D}$,*

$$\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{F}_2}(\mathscr{D}) \leq 2q^2/|\mathcal{K}_1| \ .$$

*Proof.* Intuitively the simulator $\mathcal{S}_H$ for the random oracle performs lazy sampling. The simulator $\mathcal{S}_1^{\mathcal{F}_2}$ for $\mathcal{F}_1$ on an input $(K_1, X)$ attempts to look up a preimage for $K_1$ using the list for the random oracle. If a unique preimage $K_2$ is found, it answers the query via a call to $\mathcal{F}_2(K_2, X)$. There is, however, the possibility that either no key or more than one key is found. In the former case, whose probability can be bounded via the birthday bound, the simulator will return $\perp$. In the latter case, the differentiator is forming a "non-chain" query and the simulator will answer by simulating an independent instance of $\mathcal{F}_1$ via lazy sampling.[22] The pseudocode for this simulator $\mathcal{S}^{\mathcal{F}_2} := (\mathcal{S}_H, \mathcal{S}_1^{\mathcal{F}_2})$ is shown in Figure 15.

Let $\mathscr{D}$ be a differentiator. The views of $\mathscr{D}$ in the real and ideal world consists of tuples of the form

$$(\textsc{Const}, K_2, X_2, Y_2) \ , \quad (\textsc{Prim}_H, K_2', K_1') \ , \quad \text{and} \quad (\textsc{Prim}_1, K_1, X_1, Y_1) \ .$$

We call a triple of such entries a *chain* if for some $(K_2, X, Y_2, K_1, Y_1)$ it takes the form

$$(\textsc{Const}, K_2, X, Y_2) \ , \quad (\textsc{Prim}_H, K_2, K_1) \ , \quad (\textsc{Prim}_1, K_1, X, Y_1) \ .$$

We call a construction or primitive entry that does not belong to any chain an *isolated* query. We assume, without loss of generality, that any $(\textsc{Prim}_H, K_2, K_1)$ entry and any $(\textsc{Const}, K_2, X, Y_2)$

[22] The ability to lazily sample $\mathcal{F}_1$ does not pose a restriction on the generality of the theorem: by our assumption that $\mathcal{F}_1(K_1, X)$ is identically distributed to $\mathcal{F}_2(K_2, X)$ oracle $\mathcal{F}_2$ can be used to sample an independent instance of $\mathcal{F}_1$.

| Algo. $\mathcal{S}_H(K_2)$ | Algo. $\mathcal{S}_1^{\mathcal{F}_2}(K_1, X)$ |
|---|---|
| if flag return $\perp$ | if flag return $\perp$ |
| $(K_1; L_H) \twoheadleftarrow \text{LazyRO}(K_2, X; L_H)$ | if $\exists (K_2, K_1) \in L_H$ |
| if $K_1 \in L_H \cup L_1$ | return $\mathcal{F}_2(K_2, X)$ |
| flag $\leftarrow 1$ | $(Y_1; L_1) \twoheadleftarrow \text{LazyF1}(K_1, X; L_1)$ |
| return $K_1$ | return $Y_1$ |

**Fig. 15.** The simulator for converting a fixed-key-length ideal object to its variable-key-length counterpart. Here LazyRO denotes a lazy sampler for the random oracle and LazyF1 denote a lazy sampler for the ideal object $\mathcal{F}_1$. The two lists $L_H$ and $L_1$ are initialized to empty.

entry is in a chain. For example we can simply consider a differentiator $\mathscr{D}'$ that runs $\mathscr{D}$ and, after $\mathscr{D}$ has terminated and before returning its guess, completes all remaining chains by: (1) querying $\textsc{Const}(K_2, 0)$ and $\textsc{Prim}_1(K_1, 0))$ for any remaining isolated $(\textsc{Prim}_H, K_2, K_1)$ entries; and (2) completing chains for isolated $(\textsc{Const}, K_2, X, Y_2)$ entries by calling $\textsc{Prim}_H(K_2)$ to get $K_1$ and then querying $\textsc{Prim}_1(K_1, X)$. This at most triples the query complexity of $\mathscr{D}$, without changing its overall advantage, and it also implies that only queries to $\textsc{Prim}_1$ can be isolated.

Isolated entries. Consider the distribution of the outputs of an isolated $(\textsc{Prim}_1, K_1, X_1, Y_1)$ entry in the real world. These entries are distributed independently of the entries $(\textsc{Prim}_H, K_2', K_1')$. Furthermore, the output $Y_1$ is also distributed independently of all $Y_2$ that appear in an entry $(\textsc{Const}, K_2, X_2, Y_2)$. This means that the distribution of $Y_1$ in an isolated $(\textsc{Prim}_1, K_1, X_1, Y_1)$ entry follows that induced by $\mathcal{F}_1(K_1, X_1)$.

Consider now the distribution of the simulated outputs for an isolated $(\textsc{Prim}_1, K_1, X_1, Y_1)$ entry in the ideal world. Once again, by the construction of the simulator, these are independent of outputs for $\textsc{Prim}_H$. Moreover, since no $(K_2, K_1)$ exists in $L_H$ (the entry is isolated) it must be the case that $Y_1$ was lazily sampled. It follows that the distribution of $Y_1$ in $(\textsc{Prim}_1, K_1, X_1, Y_1)$ is identical to $\mathcal{F}_1(K_1, X_1)$.

Real chained entries. The first and last output entries of a chain in the real world always satisfy $Y_2 = Y_1$, which is distributed according to $\mathcal{F}_1$. The distribution of the output of the hash, $K_1$, is random and independent of all other values. We now consider a modified version of the real world where the distribution of isolated entries is as before, but the distribution of chained entries, as we will show below, is exactly matched by our simulator. In this modified real world, the primitive oracles return $\perp$ if a collision in the outputs of $\mathcal{H}$ is found, or if an entry $(\textsc{Prim}, K_1, X, Y_1)$ appears *before* the entry $(\textsc{Prim}_2, K_2, K_1)$ does. It is easily seen that the probability of both events is upper-bounded by the birthday bound $q^2/|\mathcal{K}_1|$. This means that, for every $q$-query attacker $\mathscr{D}$, the statistical distance between the original and modified worlds is at most $2q^2/|\mathcal{K}_1|$.

Ideal chained entries. We show that the distribution of the simulated view and that of the *modified* real world are identical. Observe that the simulator returns $\perp$ when a collision in $L_H \cup L_1$ is found, i.e., when the bad flag is set. This could happen as a result of two $\textsc{Prim}_H$ queries or as a result of a $\textsc{Prim}_1$ query followed by a $\textsc{Prim}_H$ query. The modified real world, however, also returns $\perp$ in these scenarios and the simulation is perfect in these settings. Conditioned on flag not set, it is easy to see that $\mathcal{S}_H$ simulates a random oracle. Furthermore, if flag is not set, the simulation of $\mathcal{F}_1$ on a chained entry $(K_1, X)$ would always recover a $K_2$ (as otherwise flag would have been set) and hence $Y_1 := \mathcal{F}_2(K_2, X)$. By our assumption this is identically distributed to $\mathcal{F}_1(K_1, X)$. Hence the simulation is also perfect when flag is not set. □

THE FULL CONSTRUCTION. Our final AEAD construction can be written as $\mathcal{AE}(K, N, A, M, \tau) = \Phi_3(K', M)$, where $K' = \mathcal{H}(K, N, A, \tau)$ and $\Phi_3$ is the ideal injection instantiated with 3-round Feistel. The latter uses independent keyed random oracles $\mathcal{F}_i$ all with key space $\mathcal{K}$ matching the co-domain of $\mathcal{H}$. Combining Theorem 5 with Lemmas 2 and 3 we obtain an overall bound $9q^3/2^\tau + 2q^2/|\mathcal{K}|$, where $q$ is an upper bound on the number of oracles queries.

## 6 Efficiency Lower Bounds

Suppose we instantiate the random oracles underlying our Feistel-based construction with the Sponge construction [BDPV08]. Suppose also that the underlying Sponges absorb inputs and expand outputs in blocks of $n$ bits (i.e., the Sponge has bit-rate $n$). Finally, assume that our input message is $w$ blocks long. This means that in both of our constructions roughly $w$ primitive calls are used in each round of Feistel. Indeed, when using a balanced Feistel network, each round function will consume and expand $w/2$ blocks, whereas in the 3-round construction each round alternatively uses a compressing and an expanding version of Sponge for $w$ blocks. This adds up to $3w$ overall primitive calls for the second construction and $8w$ calls for the first one. Our second construction is therefore almost 3 times faster than the first.

We next show that the efficient construction is not too far from the theoretically optimal solution by proving that at least $2w$ calls are necessary for *any* indifferentiable construction. We do this by first giving a lower bound for indifferentiable constructions of random oracles (which is tight as it is essentially matched by Sponge) and then show how to derive the lower bound for random injections from it.

Figure 16 shows a schematic diagram of a general construction of a length-preserving random oracle from a random permutation $\pi$. The functions $f_i$ represent how the construction manages internal state and prepares the next call to the permutation. Overall the construction makes $q$ queries to transform a $w$-block input into a $w$-block output. The permutation $\pi$ operates on a single $n$-bit block.

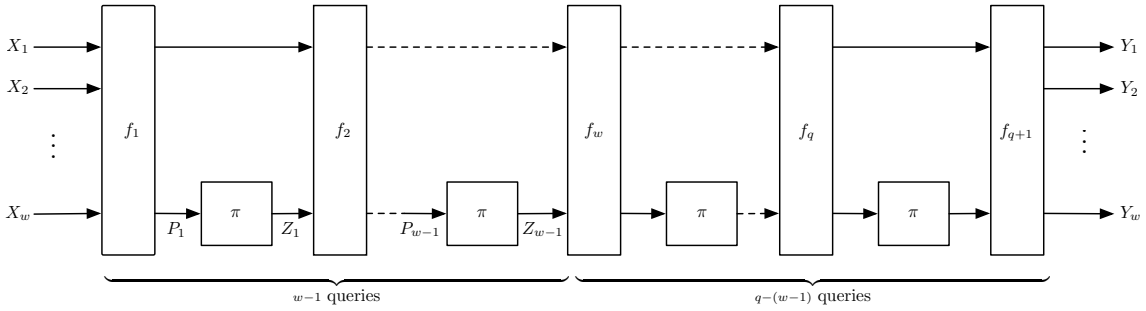


**Fig. 16.** The structure of a general construction of a $wn$-bit random oracle from an $n$-bit permutation $\pi$.

The following theorem establishes a relation between a lower bound on $q$ and the number of input blocks $w$ for any indifferentiable construction of a random oracle.

