

# Forkcipher-Based Committing Security from Explicit Collision Security Assumptions

Elena Andreeva <sup>1</sup>, Maria Eichlseder <sup>2</sup>, Simon Gerhalter <sup>2</sup>, Marcel Nageler <sup>2</sup>, Oliver Christoph Popa <sup>2</sup>, and Andreas W€ninger <sup>1</sup>

<sup>1</sup> TU Wien, Vienna, Austria

elena.andreeva@tuwien.ac.at, andreas.weninger@house-of-innovation.com

<sup>2</sup> Graz University of Technology, Graz, Austria

maria.eichlseder@tugraz.at, simon.gerhalter@tugraz.at,  
marcel.nageler@tugraz.at, oliver.popa@tugraz.at

**Abstract.** Context-committing security binds an authenticated encryption (AE) ciphertext uniquely to its encryption context. Many conventional AE schemes lack this guarantee, while most generic committing transforms rely on idealized primitives.

We develop committing AE from explicit forkcipher collision-resistance properties: fCR and zero-message collision resistance (zmCR). For an  $n$ -to- $2n$ -bit forkcipher, we establish matching query complexities of order  $2^{n/2}$  and  $2^n$ , respectively. MILP- and SAT-based differential and reduced-round collision analyses of ForkSkinny-128-384 support the corresponding generic security levels for the full-round primitive. We also analyze collisions in ButterKnife and ZIP-AES.

We lift fCR and zmCR through FCPRF and FIXM to fixed-length collision-resistant PRFs. We introduce nonce-based PRF security to capture the nonce-respecting pseudorandomness needed by our constructions, and construct FHASHN, proving its variable-input-length collision resistance, pseudorandomness, and nonce-based pseudorandomness.

We present AEAH-2K, a two-key variant of AEAH combining secure AE with a variable-input-length collision-resistant PRF. Including the AE key in the commitment input gives confidentiality, authenticity, and CMT-4 security in the standard model. We also construct FCTR-CMT, an AEAD mode using a single forkcipher, and prove nonce-respecting privacy, authenticity, and CMT-4 security. With FHASHN based on FCPRF, FCTR-CMT achieves  $n$ -bit confidentiality and  $n/2$ -bit authenticity and CMT-4 security; with FIXM, all 3 reach  $n$  bits. All construction proofs reduce to explicit pseudorandomness and collision-resistance assumptions.

**Keywords:** Committing Security · Forkciphers · PRFs

## 1 Introduction

Authenticated encryption (AE) provides *confidentiality* and *authenticity*, but a ciphertext may be valid under distinct encryption contexts. *Committing security* excludes this ambiguity by binding a ciphertext to its encryption inputs [27,8,16]:

key commitment (CMT-1) binds the encryption key, while context commitment (CMT-4) binds the tuple of key, nonce, associated data, and plaintext. Related notions are robust encryption [1] and receiver-binding encryption [29].

Commitment matters whenever a protocol treats a ciphertext as evidence of an encryption context. Its importance was first demonstrated in public-key applications: missing commitment enables attacks on oblivious-transfer (OT) protocols [34], while other OT proofs explicitly require it [20]. Abdalla et al. [1] likewise establish the need for robustness in public-key encryption with keyword search. In the symmetric-key setting, Farshim et al. [27] identify several applications requiring commitment. Len et al. [35] introduce partitioning-oracle attacks, which test whether an unknown key belongs to a chosen subset and can efficiently recover low-entropy secrets when AE lacks key commitment. Verifiable abuse reporting also requires ciphertexts to provide unambiguous evidence of the sender’s message and context [29,25]. Albertini et al. [2] show attacks from missing key commitment in widely deployed schemes, including AES-GCM, ChaCha20-Poly1305, AES-GCM-SIV, and OCB3. CCM, EAX, and SIV also lack committing security [38]. With this growing importance, NIST lists commitment among the desired properties of its future Accordion Cipher mode [19].

EXISTING CONSTRUCTIVE APPROACHES. Many works transform conventional symmetric AE into CMT-1- or CMT-4-secure schemes [8,16,9,30,46]. Existing designs such as the sponge-based Ascon already offer forms of commitment [45,33], and newer schemes explicitly target it [21].