**Theorem 6 (Efficiency lower bound).** *Any indifferentiable construction of a random function* $\mathrm{C}^\pi : \{0,1\}^{wn} \longrightarrow \{0,1\}^{wn}$ *from a random permutation* $\pi : \{0,1\}^n \longrightarrow \{0,1\}^n$ *must place at least*

$q \geq 2w-2$ *queries to* $\pi$. *More precisely, for any such* $q$*-query construction* $\mathrm{C}^\pi$ *and any* $q_S$*-query indifferentiability simulator* $\mathcal{S}$ *there is a* $w$*-query differentiator* $\mathscr{D}$ *such that*

$$2 \cdot \mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D}) \geq 1 - 1/2^{(q-(2w-2))n} - (q^2 + q_S)/2^n \ .$$

*Proof.* We prove this result by constructing a differentiator against any construction $\mathrm{C}^\pi$ that places $q < 2w-2$ queries to $\pi$. Any such $\mathrm{C}^\pi$ can be written in the form shown in Figure 16 for ($\pi$-independent) functions $f_1, \ldots f_{q+1}$ where

$$f_i : \{0,1\}^{(w+i-1)n} \longrightarrow \{0,1\}^{(w+i)n} \quad \text{for} \quad 1 \leq i \leq q \ ,$$
$$f_{q+1} : \{0,1\}^{(w+q)n} \longrightarrow \{0,1\}^{wn} \ .$$

This structure reflects the fact that each $f_i$ can recompute everything that depends only on the initial inputs, but also needs to take as additional inputs the values returned by $\pi$ at each of the previous calls.

Consider the first $w-1$ calls to $\pi$. There are $2^{(w-1)n}$ possible tuples $\mathrm{P} = (P_1, \ldots, P_{w-1})$ that can define the inputs to such queries. Since in total there are $2^{wn}$ possible inputs, by a counting argument, a subset $\mathrm{D}[\mathrm{C},\pi]$ of the input values of size at least $2^n = 2^{wn}/2^{(w-1)n}$ will be mapped by a construction $C$ to the same $\mathrm{P}[\mathrm{C},\pi]$, for any given $\pi$. Set $\mathrm{D}[\mathrm{C},\pi]$ and points $\mathrm{P}[\mathrm{C},\pi]$ can be found by a (possibly unbounded) attacker $\mathscr{D}$ using only $w-2$ queries to $\pi$. Algorithm $\mathscr{D}$ proceeds in rounds as follows. There is at least one point $P_1 \in \{0,1\}^n$ such that $f_1$ always chooses $P_1$ for at least $2^{wn}/2^n = 2^{(w-1)n}$ of its inputs. No queries to $\pi$ are needed to find $P_1$ and we set $\mathrm{D}[\mathrm{C},\pi]$ to a corresponding set of colliding inputs. We then get $Z_1 := \pi(P_1)$ and we use it to analyze the operation of $f_2$. Given $Z_1$ and $\mathrm{D}[\mathrm{C},\pi]$, at least $2^{(w-1)n}/2^n$ of the inputs in $\mathrm{D}[\mathrm{C},\pi]$ are such that $f_2$ always chooses the same query point $P_2$ to $\pi$. We update $\mathrm{D}[\mathrm{C},\pi]$ to this subset. Continuing in this manner, we obtain a set $\mathrm{D}[\mathrm{C},\pi]$ of at least $2^n$ points such that $f_{w-1}$ chooses a point $P_{w-1}$ for all inputs in $\mathrm{D}[\mathrm{C},\pi]$.

Put together, the restriction of $\mathrm{C}^\pi$ to inputs in $\mathrm{D}[\mathrm{C},\pi]$ guarantees that the construction always queries $\pi$ at $P_i$ for queries $i = 1, \ldots, w-1$ and then places an arbitrary sequence of $q-(w-1)$ queries to $\pi$. Furthermore, from the previous discussion we can assume that the description of set $\mathrm{D}[\mathrm{C},\pi]$ and values

$$\mathrm{Z}[\mathrm{C},\pi] := (Z_1, \ldots, Z_{w-1}) = (\pi(P_1), \ldots, \pi(P_{w-1}))$$

are known to $\mathscr{D}$. If $\mathscr{D}$ can distinguish the output of PRG from a random string, this will allow differentiating $C^\pi$ from a random function. We now show that such an attack is guaranteed to exist if $C$ does not make a sufficient number of queries to $\pi$.

Now consider a pseudorandom generator $\mathrm{PRG} : \mathrm{D}[\mathrm{C},\pi] \times \{0,1\}^{(q-(w-1))n} \longrightarrow \{0,1\}^{wn}$ that has $\mathrm{Z}[\mathrm{C},\pi]$ hardwired in and operates as

$$\mathrm{PRG}[\mathrm{Z}[\mathrm{C},\pi]](X, Z_w, \ldots, Z_q) := \mathrm{C}^{Z_1,\ldots,Z_q}(X) \ ,$$

where $\mathrm{C}^{Z_1,\ldots,Z_q}(X)$ denotes running $\mathrm{C}^\pi(X)$ and answering the $i$-th query with $Z_i$. It is at this step that we follow the techniques of Gennaro and Trevisan [GT00].

We will first prove that if $\mathrm{C}^\pi$ is indifferentiable then $\mathrm{PRG}[\mathrm{Z}[\mathrm{C},\pi]]$ is a secure pseudorandom generators over a random choice of $\pi$. More precisely, our goal is to show that under the indifferentiability of $\mathrm{C}^\pi$, the following distributions are statistically close.

$$(\mathrm{PRG}[\mathrm{Z}[\mathrm{C},\pi]](X, Z_w, \ldots, Z_q), \mathrm{Z}[\mathrm{C},\pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C},\pi];\ Z_w, \ldots, Z_q \twoheadleftarrow \{0,1\}^{(q-(w-1))n}$$
$$(Y, \mathrm{Z}[\mathrm{C},\pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ Y \twoheadleftarrow \{0,1\}^{wn}$$

The points in $\mathrm{Z}[\mathrm{C}, \pi]$ are computed using oracle access to $\pi$ at the onset and, being part of the description of the PRG, are in the view of a PRG distinguisher. To prove this claim, we first observe that the distributions

$$\begin{aligned}\mathrm{PRG}[\mathrm{Z}[\mathrm{C}, \pi]](X, Z_w, \ldots, Z_q) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \pi];\ Z_w, \ldots, Z_q \twoheadleftarrow \{0,1\}^{(q-(w-1))n} \\ \mathrm{C}^{\pi}(X) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \pi]\end{aligned}$$

are statistically close. To see this, note that the simulation of $\pi$ using $Z_i$ is fully consistent for queries $i = 1, \ldots, w-1$. This is also the case for $i \geq w$ unless $Z_1, \ldots, Z_q$ are not all distinct, which by the birthday bound occurs with probability at most $q^2/2^n$.

Our goal can now be restated as showing that the following distributions to be statistically close.

$$\begin{aligned}(\mathrm{C}^{\pi}(X), \mathrm{Z}[\mathrm{C}, \pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \pi] \\ (Y, \mathrm{Z}[\mathrm{C}, \pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ Y \twoheadleftarrow \{0,1\}^{wn}\end{aligned}$$

Here we cannot directly apply indistinguishability of $\mathrm{C}^{\pi}(X)$ from a truly random $wn$-bit function $\mathcal{H}(X)$ (which follows from indifferentiability) as the hardwired values $\mathrm{Z}[\mathrm{C}, \pi]$ are in the distinguisher's view. Instead we proceed via a sequence of games as follows. First, we use the indifferentiability simulator $\mathcal{S}$ to deduce that the following distributions are statistically close.

$$\begin{aligned}(\mathrm{C}^{\pi}(X), \mathrm{Z}[\mathrm{C}, \pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \pi] \\ (\mathcal{H}(X), \mathrm{Z}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]) : \mathcal{H} \twoheadleftarrow \mathrm{Fun}[wn, wn];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]\end{aligned}$$

This follows directly from the definition of indifferentiability. Consider a differentiator that constructs $\mathrm{Z}[\mathrm{C}, \textsc{Prim}]$ and $\mathrm{D}[\mathrm{C}, \textsc{Prim}]$ using the real or simulated $\pi$-oracle $\textsc{Prim}$, then queries its real or ideal construction oracle on $X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \textsc{Prim}]$ to obtain the first component above. Any successful distinguisher for the above distributions could be used by this differentiator to contradict the indifferentiability assumption with the same advantage. Note also that this differentiator places exactly $w$ queries ($w-1$ queries to the real or simulated $\pi$-oracle $\textsc{Prim}$ to construct $\mathrm{Z}[\mathrm{C}, \textsc{Prim}]$ and one additional query to the real or ideal construction oracle). Note that this argument also shows that $\mathrm{D}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]$ must also have at least $2^n$ points.

The next step in the proof is to show that we can replace $\mathcal{H}(X)$ with $Y$ for an *independently sampled* random string $Y$ that is *not* computed via the random oracle. More precisely, we argue that the following distributions are statistically close.

$$\begin{aligned}(\mathcal{H}(X), \mathrm{Z}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]) : \mathcal{H} \twoheadleftarrow \mathrm{Fun}[wn, wn];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}] \\ (Y, \mathrm{Z}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]) : \mathcal{H} \twoheadleftarrow \mathrm{Fun}[wn, wn];\ Y \twoheadleftarrow \{0,1\}^{wn}\end{aligned}$$

Suppose $\mathcal{S}$ places at most $q_S$ queries to $\mathcal{H}$. The set $\mathrm{D}[\mathrm{C}, \pi]$ has size at least $2^n$ and hence so does the set $\mathrm{D}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]$. Now since $X$ is chosen uniformly at random from $\mathrm{D}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]$, the simulator $\mathcal{S}$ will query $\mathcal{H}$ on $X$ with probability at most $q_S/2^n$. Hence $\mathcal{H}(X)$ is independent of the simulators view and we may replace it with independent random value $Y$.

Finally, we use indifferentiability once more to show that we can replace $\mathrm{Z}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]$ back by $\mathrm{Z}[\mathrm{C}, \pi]$ in the presence of the independently sampled random string $Y$. The differentiator we construct uses the real or simulated $\pi$-oracle $\textsc{Prim}$ to construct set $\mathrm{Z}[\mathrm{C}, \pi]$ or $\mathrm{Z}[\mathrm{C}, \mathcal{S}^{\mathcal{H}}]$, respectively, and then

samples value $Y$. Again, any successful distinguisher for the above distributions will be translated into a differentiating attack with the same advantage, resulting in a successful differentiator that places exactly $w-1$ queries.

We have therefore established that the distributions

$$(\mathrm{PRG}[\mathrm{Z}[\mathrm{C},\pi]](X,Z_w,\ldots,Z_q),\mathrm{Z}[\mathrm{C},\pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ X \twoheadleftarrow \mathrm{D}[\mathrm{C},\pi];\ Z_w,\ldots,Z_q \twoheadleftarrow \{0,1\}^{(q-(w-1))n}$$
$$(Y,\mathrm{Z}[\mathrm{C},\pi]) : \pi \twoheadleftarrow \mathrm{Perm}[n];\ Y \twoheadleftarrow \{0,1\}^{wn}$$

are within statistical distance $(q^2+q_S)/2^n+2\delta$, where $\delta$ is the maximum advantage $\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D})$ over all $\mathscr{D}$ placing at most $w$ queries. This concludes the proof that if $\mathrm{C}^\pi$ is indifferentiable from a random oracle then PRG is secure over seed space $\mathrm{D}'[\mathrm{C},\pi] := \mathrm{D}[\mathrm{C},\pi]\times\{0,1\}^{(q-(w-1))n}$ (of overall size at least $2^{(q-w+2)n}$) and range $\mathrm{R} := \{0,1\}^{wn}$.

We now show that, unless $\mathrm{C}^\pi$ makes a sufficient large number of queries to $\pi$ the above PRG cannot be secure. The number of queries of $\mathrm{C}^\pi$ translates to the size of the seed space of PRG. Note that PRG does not make any queries to $\pi$ (beyond the initial $w-1$ queries used to hardwire the fixed $\mathrm{Z}[\mathrm{C},\pi]$ values). However, the outputs of any PRG with domain $\mathrm{D}'[\mathrm{C},\pi]$ and range R can be information-theoretically distinguished from random with advantage $1-|\mathrm{D}'[\mathrm{C},\pi]|/|\mathrm{R}|$: simply enumerate all the $|\mathrm{D}'[\mathrm{C},\pi]|$ possible outputs of PRG and check if the challenge string lies in this set. When the challenge is an output of PRG, this check will pass with probability 1. When the challenge is a random element in R, the check will pass with probability at most $|\mathrm{D}'[\mathrm{C},\pi]|/|\mathrm{R}|$. Hence, we must have that

$$1-|\mathrm{D}[\mathrm{C},\pi]\times\{0,1\}^{(q-(w-1))n}|/|\{0,1\}^{wn}| \le (q^2+q_S)/2^n+2\delta\ .$$

This proves the theorem and assuming $q^2+q_S\le 2^n/2$, the inequality simplifies to

$$(q-w+2)n-wn \ge -\log(\delta) \iff q \ge 2w-2-\log(\delta)/n\ .$$

Hence, if $\mathrm{C}^\pi$ indifferentiable we must have that $q\ge 2w-2$.[23] □

The above lower bound is essentially tight for random functions as the Sponge construction meets the bound up to constant terms. (Or put differently, the above result establishes the optimality of the Sponge construction.) The proof, however, does not directly apply to random injections $\rho$, as the inverse oracle $\rho^-$ would allow an adversary to invert the outputs of the PRG. The next proposition shows that by chopping sufficiently many bits of the outputs of $\rho$, a random function can be *indifferentiably* obtained from a random injection in a *single* query. Together with the above result this extends the lower bound to random injections as well.

**Proposition 4.** *Let $\rho : \{0,1\}^{wn} \longrightarrow \{0,1\}^{wn+n}$ be a random injection with inverse $\rho^-$. Let $\mathrm{C}^\rho(X) := \rho(X)[1..wn]$ be a construction that outputs bits 1 through $wn$ of $\rho(X)$. Then $\mathrm{C}^\rho$ is indifferentiable from a random function $\mathcal{H} : \{0,1\}^{wn} \longrightarrow \{0,1\}^{wn}$.*

[23] We note that by restricting the PRG to a subset of $\mathrm{D}[\mathrm{C},\pi]$ of size $2^{\alpha n}$, for any constant $\alpha\in(0,1]$, we can show $q\ge 2w-(1+\alpha)-\log(\delta)/n$, which slightly improves the lower bound to $q\ge 2w-1$ for $\alpha := 1/2$.

*Proof.* Consider the simulator $(\mathcal{S}_+^{\mathcal{H}}, \mathcal{S}_-^{\mathcal{H}})$ given in Figure 17. We show that

$$(\mathrm{C}^{\rho}, \rho, \rho^{-}) \approx (\mathcal{H}, \mathcal{S}_+^{\mathcal{H}}, \mathcal{S}_-^{\mathcal{H}}) \ ,$$

for any differentiator $\mathscr{D}$ that, without loss of generality, does not make repeat queries to any of the oracles.

| Algo. $\mathcal{S}_+^{\mathcal{H}}(X)$ | Algo. $\mathcal{S}_-^{\mathcal{H}}(R\|T)$ |
|---|---|
| $R \twoheadleftarrow \mathcal{H}(X)$ | if $\exists (X, R\|T) \in L$ |
| $T \twoheadleftarrow \{0,1\}^n$ | return $X$ |
| $L \leftarrow L \cup (X, R\|T)$ | return $\perp$ |
| return $R\|T$ | |

**Fig. 17.** Simulator for converting a random injection to a random function.

We first analyze the real world execution and identify a rare event: $\mathscr{D}$ queries a value $R|T$ to $\rho^{-}$ that is successfully inverted but $R$ was not previously obtained from a query to $\rho$. By a union bound, this event occurs with probability at most $q/2^n$ for a $q$-query $\mathscr{D}$ as the length of $T$ is $n$. Note that the probability of this event in the simulated world is zero. We therefore condition on this event and consider the simulation of $\rho$ only.

Without loss of generality we assume that $\mathscr{D}$ calls the construction oracle on any query that it submits to the forward primitive oracle. Hence the view of $\mathscr{D}$ in the real world consists of values $(Y, Y|Y')$ where $Y|Y'$ is obtained via a random injection. The view of $\mathscr{D}$ in the ideal world consists of values $(R, R|T)$ where $R$ and $T$ are obtained via two independent random oracles. The statistical distance between $Y|Y'$ and $R|T$ (and hence the two views) is bounded above by $q^2/2^{(w+1)n}$, as the real and simulated world are identical until two simulated outputs of the form $R|T$ collide. This results in an overall statistical distance of $q/2^n + q^2/2^{(w+1)n}$ between the simulated and real views. □

Our construction of random injections in the previous section via the 3-round Feistel construction places $3w + O(1)$ queries to $\pi$. This is somewhat higher than the $2w - 2$ required by the lower bound. We leave bridging this gap between the lower and upper bounds for random injections (and indeed also permutations) as the main open problem in this area. The possibilities are that: (1) an optimized construction of rate 2 exists; (2) the lower bound can be strengthen to $3w - o(w)$; or (3) the lower bound can be strengthened to $\rho w - o(w)$ for a $\rho \in (2, 3)$ and a more efficient construction that (essentially) meets this bound and places at most $\rho w + o(w)$ queries to $\pi$ exits. We emphasize that since Sponge meets the bound, a lower bound for random injections with $\rho \geq 2$ must exploit the invertibility of the construction, a fact that we did *not* use in our proof.

## 7 Ideal Online AEAD

Offline AEAD schemes can fall short of providing adequate levels of functionality or efficiency in settings where data arrives one segment at a time and should be processed immediately without the knowledge of future segments. In an *online* AEAD scheme, the encryption and decryption algorithms are replaced by segment-oriented ones that process the inputs one segment at a time. We formalize the syntax and security of online AEAD schemes next.

### 7.1 Online primitives and reference objects

We follow [HRRV15, BMM+15] in our treatment, but deviate from it in a number of details that have implications for relating the ideal objects used in the definitions to practical constructions.

Syntax. An online AEAD scheme is a seven tuple of algorithms

$$\Pi = (\mathcal{K}, \mathcal{AE}.\text{init}, \mathcal{AE}.\text{next}, \mathcal{AE}.\text{last}, \mathcal{AD}.\text{init}, \mathcal{AD}.\text{next}, \mathcal{AD}.\text{last}) \ .$$

Here $\mathcal{K}$ is the key-generation algorithm as in offline schemes and the remaining algorithms have signatures as shown below.

$$\begin{array}{ll} \mathcal{AE}.\text{init} : \mathcal{K} \times \mathcal{N} \longrightarrow \mathcal{S} & \mathcal{AD}.\text{init} : \mathcal{K} \times \mathcal{N} \longrightarrow \mathcal{S} \\ \mathcal{AE}.\text{next} : \mathcal{S} \times \mathcal{H} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C} \times \mathcal{S} & \mathcal{AD}.\text{next} : \mathcal{S} \times \mathcal{H} \times \mathcal{C} \times \mathcal{X} \longrightarrow (\mathcal{M} \times \mathcal{S}) \cup \{\perp\} \\ \mathcal{AE}.\text{last} : \mathcal{S} \times \mathcal{H} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C} & \mathcal{AD}.\text{last} : \mathcal{S} \times \mathcal{H} \times \mathcal{C} \times \mathcal{X} \longrightarrow \mathcal{M} \cup \{\perp\} \end{array}$$

Here $\mathcal{S}$ is some state space, and other spaces are interpreted as in offline AEAD schemes.[24] Note that a single nonce is provided for the entire sequence of segments. In contrast to [HRRV15], who focus on a fixed expansion $\tau$, our definitions allow for varying expansions across the segments. For instance the scheme might be configured to not provide any authenticity until the last segment is reached, at which point the entire stream is authenticated. We assume that $\mathcal{M} = \{0,1\}^*$, and denote by $\{0,1\}^{**} := (\{0,1\}^*)^*$ the set of *segmented-strings*. A segmented string $\boldsymbol{X} \in \{0,1\}^{**}$ is a *vector* of strings.

Correctness. The correctness of an online AEAD scheme $\Pi$ is defined via an *associated* offline AEAD scheme with encryption and decryption algorithms $\boldsymbol{\mathcal{AE}}$, $\boldsymbol{\mathcal{AD}} : \mathcal{K} \times \mathcal{N} \times \{0,1\}^{**} \times \mathcal{X}^* \longrightarrow \{0,1\}^{**}$ that operate iteratively on segmented strings. Algorithm $\boldsymbol{\mathcal{AD}}(K, N, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\tau})$ returns the longest $\boldsymbol{M}$ whose encryption (using $K$, $N$, $\boldsymbol{A}$, and $\boldsymbol{\tau}$) is a prefix of $\boldsymbol{C}$. (See Figure 25 in Appendix E.1.) We require the following correctness condition for an online AEAD scheme: if $K \in \mathcal{K}$, $N \in \mathcal{N}$, $\boldsymbol{A} \in \{0,1\}^{**}$, $\boldsymbol{M} \in \{0,1\}^{**}$, $\boldsymbol{\tau} \in \mathcal{X}^*$, and $\boldsymbol{C} = \boldsymbol{\mathcal{AE}}(K, N, \boldsymbol{A}, \boldsymbol{M}, \boldsymbol{\tau})$, then $\boldsymbol{M} = \boldsymbol{\mathcal{AD}}(K, N, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\tau})$.