Bellare and Hoang [8] modify GCM and AES-GCM-SIV to achieve 64-bit CMT-1 security with little computational overhead. They also give a generic CMT-1 transform with ciphertext expansion and a hash-then-encrypt upgrade to CMT-4 requiring additional hashing. The generic CTX transform [16] and its refinement CTY [9] add  $2s$  ciphertext bits for  $s$ -bit committing security, Hash-then-Mask reduces this overhead to  $s$  bits [9].

Chen and Karadžić [17] compare generic CMT-4 transforms and propose the practically efficient AEaH, which combines AE with a hash-based commitment tag analyzed in the random-oracle model. Most committing AE constructions rely on the random-oracle model (ROM) or ideal-cipher model (ICM), often combining idealized components with components analyzed under standard assumptions [16,9,30,17]. Although these models enable efficient constructions and clean analyses, their guarantees need not survive concrete instantiation. Mine-matsu and Inoue [42] show component dependencies that invalidate the expected security of several such transforms, their AEaH-KS construction addresses this through separate AE and commitment keys. Concerns about idealization also hinder adoption: the UK National Cyber Security Centre does not target CMT-4 in its proposed AE functionality due to concerns about ideal-cipher-based constructions [15]. An alternative uses collision-resistant pseudorandom functions (cPRFs) [8,46], which can be instantiated with SHA-256 or SHA-3 [8].

SYMMETRIC PRIMITIVES AND COMMITMENT. Most AE schemes use (tweakable) block ciphers, but a single evaluation of either primitive cannot provide full context commitment. For each fixed key and tweak, the primitive permutes the  $n$ -bit

message space: any ciphertext can therefore be inverted under an alternative key and tweak to obtain a matching plaintext. Tweaks provide domain separation, but not the redundancy needed to bind the key, tweak, and plaintext.

Non-invertible keyed primitives avoid this structural limitation, examples include the fixed-length expanding PRF family **ButterKnife** [4] and the length-preserving ZIP ciphers [28]. Forkciphers [5] offer another expanding primitive. They process an  $n$ -bit input through initial rounds, fork the state, and continue along two branches, each producing an  $n$ -bit output. Each branch is invertible: either output, together with the key and tweak, determines the input and the other output. However, under an alternative key and tweak, inverting one branch fixes a candidate input whose other output must still match. This consistency condition enables meaningful collision resistance across adversarially chosen keys, tweaks, and inputs, suitable for higher-level commitment security.

The Iterate-Fork-Iterate (IFI) analysis of forkciphers [5] shows that three independent random permutations, one before the fork and one per branch, yield a perfect forkcipher: under a fixed key, its external interface is indistinguishable from two independent random permutations with advantage zero. This **prtfp** notion does not, however, guarantee collision resistance across adversarially chosen keys and tweaks. In particular, it does not capture the cross-key aspect of finding  $(K, T, M) \neq (K', T', M')$  with  $F(K, T, M) = F(K', T', M')$ . Our notions **fCR** and **zmCR** address this gap.

Forkciphers are attractive for committing AE for two reasons. First, their branches can support encryption and commitment in one primitive: one may carry encrypted data while the other supplies redundancy binding the context. Second, concrete designs, e.g., **ForkSkinny** [5], have numerous applications and established cryptanalysis. Forkciphers underpin the **SAEF**, **PAEF**, **RPAEF**, **ZAFE**, and **Eevee** IoT-to-Cloud AE modes, **GCTR** encryption, and **Sonikku** authentication [5,4,13,3,12], as well as leakage-resilient constructions including **FEDT**, **ForkDTE1/2**, and **HOP-1** [23,11,32]. These applications and the existing cryptanalysis motivate combining standard-model reductions with dedicated analysis of the required collision-resistance assumptions.

Rahman et al. [48] show that **SAEF**, **PAEF**, and the **Eevee** family lack committing security. Their results evidence that the surrounding mode must expose enough of the two-branch output and process the context so that a ciphertext collision yields a forkcipher collision. Hence, we take a primitive-centered approach grounded in cryptanalysis, identifying the collision-resistance properties needed for commitment so that the higher-level constructions and modes preserve them.