Memory, state, efficiency. An online algorithm, as understood commonly, processes its input one segments at a time, possibly depending on the past segments, but without the knowledge of the future ones. A state value is forwarded in each invocation of the algorithm to its next stage. HRRV [HRRV15] take a different approach and define online computability in terms of the size of the working memory and state used.[25] We agree with HRRV that an *efficient* online AEAD scheme should use both a reasonable amount of working memory and state size. Conversely, small state usage implies that an algorithm is online in the usual sense: if the state cannot grow, one cannot defer processing until future segments arrive. Hence the two approaches match with respect to efficient AEAD schemes. This, however, raises the question of what an adequate ideal reference object for online AEAD is. HRRV define this as essentially an offline AEAD whose invocations are tweaked based on the *entire* history of prior inputs. (See Figure 4 in [HRRV15].) At first sight this may look like the "right" choice, as one would expect that an ideal scheme "remembers" everything that has been processed so far. This notion can be formalized by a function that stores the history in its state. This approach, however, is not compatible with our efficiency requirements above. Indeed, in the indifferentiability setting, one must give complete control over all interfaces of the scheme to the adversary. This, in particular, includes the input/output states, which means that an indifferentiable scheme would forcefully also have a growing state where all prior history was kept. Instead, and consistently with efficiency requirements, we require that the state value of the ideal object acts as a *small and random summary* of the inputs processed so far. A second benefit

[24] In our definition both encryption and decryption are online. One can also consider other variants where only one of these algorithms is online. The practical use-cases for these primitives, however, seem to be limited.

[25] The authors write: "We say that an online AEAD scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *online-encryption* if its state space $\mathcal{S}$ is finite and, additionally, there's a constant $w$ such that $\mathcal{E}$.next and $\mathcal{E}$.last use at most $w$ bits of working memory." They end with: "Our security definitions don't care if a segmented-AE scheme is online: that's an efficiency requirement, not a security requirement."

of this approach is that under a threat model that allows for state revelation, an indifferentiable scheme, according to our ideal objects, would retain maximal security. This stands in contrast to a scheme that stores its history in the clear, which fails to protect the confidentiality of the previously processed plaintexts as soon as the state is revealed.

ONLINE FUNCTIONS. An online function(ality) is a triple of functions with signatures

$$\mathcal{F}_0 : \mathcal{A}_0 \longrightarrow \mathcal{S} \ , \qquad \mathcal{F}_1 : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}_1 \times \mathcal{S} \ , \qquad \mathcal{F}_2 : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}_2 \ .$$

We allow for different auxiliary spaces and ranges above for the sake of generality. We define $\mathrm{Onj}[\mathcal{A}_0, \mathcal{A}, \mathcal{M}, \mathcal{X}, \mathcal{S}, \mathcal{R}_1, \mathcal{R}_2]$ as the set of online functions for which $\mathcal{F}_1$ and $\mathcal{F}_2$ are injective over $\mathcal{M}$ and respect the usual length-expansion requirement. An online injection gives rise to a unique online inversion $\mathcal{F}_1^-$ associated to $\mathcal{F}_1$ that satisfies

$$\begin{aligned} \forall (S, A, M, \tau, C, S') : \mathcal{F}_1(S, A, M, \tau) = (C, S') &\iff \mathcal{F}_1^-(S, A, C, \tau) = (M, S') \\ \forall (S, A, M, \tau, C, S') : (\nexists (M, S') : \mathcal{F}_1(S, A, M, \tau) = (C, S')) &\iff \mathcal{F}_1^-(S, A, C, \tau) = \ \perp \ . \end{aligned}$$

Similarly a unique inversion $\mathcal{F}_2^-$ corresponds to $\mathcal{F}_2$. We can associate to $(\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2)$ an offline function on vectors of segments. This offline function starts by running $\mathcal{F}_0$, then processes each segment up to and including the penultimate one using $\mathcal{F}_1$, and terminates by processing the final segment using $\mathcal{F}_2$. Similarly, we define the associated offline inversion for $(\mathcal{F}_0, \mathcal{F}_1^-, \mathcal{F}_2^-)$, which is also an inversion for the offline function associated to $(\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2)$. We emphasize that the propagation of state is identical in $\mathcal{F}_1$ and $\mathcal{F}_1^-$ and hence the ideal objects for this set of functions will all have this property. Finally, we note that the distribution induced on offline functions through this association is *not* the uniform one on the set of all offline functions. Indeed, the extra functionality offered by online schemes has security implications. For example any online AEAD scheme will preserve prefixes at the segment level and hence can never meet the level of security enjoyed by offline schemes [HRRV15, BMM+15].

IDEAL ONLINE AEAD. An online AEAD is a uniform function in $\mathrm{Onj}[\mathcal{A}_0, \mathcal{A}, \mathcal{M}, \mathcal{X}, \mathcal{S}, \mathcal{R}_1, \mathcal{R}_2]$ where $\mathcal{A}_0 := \mathcal{K} \times \mathcal{N}$, $\mathcal{A} := \mathcal{H}$, and $\mathcal{R}_1 := \mathcal{R}_2 := \mathcal{C}$. Given an online AEAD scheme $\Pi$, we define $\mathrm{OAE}[\Pi]$ as the set of all expanding online injections with signatures that match those of $\Pi$.

ORAE SECURITY. We define the online RAE (ORAE) security of an online AEAD scheme by requiring indistinguishability from an ideal online AEAD under a random key. Let

$$\mathbf{Adv}^{\mathrm{orae}}_{\Pi}(\mathscr{A}) := \Pr\left[\mathbf{ORAE\text{-}Real}^{\mathscr{A}}_{\Pi}\right] - \Pr\left[\mathbf{ORAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}\right] \ ,$$

where games $\mathbf{ORAE\text{-}Real}^{\mathscr{A}}_{\Pi}$ and $\mathbf{ORAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}$ are defined via the natural generalization of the offline variants in Figure 2: the attacker is given access either to the scheme or to an ideal online AEAD sampled from $\mathrm{OAE}[\Pi]$ under a random key, and is allowed to construct arbitrary sequences of segments. (The state of the encryption and decryption algorithms are *not* under the view or control of the adversary.) This definition is similar to the OAE2b notion in [HRRV15, Figure 5] and we give the details in Figure 26 in Appendix E.2. A strengthening of this definition that allows adversarial corruption of state values could be considered. This would be implied by indifferentiability (as it is single-stage), but we leave a formal treatment for future work.

CONSTRUCTION VIA COMPOSITION. Let $\mathcal{F}, \mathcal{F}_1, \ldots, \mathcal{F}_n$ be ideal objects and C be a construction of type $\mathcal{F}$ from $\mathcal{F}_1, \ldots, \mathcal{F}_n$. Suppose that the outputs of $\mathcal{F}$ and $\mathrm{C}^{\mathcal{F}_1, \ldots, \mathcal{F}_n}$ are identically distributed

and that one can compute $\mathcal{F}_i$ using $\mathcal{F}$ for all $i$. More precisely, for some (oracle-free) algorithms $E_i$ and $D_i$ we have that $\mathcal{F}_i(X) = D_i(\mathcal{F}(E_i(X))$ for all $i$ and $X$. Then the distributions

$$(\mathrm{C}^{\mathcal{F}_1,\ldots,\mathcal{F}_n}, \mathcal{F}_1, \ldots, \mathcal{F}_n) \equiv (\mathcal{F}, D_1(\mathcal{F}(E_1(\cdot)), \ldots, D_n(\mathcal{F}(E_n(\cdot)))$$

are also identical. We can define sub-simulators $\mathcal{S}_i^{\mathcal{F}} := D_i(\mathcal{F}(E_i(\cdot))$ and obtain a perfect simulator by running them in parallel. Hence $\mathrm{C}^{\mathcal{F}_1,\ldots,\mathcal{F}_n}$ is indifferentiable from $\mathcal{F}$. (The composed simulator places the same number of queries to its oracle as the differentiator does.) By Theorem 1 we also obtain as a corollary that if $\mathrm{C}_i^{f_i}$ are indifferentiable from $\mathcal{F}_i$ for all $i$, then the composed construction $\mathrm{C}^{\mathrm{C}_1^{f_1},\ldots,\mathrm{C}_n^{f_n}}$ is also indifferentiable from $\mathcal{F}$. This follows from a hybrid argument over $\mathrm{C}_i^{f_i}$.

A closer look at our definition of ideal online AEADs shows that it can be decomposed in this way. Consider the construction shown in Figure 18 based on two independent ideal (offline) AEAD schemes $(\mathcal{AE}_1, \mathcal{AD}_1)$, $(\mathcal{AE}_2, \mathcal{AD}_2)$ and two independent random oracles $\mathcal{H}_0$ and $\mathcal{H}_1$. We choose these objects so that various spaces match what is expected from an ideal AEAD, i.e., for $i = 1, 2$

$$\mathcal{A}_i = \mathcal{A} \ , \quad \mathcal{M}_i = \mathcal{M} \ , \quad \mathcal{R}_i = \mathcal{R} \ , \quad \mathcal{N}_i = \{\varepsilon\} \ , \quad \mathcal{K}_i = \mathcal{S} \ .$$

Furthermore, we choose $\mathcal{H}_0 : \mathcal{K} \times \mathcal{N} \longrightarrow \mathcal{S}$ and $\mathcal{H}_1 : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{S}$. It is easy to see that this construction falls within parallel composition discussed above. On the one hand, the distribution of the outputs of the construction is identical to those presented by the ideal object. On the other hand, given access to an ideal AEAD, one can encode inputs to it in way that we obtain the underlying ideal objects $(\mathcal{AE}_1, \mathcal{AD}_1)$, $(\mathcal{AE}_2, \mathcal{AD}_2)$, $\mathcal{H}_0$ and $\mathcal{H}_1$. Indeed, $\mathcal{H}_0$ exactly matches $\mathcal{F}_1$ part of the functionality, $(\mathcal{AE}_2, \mathcal{AD}_2)$ matches $\mathcal{F}_2$, and one can use $\mathcal{F}_1$ to compute both $(\mathcal{AE}_1, \mathcal{AD}_1)$ and $\mathcal{H}_1$ (the latter using the forward direction of $F_1$).

| | |
|---|---|
| ALGO. $\mathcal{AE}.\mathrm{init}^{\mathcal{H}_0}(K, N)$<br>$S \leftarrow \mathcal{H}_0(K, N)$<br>return $S$ | ALGO. $\mathcal{AD}.\mathrm{init}^{\mathcal{H}_0}(K, N)$<br>$S \leftarrow \mathcal{H}_0(K, N)$<br>return $S$ |
| ALGO. $\mathcal{AE}.\mathrm{next}^{\mathcal{AE}_1,\mathcal{H}_1}(S, A, M, \tau)$<br>$C \leftarrow \mathcal{AE}_1(S, \varepsilon, A, M, \tau)$<br>$S \leftarrow \mathcal{H}_1(S, A, M, \tau)$<br>return $(C, S)$ | ALGO. $\mathcal{AD}.\mathrm{next}^{\mathcal{AD}_1,\mathcal{H}_1}(S, A, C, \tau)$<br>$M \leftarrow \mathcal{AD}_1(S, \varepsilon, A, C, \tau)$<br>if $M = \perp$ return $\perp$<br>$S \leftarrow \mathcal{H}_1(S, A, M, \tau)$<br>return $(M, S)$ |
| ALGO. $\mathcal{AE}.\mathrm{last}^{\mathcal{AE}_2}(S, A, M, \tau)$<br>$C \leftarrow \mathcal{AE}_2(S, \varepsilon, A, M, \tau)$<br>return $C$ | ALGO. $\mathcal{AD}.\mathrm{last}^{\mathcal{AD}_2}(S, A, C, \tau)$<br>$M \leftarrow \mathcal{AD}_2(S, \varepsilon, A, C, \tau)$<br>return $M$ |

**Fig. 18.** Ideal online AEAD decomposed into two random oracles $\mathcal{H}_0$ and $\mathcal{H}_1$ and two ideal offline AEAD schemes $(\mathcal{AE}_1, \mathcal{AD}_1)$ and $(\mathcal{AE}_2, \mathcal{AD}_2)$.

It therefore follows that one can instantiate the construction in Figure 18 with indifferentiable constructions of its underlying components and obtain an indifferentiable online AEAD. Interestingly, the resulting construction is similar to **CHAIN** [HRRV15], which although more efficient, is differentiable as we show below. We will also show how to obtain a more efficient indifferentiable construction than that in Figure 18.

REMARK. We can further explore the decomposition paradigm to derive other constructions of complex ideal objects. In Appendix E.3 we give a natural construction of *parallel* AEAD schemes. This further highlights the merits of indifferentiability in designing complex objects in a modular way from simpler ones.

### 7.2 The HashCHAIN construction

Our starting point to improve the efficiency of the above scheme is the **CHAIN** construction of [HRRV15], which is proven OAE2 secure for any arbitrary but fixed (and possibly small) value of $\tau$. This construction processes messages and associated data one segment at a time using an offline AEAD that is tweaked via a truncated XOR of the inputs and outputs in the previous segment; see [HRRV15, Figure 8].

**CHAIN** is trivially differentiable as its initialization procedure $\mathcal{AE}$.init (as well as its implicit state-update procedures) are not random. This raises the possibility that a modified version that derives state values via hashing could be indifferentiable. As we saw above, this would indeed be the case if *all* inputs to $\mathcal{AE}$.init and $\mathcal{AE}$.next are included in hash computations. **CHAIN**, however, aims to avoid this extra hash computation via the simpler truncated XOR operations. To preserve efficiency, one possibility would be to hash the ciphertext instead of the authenticated data to derive the next state (or even only the ciphertext as it implicitly "summarizes" both the message and the authenticated data). This construction would therefore set $S_0 := \mathcal{H}_0(K, N)$ in initialization, and update it within $\mathcal{AE}$.next via $S_i := \mathcal{H}_1(S_{i-1}, M_i, C_i)$. This modification, however, once again results in a differentiable construction.[26]

Intuitively, to achieve indifferentiability the computation of ciphertext/state pair must be done in a way that forces the differentiator to reveal all necessary information that is needed to recompute them via the ideal objects to the simulator. Following this, we propose a new construction in Figure 19, which we call **HashCHAIN**. Here, $(\mathcal{AE}, \mathcal{AD})$ is an offline ideal AEAD with key length $k$, and $\mathcal{H}$ is a VIL/VOL keyed random oracle with key size $k$ that admits outputs of lengths $k$ and $2k$. The nonce and authenticated data spaces of the online scheme are arbitrary. Its message, expansion and ciphertext spaces match those of the offline scheme. The state space is $\mathcal{S} := \mathcal{K}$. To ensure independence of the outputs of $\mathcal{AE}$.init, $\mathcal{AE}$.next, and $\mathcal{AE}$.last (and similarly for decryption) we attach two bits to associated data for domain separation. To avoid hashing the associated data twice, we key encryption and state update via Proposition 3.[27]

**Theorem 7 (HashCHAIN is indifferentiable).** *The* **HashCHAIN** *construction in Figure 19 is indifferentiable from an ideal online AEAD. More precisely, there is a universal simulator $\mathcal{S}$ such that for any $q$/3-query differentiator $\mathscr{D}$,*

$$\mathbf{Adv}^{\text{indiff}}_{\mathbf{HashCHAIN},\mathcal{S}}(\mathscr{D}) \leq 4q^2/2^k \ ,$$

*where $\mathcal{S}$ makes at most $q$ queries to its oracles.*

[26] Consider a differentiator that for $i = 1, 2$ uses the construction oracle to obtain encryption/state pairs $(C_i, S_i)$ for arbitrary $(S_0, A_i, M_i, \tau)$ with random authenticated data $A_i$. It then calls the primitive oracle for $\mathcal{H}_1$ on $(S_1, M_2, C_2)$ and checks if the response matches $S_2$. By construction, this is always be the case in the real world. In the ideal world, however, if the simulator does not call the ideal $\mathcal{AE}$.next on $(S_1, A_1, M_1, C_1)$ it can only guess the random state value $S_2$. On the other hand, to place this query it needs to guess the random $A_1$, which can be done only with a small probability after a reasonable number of queries.

[27] Using a single key that is tweaked based on $(S, A)$ would be sufficient. However, for our scheme to remain amenable to standard-model analyses we compute two separate keys.

| Algo. $\mathcal{AE}.\text{init}^{\mathcal{H}}(K, N)$ | Algo. $\mathcal{AD}.\text{init}^{\mathcal{H}}(K, N)$ |
|---|---|
| $S \leftarrow \mathcal{H}(K, (00, N), k)$ | $S \leftarrow \mathcal{H}(K, (00, N), k)$ |
| return $S$ | return $S$ |
| Algo. $\mathcal{AE}.\text{next}^{\mathcal{AE},\mathcal{H}}(S, A, M, \tau)$ | Algo. $\mathcal{AD}.\text{next}^{\mathcal{AD},\mathcal{H}}(S, A, C, \tau)$ |
| $(K_1, K_2) \leftarrow \mathcal{H}(S, (01, A, \tau), 2k)$ | $(K_1, K_2) \leftarrow \mathcal{H}(S, (01, A, \tau), 2k)$ |
| $C \leftarrow \mathcal{AE}(K_1, \varepsilon, \varepsilon, M, \tau)$ | $M \leftarrow \mathcal{AD}(K_1, \varepsilon, \varepsilon, C, \tau)$ |
| $S \leftarrow \mathcal{H}(K_2, (10, M), k)$ | if $M = \perp$ return $\perp$ |
| return $(C, S)$ | $S \leftarrow \mathcal{H}(K_2, (10, M), k)$ |
| | return $(M, S)$ |
| Algo. $\mathcal{AE}.\text{last}^{\mathcal{AE},\mathcal{H}}(S, A, M, \tau)$ | Algo. $\mathcal{AD}.\text{last}^{\mathcal{AD},\mathcal{H}}(S, A, C, \tau)$ |
| $K_1 \leftarrow \mathcal{H}(S, (11, A, \tau), k)$ | $K_1 \leftarrow \mathcal{H}(S, (11, A, \tau), k)$ |
| $C \leftarrow \mathcal{AE}(K_1, \varepsilon, \varepsilon, M, \tau)$ | $M \leftarrow \mathcal{AD}(K_1, \varepsilon, \varepsilon, C, \tau)$ |
| return $C$ | return $M$ |

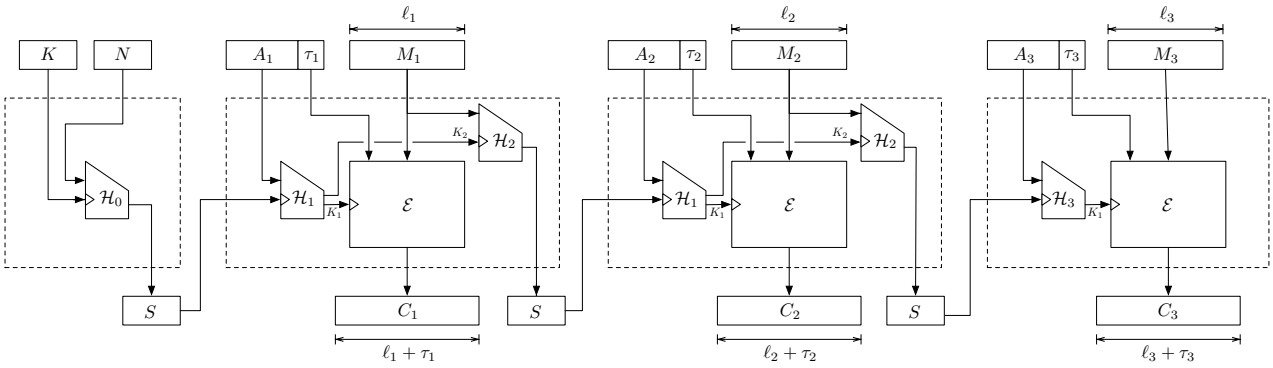


**Fig. 19.** The **HashCHAIN** transform (top) and its schematic diagram (bottom).

*Proof (Sketch).* We show via a sequence of replacements of various objects that the view of any differentiator in the real game with respect to this construction is close to that with respect to the construction in Figure 18. Wherever the construction uses domain separation on the inputs of $\mathcal{H}$, we introduce independent random oracles instead. This means that $\mathcal{AE}$.init and $\mathcal{AD}$.init use a random oracle $\mathcal{H}_{00}(K, N, k)$ independent of the rest of the oracles used. Algorithms $\mathcal{AE}$.next and $\mathcal{AD}$.next use a random oracle $\mathcal{H}_{01}$ and $\mathcal{AE}$.last and $\mathcal{AD}$.last use a random oracle $\mathcal{H}_{11}$. We then replace the computation of $(K_1, K_2)$ by the parallel application of two independent random oracles $(\mathcal{H}_{01,1}(S, (A, \tau), k), \mathcal{H}_{01,2}(S, (A, \tau), k)$. Both of the above transformations are lossless as a consequence of parallel composition: indeed, both domain separation and range separation are particular cases of parallel composition. By Proposition 3 we can then replace the lines $K_1 \leftarrow \mathcal{H}_{01,1}(S, (A, \tau), k)$ and $C \leftarrow \mathcal{AE}(K_1, \varepsilon, \varepsilon, M, \tau)$ in the operation of $\mathcal{AE}$.next with $\mathcal{AE}_1(S, \varepsilon, A, M, \tau)$. Similar we use $\mathcal{AD}_1(S, \varepsilon, A, M, \tau)$ in $\mathcal{AD}$.next. This incurs a loss of $2q_1^2/2^k$ in security for $q_1/3$ the total number of queries to $\mathcal{AE}$.next and $\mathcal{AD}$.next. Similarly, we use $\mathcal{AE}_2(S, \varepsilon, A, M, \tau)$ in $\mathcal{AE}$.last and $\mathcal{AD}_2(S, \varepsilon, A, M, \tau)$ in $\mathcal{AD}$.last. Again this incurs a loss of $2q_2^2/2^k$ in security for $q_2/3$ the total number of queries to $\mathcal{AE}$.last and $\mathcal{AD}$.last. The resulting scheme is now identical to that in Figure 18 and therefore indifferentiable from an online OAED. The bound in the theorem follows from the fact that $q_1/3 + q_2/3 \leq q/3$. □

FURTHER OPTIMIZATIONS. **HashCHAIN**, in addition to computing the offline AEAD for each segment, also computes a $k$-bit hash value $\mathcal{H}(K_2, (01, M), k)$, where $k$ is the key length. If $M$ is $wn$ bits long and $\mathcal{H}$ has rate $n$, this incurs an extra $w$ calls to the ideal primitive underlying $\mathcal{H}$. We can eliminate this overhead for our 3-round Feistel construction as follows. Recall that the first round of Feistel computes a $\tau$-bit hash of the $wn$-bit input message $M$. We can extend this hash value to $\tau + k$ bits via a *single* extra call to the primitive underlying $\mathcal{H}$. We then replace $\mathcal{H}(K_2, (01, M), k)$ with the last $k$ bits of this hash value. This reduces the $w$ extra ideal-primitive calls to a single call. A proof similar to that for **HashCHAIN** can be given for this construction as we can first decompose the random oracle implicit in the first round of Feistel and then proceed as in the proof of **HashCHAIN**.

## Acknowledgments

The authors would like to thank Phillip Rogaway, Martijn Stam, and Stefano Tessaro for their comments. Barbosa was supported in part by Project NORTE-01-0145-FEDER-000020, financed by the North Portugal Regional Operational Programme (NORTE 2020) under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF). Farshim was supported in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 - CryptoCloud). This work was initiated during a short-term scientific mission sponsored by the COST CryptoAction (IC1306).

## References

ABD+13. Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Heidelberg, August 2013.

ABL+14. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125. Springer, Heidelberg, December 2014.

ABM+13. Martín Abadi, Dan Boneh, Ilya Mironov, Ananth Raghunathan, and Gil Segev. Message-locked encryption for lock-dependent messages. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 374–391. Springer, Heidelberg, August 2013.

ADL17. Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2017.

AFL+16. Farzaneh Abed, Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. RIV for robust authenticated encryption. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 23–42. Springer, Heidelberg, March 2016.

AFPW11. Martin R. Albrecht, Pooya Farshim, Kenneth G. Paterson, and Gaven J. Watson. On cipher-dependent related-key attacks in the ideal-cipher model. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 128–145. Springer, Heidelberg, February 2011.

BBT16. Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Heidelberg, May 2016.

BCS05. John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 526–541. Springer, Heidelberg, May 2005.

BDPV08. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008.

Ber14. Daniel J. Bernstein. Cryptographic competitions, 2014. https://competitions.cr.yp.to.

BK03. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

BK11. Mihir Bellare and Sriram Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 610–629. Springer, Heidelberg, August 2011.

BKR13. Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 296–312. Springer, Heidelberg, May 2013.

BMM+15. Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. Robust authenticated encryption and the limits of symmetric cryptography. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 112–129. Springer, Heidelberg, December 2015.

BMOS17. Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 693–723. Springer, Heidelberg, December 2017.

BN00. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000.

BR00. Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, Heidelberg, December 2000.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

BRS03. John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

BRW04. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer, Heidelberg, February 2004.

BT16. Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, Heidelberg, August 2016.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

CDMS10. Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A domain extender for the ideal cipher. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 273–289. Springer, Heidelberg, February 2010.

CHK+16. Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

CO15. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 40–58. Springer, Heidelberg, August 2015.

CPS08. Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2008.

DGHM13. Gregory Demay, Peter Gaži, Martin Hirt, and Ueli Maurer. Resource-restricted indifferentiability. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 664–683. Springer, Heidelberg, May 2013.

DKS+17. Yevgeniy Dodis, Jonathan Katz, John Steinberger, Aishwarya Thiruvengadam, and Zhe Zhang. Provable security of substitution-permutation networks. Cryptology ePrint Archive, Report 2017/016, 2017. http://eprint.iacr.org/2017/016.

DKT16. Dana Dachman-Soled, Jonathan Katz, and Aishwarya Thiruvengadam. 10-round Feistel is indifferentiable from an ideal cipher. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 649–678. Springer, Heidelberg, May 2016.

DRRS09. Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *LNCS*, pages 104–121. Springer, Heidelberg, February 2009.

DRS09. Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Heidelberg, April 2009.

DRST12. Yevgeniy Dodis, Thomas Ristenpart, John P. Steinberger, and Stefano Tessaro. To hash or not to hash again? (In)differentiability results for $H^2$ and HMAC. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 348–366. Springer, Heidelberg, August 2012.

DS15. Yuanxi Dai and John Steinberger. Indifferentiability of 10-round Feistel networks. Cryptology ePrint Archive, Report 2015/874, 2015. http://eprint.iacr.org/2015/874.

DS16. Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round Feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Heidelberg, August 2016.

DSSL16. Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 679–704. Springer, Heidelberg, May 2016.

DSST17. Yuanxi Dai, Yannick Seurin, John P. Steinberger, and Aishwarya Thiruvengadam. Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 524–555. Springer, Heidelberg, August 2017.

FLLW17. Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Reforgeability of authenticated encryption schemes. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 17, Part II*, volume 10343 of *LNCS*, pages 19–37. Springer, Heidelberg, July 2017.

FOR17. Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symm. Cryptol.*, 2017(1):449–473, 2017.

FP15. Pooya Farshim and Gordon Procter. The related-key security of iterated Even-Mansour ciphers. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 342–363. Springer, Heidelberg, March 2015.

GL15. Shay Gueron and Yehuda Lindell. GCM-SIV: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 109–119. ACM Press, October 2015.

GLR17. Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 66–97. Springer, Heidelberg, August 2017.

GT00. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, November 2000.

HK07. Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 466–475. ACM Press, October 2007.

HKR15. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, April 2015.

HKR17. Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. AEZ v5: Authenticated encryption by enciphering, 2017. https://competitions.cr.yp.to/round3/aezv5.pdf.

HKT11. Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.

HRRV15. Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 493–517. Springer, Heidelberg, August 2015.

IK04. Tetsu Iwata and Tadayoshi Kohno. New security proofs for the 3GPP confidentiality and integrity algorithms. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 427–445. Springer, Heidelberg, February 2004.

JNPS16. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41, 2016. https://competitions.cr.yp.to/round3/deoxysv141.pdf.

KPS13. Eike Kiltz, Krzysztof Pietrzak, and Mario Szegedy. Digital signatures with minimal overhead from indifferentiable random invertible functions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 571–588. Springer, Heidelberg, August 2013.

KT09. Ralf Küsters and Max Tuengerthal. Universally composable symmetric encryption. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*, pages 293–307. IEEE Computer Society, 2009.

Luc96. Stefan Lucks. Faster Luby-Rackoff ciphers. In Dieter Gollmann, editor, *FSE'96*, volume 1039 of *LNCS*, pages 189–203. Springer, Heidelberg, February 1996.

MRH04. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

MW04. Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 133–151. Springer, Heidelberg, February 2004.

NRS13. Chanathip Namprempre, Phillip Rogaway, and Tom Shrimpton. AE5 security notions: Definitions implicit in the CAESAR call. Cryptology ePrint Archive, Report 2013/242, 2013. http://eprint.iacr.org/2013/242.

NRS14. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.

PS16. Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 33–63. Springer, Heidelberg, August 2016.

PSV15. Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15*, pages 96–108. ACM Press, October 2015.

RBBK01. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01*, pages 196–205. ACM Press, November 2001.

Rog02. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 02*, pages 98–107. ACM Press, November 2002.

RS06. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

RVV16. Reza Reyhanitabar, Serge Vaudenay, and Damian Vizár. Authenticated encryption with variable stretch. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 396–425. Springer, Heidelberg, December 2016.

Sta08. Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 397–412. Springer, Heidelberg, August 2008.

Unr12. Dominique Unruh. Programmable encryption and key-dependent messages. Cryptology ePrint Archive, Report 2012/423, 2012. http://eprint.iacr.org/2012/423.

Vau02. Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 534–546. Springer, Heidelberg, April / May 2002.

## A Generically Composed Schemes

We recall the A, B, and N-schemes from [NRS14] below.

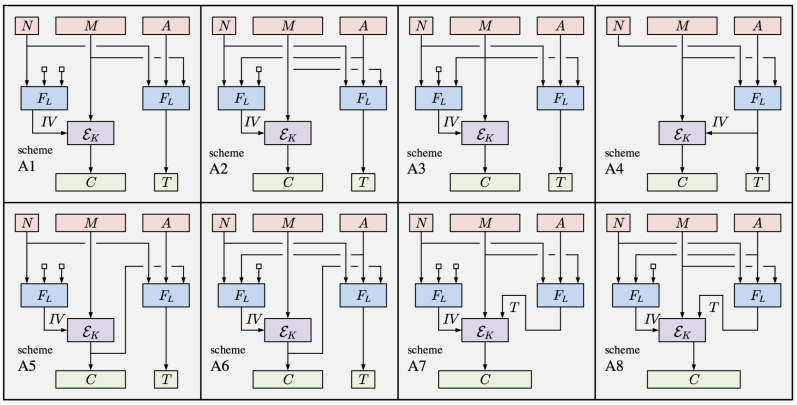


**Fig. 20.** A-schemes from [NRS14, Figure 2] (with permission).

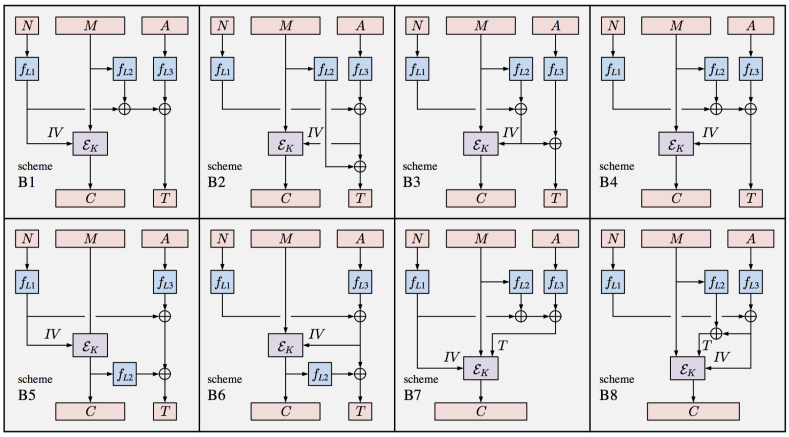


**Fig. 21.** B-schemes from [NRS14, Figure 3] (with permission).

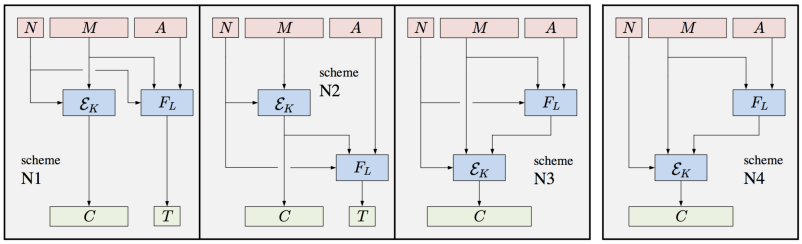


**Fig. 22.** N-schemes from [NRS14, Figure 6] (with permission).

## B Schematic Diagram for AEZ

We recall the operation of AEZ v5 [HKR17] below.

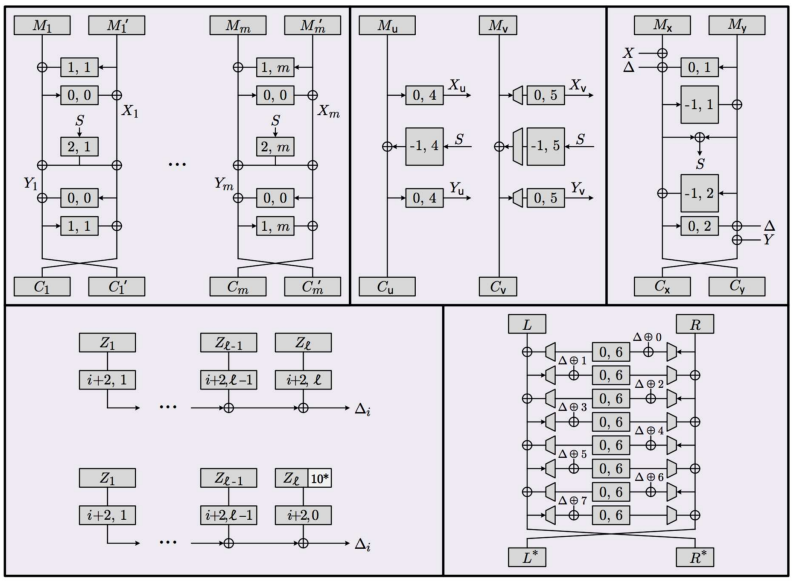

**Fig. 23.** AEZ-core (top), AEZ-hash (bottom left), and AEZ-tiny (bottom right) from [HKR17, Figure 5] (with permission).

## C RIV Is Differentiable

The Robust Initialization Vector [AFL+16] (RIV) mode of encryption is shown in Figure 24. For indifferentiability we view $\mathcal{H}_1$ and $\mathcal{H}_2$ as independent random oracles, and $(\mathcal{E}, \mathcal{D})$ as an ideal encryption scheme without associated data. RIV is a type-I scheme (as defined in Figure 4) and hence by Theorem 3 it is differentiable. To see this note that $\mathcal{I}_+$ can omit $K_2$ from $est_1$ as it is not used in post processing (but note that it must still include $M$ to recompute the $IV$). Furthermore, $\Delta_C = |T|$, which is large. Finally, the recovery algorithm $\mathcal{R}_1$ removes the tag $T$.

| Algo. $\mathcal{AE}((K_1, K_2), N, A, M)$ | Algo. $\mathcal{AD}((K_1, K_2), N, A, (C', T))$ |
|---|---|
| $IV \leftarrow \mathcal{H}_1(K_1, N, A, M)$ | $IV \leftarrow T \oplus \mathcal{H}_2(K_1, N, A, C')$ |
| $C' \leftarrow \mathcal{E}(K_2, IV, \varepsilon, M)$ | $M \leftarrow \mathcal{D}(K_2, IV, \varepsilon, C')$ |
| $T \leftarrow \mathcal{H}_2(K_1, N, A, C') \oplus IV$ | if $T \neq \mathcal{H}_1(K_1, N, A, M)$ return $\perp$ |
| return $(C', T)$ | return $M$ |

**Fig. 24.** The Robust Initialization Vector mode.

## D Proof of Lemma 2: Hybrid over Keys

**Lemma 3 (Lemma 2, restated).** *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two uniformly keyed objects and $\mathrm{C}^{\mathcal{F}_1}$ be a key-respecting construction of $\mathcal{F}_2$ from $\mathcal{F}_1$. Then if $\mathrm{C}^{\mathcal{F}_1}$ is key-wise indifferentiable, it is also (fully) indifferentiable. More precisely, for any key-respecting simulator $\mathcal{S}$ and any $q$-query (unrestricted) differentiator $\mathscr{D}$ there is a key-respecting differentiator $\mathscr{D}'$ such that*

$$\mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D}) \leq q \cdot \mathbf{Adv}^{\mathrm{indiff}}_{\mathrm{C},\mathcal{S}}(\mathscr{D}') \ .$$

*Proof.* For $i \in \{0, \ldots, q\}$ consider a hybrid world $\mathbf{G}_i^{\mathcal{F}_1, \mathcal{F}_2}$ that operates as follows. On the $\ell$-th queried key $K_\ell$ (to either Const or Prim), if $\ell \leq i$, the real procedures $\mathrm{C}_1^{\mathcal{F}}$ or $\mathcal{F}_1$ are used to

answer the query. If $\ell > i$ and there was a key $K_j$ with query number $j \leq i$ such that $K_j = K_\ell$, again $\mathrm{C}_1^{\mathcal{F}}$ or $\mathcal{F}_1$ are used to answer the query. If $\ell > i$ and no key with query number $j \leq i$ matches $K_\ell$, then $\mathcal{F}_2$ and $\mathcal{S}^{\mathcal{F}_2}$ are used. Observe that when $i = 0$ the hybrid game matches the ideal world, and when $i = q$ the hybrid matches the real world.

We show that no differentiator $\mathscr{D}$ can distinguish hybrid $i-1$ from hybrid $i$ for any $i \in \{1, \ldots, q\}$. Consider a differentiator $\mathscr{D}'$ that runs $\mathscr{D}$ and answers its $\ell$-th queried key $K_\ell$ (to either Const or Prim) as follows.

1. If $\ell < i$ the procedures $\mathrm{C}_1^{\mathcal{F}}$ or $\mathcal{F}_1$ with a *lazily sampled* $\mathcal{F}_1$ are used to answer the query.
2. If $\ell = i$ the provided (challenge) oracles are used to answer the query.
3. If $\ell > i$ and $K_\ell = K_j$ for a previous key $K_j$ with query number $j < i$, again $\mathrm{C}_1^{\mathcal{F}}$ or $\mathcal{F}_1$ with a lazily sampled $\mathcal{F}_1$ are used.
4. If $\ell > i$ and $K_\ell = K_i$ the provided (challenge) oracles are used to answer the query.
5. If $\ell > i$ and $K_\ell \neq K_j$ for all keys $K_j$ with query number $j \leq i$, then $\mathcal{F}_2$ and $\mathcal{S}^{\mathcal{F}_2}$ with a lazily sampled $\mathcal{F}_2$ are used.

We observe that $\mathscr{D}'$ is indeed a key-respecting differentiator as it only queries $K_i$ to its provided oracles. Since both the construction and the simulator are key-respecting, the construction queries $\mathcal{F}_1$ only on keys that are input to it, and similarly the simulator queries $\mathcal{F}_2$ only on the keys that are input to it. Since $\mathcal{F}_1$ and $\mathcal{F}_2$ are uniformly keyed, their instances over distinct keys are independent. This means the lazy sampling of oracles above are distributed identically to those that $\mathscr{D}$ expects. Furthermore, $\mathscr{D}'$ runs $\mathscr{D}$ in a environment identical to the $(i-1)$-st or the $i$-th hybrid:

1. For queries $\ell < i$ or $\ell > i$ with $K_\ell = K_j$ for $j < i$ the outputs are distributed identically to both hybrid $i-1$ and hybrid $i$.
2. All queries $\ell > i$ with $K_\ell \neq K_j$ for all $j \leq i$ are distributed identically to hybrids $i-1$ and $i$.
3. If $\ell = i$ or $\ell > i$ and $K_\ell = K_i$ and the provided (challenge) oracles implement the real procedures, the output distribution is identical to hybrid $i$. When the procedures are implement via the ideal $\mathcal{F}_2$ and $\mathcal{S}^{\mathcal{F}_2}$, the output distribution is identical to hybrid $(i-1)$.

The lemma follows. □

# E Missing Details for Online AEADs

## E.1 Associated offline scheme

The correctness of an online AEAD scheme $\Pi$ is defined via an *induced* offline AEAD scheme with encryption and decryption algorithms $\boldsymbol{\mathcal{AE}}$, $\boldsymbol{\mathcal{AD}} : \mathcal{K} \times \mathcal{N} \times \{0,1\}^{**} \times \mathcal{X}^* \longrightarrow \{0,1\}^{**}$ that operate iteratively on segmented strings. Algorithm $\boldsymbol{\mathcal{AD}}(K, N, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\tau})$ returns the longest $\boldsymbol{M}$ whose encryption (using $K$, $N$, $\boldsymbol{A}$, and $\boldsymbol{\tau}$) is a prefix of $\boldsymbol{C}$. Figure 25 details the operation of these algorithms. We require the following correctness condition for an online AEAD scheme: if $K \in \mathcal{K}$, $N \in \mathcal{N}$, $\boldsymbol{A} \in \{0,1\}^{**}$, $\boldsymbol{M} \in \{0,1\}^{**}$, $\boldsymbol{\tau} \in \mathcal{X}^*$, and $\boldsymbol{C} = \boldsymbol{\mathcal{AE}}(K, N, \boldsymbol{A}, \boldsymbol{M}, \boldsymbol{\tau})$, then $\boldsymbol{M} = \boldsymbol{\mathcal{AD}}(K, N, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{\tau})$. The segmented-string with zero components is the empty vector $[\,]$, which is different from the empty string $\varepsilon$.

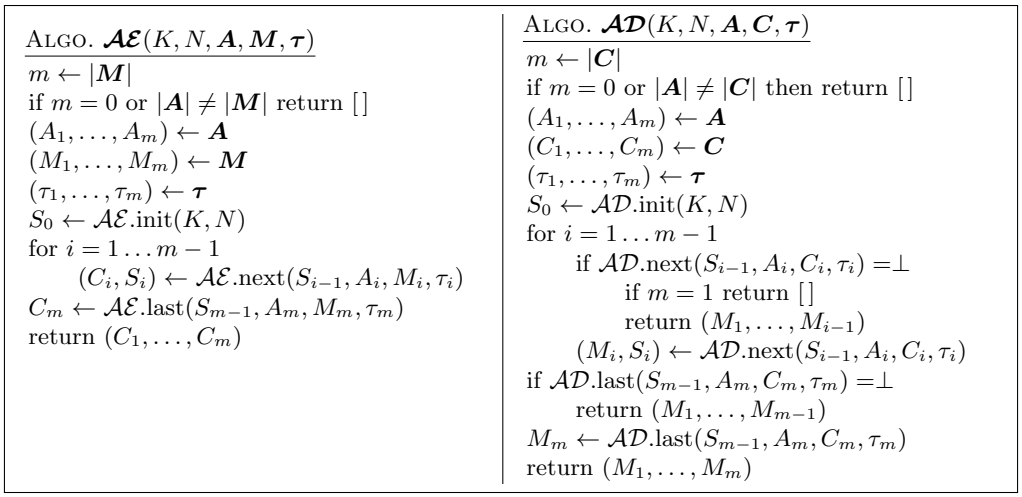


**Fig. 25.** The offline AEAD scheme associated to an online AEAD scheme.

### E.2 ORAE security

We define the advantage of an adversary $\mathscr{A}$ against scheme $\Pi$ in the online RAE game as

$$\mathbf{Adv}^{\text{orae}}_{\Pi}(\mathscr{A}) := \Pr\left[\mathbf{ORAE\text{-}Real}^{\mathscr{A}}_{\Pi}\right] - \Pr\left[\mathbf{ORAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}\right] ,$$

where games $\mathbf{ORAE\text{-}Real}^{\mathscr{A}}_{\Pi}$ and $\mathbf{ORAE\text{-}Ideal}^{\mathscr{A}}_{\Pi}$ are shown in Figure 26.

### E.3 Parallel AEAD

Some application scenarios, such as video streaming or interactive terminals, demand that after some initialization stage an AEAD scheme processes incoming segments independently of *all* other segments. Following [HRRV15], we can formalize a *parallel* online AEAD scheme as a restricted type of an online AEAD scheme where *state remains unmodified* in each iteration.

SYNTAX. A parallel online AEAD scheme is a seven tuple of algorithms

$$\Pi = (\mathcal{K}, \mathcal{AE}.\text{init}, \mathcal{AE}.\text{next}, \mathcal{AE}.\text{last}, \mathcal{AD}.\text{init}, \mathcal{AD}.\text{next}, \mathcal{AD}.\text{last}) .$$

Here $\mathcal{K}$ is the key-generation algorithm as in online schemes and the remaining algorithms have signatures as shown below.

$$\begin{array}{ll}
\mathcal{AE}.\text{init} : \mathcal{K} \times \mathcal{N} \longrightarrow \mathcal{S} & \mathcal{AD}.\text{init} : \mathcal{K} \times \mathcal{N} \longrightarrow \mathcal{S} \\
\mathcal{AE}.\text{next} : \mathbb{N} \times \mathcal{S} \times \mathcal{H} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C} & \mathcal{AD}.\text{next} : \mathbb{N} \times \mathcal{S} \times \mathcal{H} \times \mathcal{C} \times \mathcal{X} \longrightarrow \mathcal{M} \cup \{\perp\} \\
\mathcal{AE}.\text{last} : \mathbb{N} \times \mathcal{S} \times \mathcal{H} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{C} & \mathcal{AD}.\text{last} : \mathbb{N} \times \mathcal{S} \times \mathcal{H} \times \mathcal{C} \times \mathcal{X} \longrightarrow \mathcal{M} \cup \{\perp\}
\end{array}$$

All spaces are interpreted as for online AEAD schemes and all algorithms except state initialization take as additional input the segment number. Note that $\mathcal{AE}.\text{next}$ and $\mathcal{AD}.\text{next}$ do not return updated states. The correctness of a parallel online AEAD scheme is again defined via an *associated* offline AEAD scheme in the natural way. We omit the details for conciseness.

| GAME **ORAE-Real**$^{\mathscr{A}}_{\Pi}$ | GAME **ORAE-Ideal**$^{\mathscr{A}}_{\Pi}$ |
|---|---|
| $I, J \leftarrow 0$; $K \twoheadleftarrow \mathcal{K}$<br>$b \twoheadleftarrow \mathscr{A}^{\text{Enc.init},\text{Enc.next},\text{Enc.last},\text{Dec.init},\text{Dec.next},\text{Dec.last}}$<br>return $b$ | $(\mathcal{AE}'.\text{init}, \ldots, \mathcal{AD}'.\text{last}) \twoheadleftarrow \text{OAE}[\Pi]$<br>$I, J \leftarrow 0$; $K \twoheadleftarrow \mathcal{K}$<br>$b \twoheadleftarrow \mathscr{A}^{\text{Enc.init},\text{Enc.next},\text{Enc.last},\text{Dec.init},\text{Dec.next},\text{Dec.last}}$<br>return $b$ |
| PROC. ENC.init($N$)<br>$I \leftarrow I + 1$; $S_I \leftarrow \mathcal{AE}.\text{init}(K, N)$<br>return $I$ | PROC. ENC.init($N$)<br>$I \leftarrow I + 1$; $S_I \leftarrow \mathcal{AE}'.\text{init}(K, N)$<br>return $I$ |
| PROC. ENC.next($i, A, M, \tau$)<br>if $i \notin [1..I]$ return $\perp$<br>$(C, S_i) \leftarrow \mathcal{AE}.\text{next}(S_i, A, M, \tau)$<br>return $C$ | PROC. ENC.next($i, A, M, \tau$)<br>if $i \notin [1..I]$ return $\perp$<br>$(C, S_i) \leftarrow \mathcal{AE}'.\text{next}(S_i, A, M, \tau)$<br>return $C$ |
| PROC. ENC.last($i, A, M, \tau$)<br>if $i \notin [1..I]$ return $\perp$<br>$C \leftarrow \mathcal{AE}.\text{last}(S_i, A, M, \tau)$<br>$S_i \leftarrow \perp$; return $C$ | PROC. ENC.last($i, A, M, \tau$)<br>if $i \notin [1..I]$ return $\perp$<br>$C \leftarrow \mathcal{AE}'.\text{last}(S_i, A, M, \tau)$<br>$S_i \leftarrow \perp$; return $C$ |
| PROC. DEC.init($N$)<br>$J \leftarrow J + 1$; $S'_J \leftarrow \mathcal{AD}.\text{init}(K, N)$<br>return $J$ | PROC. DEC.init($N$)<br>$J \leftarrow J + 1$; $S'_J \leftarrow \mathcal{AD}'.\text{init}(K, N)$<br>return $J$ |
| PROC. DEC.next($j, A, C, \tau$)<br>if $j \notin [1..J]$ return $\perp$<br>$(M, S'_j) \leftarrow \mathcal{AD}.\text{next}(S'_j, A, C, \tau)$<br>return $M$ | PROC. DEC.next($j, A, C, \tau$)<br>if $j \notin [1..J]$ return $\perp$<br>$(M, S'_j) \leftarrow \mathcal{AD}'.\text{next}(S'_j, A, C, \tau)$<br>return $M$ |
| PROC. DEC.last($j, A, C, \tau$)<br>if $j \notin [1..J]$ return $\perp$<br>$M \leftarrow \mathcal{AD}.\text{last}(S'_j, A, C, \tau)$<br>$S'_j \leftarrow \perp$; return $M$ | PROC. DEC.last($j, A, C, \tau$)<br>if $j \notin [1..J]$ return $\perp$<br>$M \leftarrow \mathcal{AD}'.\text{last}(S'_j, A, C, \tau)$<br>$S'_j \leftarrow \perp$; return $M$ |

**Fig. 26.** Games defining the ORAE security of an online AEAD scheme.

PARALLEL ONLINE FUNCTIONS. A parallel online function is a triple of functions with signatures

$$\mathcal{F}_0 : \mathcal{A}_0 \longrightarrow \mathcal{S} \ , \qquad \mathcal{F}_1 : \mathbb{N} \times \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}_1 \ , \qquad \mathcal{F}_2 : \mathbb{N} \times \mathcal{S} \times \mathcal{A} \times \mathcal{M} \times \mathcal{X} \longrightarrow \mathcal{R}_2 \ .$$

We define $\text{Inj}_{||}[\mathcal{A}_0, \mathcal{A}, \mathcal{M}, \mathcal{X}, \mathcal{S}, \mathcal{R}_1, \mathcal{R}_2]$ as the set of parallel online functions for which $\mathcal{F}_1$ and $\mathcal{F}_2$ are injective over $\mathcal{M}$ and respect the usual length-expansion requirement. The unique inverse functions are defined analogously to online functions.

IDEAL PARALLEL ONLINE AEAD. A parallel online AEAD is a function sampled uniformly from $\text{Inj}_{||}[\mathcal{A}_0, \mathcal{A}, \mathcal{M}, \mathcal{X}, \mathcal{S}, \mathcal{R}_1, \mathcal{R}_2]$ where $\mathcal{A}_0 := \mathcal{K} \times \mathcal{N}$, $\mathcal{A} := \mathcal{H}$, and $\mathcal{R}_1 := \mathcal{R}_2 := \mathcal{C}$. Given an online AEAD scheme $\Pi$, we define POAE[$\Pi$] as the set of all expanding parallel online injections with signatures that match those of $\Pi$.

Applying the same design principles based on the composition properties of indifferentiability, we obtain an indifferentiable construction of a parallel AEAD similar to the **STREAM** construction [HRRV15, Figure 10 of full version], which we present in Figure 27. Our construction uses a tweaked key $S := \mathcal{H}(K, N)$ as the initial state throughout and the segment number $i$ is used as

a nonce.[28] The original **STREAM** construction includes $(K, N)$ in $S$ and uses $(N, i)$ as nonce in each invocation. As for online schemes, our approach provides protection against state corruption.

| | |
|---|---|
| ALGO. $\mathcal{AE}.\text{init}^{\mathcal{H}}(K, N)$<br>$S \leftarrow \mathcal{H}(K, N)$<br>return $S$ | ALGO. $\mathcal{AD}.\text{init}^{\mathcal{H}}(K, N)$<br>$S \leftarrow \mathcal{H}(K, N)$<br>return $S$ |
| ALGO. $\mathcal{AE}.\text{next}^{\mathcal{AE}_1}(i, S, A, M, \tau)$<br>$C \leftarrow \mathcal{AE}_1(S, i, A, M, \tau)$<br>return $C$ | ALGO. $\mathcal{AD}.\text{next}^{\mathcal{AD}_1}(i, S, A, C, \tau)$<br>$M \leftarrow \mathcal{AD}_1(S, i, A, C, \tau)$<br>if $M = \perp$ return $\perp$<br>return $M$ |
| ALGO. $\mathcal{AE}.\text{last}^{\mathcal{AE}_2}(i, S, A, M, \tau)$<br>$C \leftarrow \mathcal{AE}_2(S, i, A, M, \tau)$<br>return $C$ | ALGO. $\mathcal{AD}.\text{last}^{\mathcal{AD}_2}(i, S, A, C, \tau)$<br>$M \leftarrow \mathcal{AD}_2(S, i, A, C, \tau)$<br>return $M$ |

**Fig. 27.** A parallel online AEAD scheme from a random oracle $\mathcal{H}$ and ideal offline AEAD schemes $(\mathcal{AE}_1, \mathcal{AD}_2)$ and $(\mathcal{AE}_1, \mathcal{AD}_2)$. Encryption and decryption take as input a sequence number $i$.

[28] Note that the nonce input is not used in **HashCHAIN**, which instead relies on the state update function for tweaking successive segments.