## 1.1 Contributions

We formalize forkcipher collision resistance, lift it through cPRF constructions, and apply these to context-committing AE. Our construction-level proofs reduce to explicit pseudorandomness and collision-resistance assumptions, without random oracles or ideal ciphers. Figure 1 summarizes the following contributions.

- **Collision resistance of forkciphers (Section 3).** We introduce forkcipher collision resistance (**fCR**), covering distinct key/tweak/message tuples, and

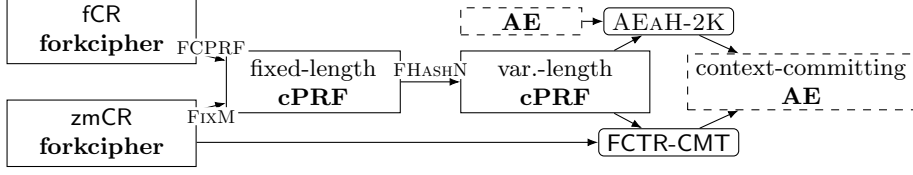


Fig. 1: Solid, dashed, and rounded boxes denote new or adapted notions, existing notions, and constructions, respectively. The upper path gives context-committing AE via AEAH-2K; the lower gives FCTR-CMT, using the forkcipher directly and through FHASHN. Only collision resistance dependencies are shown.

zero-message collision resistance (zmCR), fixing the message to zero. In the external ideal-forkcipher model, matching attacks and lower bounds give classical query complexities  $\Theta(2^{n/2})$  and  $\Theta(2^n)$ , respectively, for sufficiently large key/tweak spaces.

- **Cryptanalysis of ForkSkinny-128-384 (Section 6).** Our analysis supports 64-bit fCR and 128-bit zmCR security for the full-round primitive. We bound active S-box counts in differential collision attacks, including fixed-message attacks relevant to zmCR, and exhibit reduced-round collisions.

Supplementary analyses of ButterKnife and ZIP-AES appear in Appendix F and G. We find collisions in full-round ButterKnife restricted to one of eight branches in time  $2^{13}$ , and in reduced-round ZIP-AES-3-2 in time  $2^7$ . These results do not directly threaten eight-branch ButterKnife or full-round ZIP-AES, but our evidence is insufficient to claim their full-round, full-output collision resistance at the generic levels suggested by their designs. We therefore focus on ForkSkinny-128-384, whose collision security our analysis supports. Our cryptanalysis code is available on GitHub<sup>3</sup>.

- **Fixed-length collision-resistant and nonce-based PRFs (Section 4).** Our constructions FCPRF and FixM lift forkcipher collision resistance to cPRFs. Combining pseudorandomness and collision resistance, the cPRF notion also accommodates non-invertible keyed primitives such as ButterKnife and ZIP ciphers [4,28]; related formulations appear in [8,46].

We introduce nonce-based PRF security (nPRF), prohibiting repetition of a designated  $n$ -bit nonce portion of the input, to capture the nonce-respecting setting of our constructions. FCPRF achieves  $n$ -bit nPRF security, while collisions limit its ordinary PRF (PRF) security to approximately  $n/2$  bits.

- **A variable-input-length cPRF (Section 4).** We construct FHASHN from a fixed-length cPRF, proving collision resistance, pseudorandomness, and nonce-based pseudorandomness relative to the underlying cPRF properties. Its iterative structure resembles the FEDT hash function [23], but provides a keyed variable-input-length cPRF interface with standard-model reductions under strictly weaker assumptions than idealizing the primitive.

<sup>3</sup> <https://github.com/isec-tugraz/forkcipher-analysis>

- **A generic context-committing AE transform (Section 5).** We propose AEAH-2K, combining AE with a variable-input-length cPRF, and prove confidentiality, authenticity, and CMT-4 security under the security of AE and the cPRF properties of FHASHN.  
The expanded key  $(K_E, K_H)$  serves AE and commitment, respectively. Including  $K_E$  in the commitment input binds both components, reducing CMT-4 security directly to collision resistance. Independently, Minematsu and Inoue [42] identify dependencies invalidating mixed-model transforms and propose AEAH-KS, whose PRG-derived keys additionally require right-half collision resistance. We also discuss a master-key variant requiring pseudorandomness and collision resistance of key derivation.  
Using FCPRF within FHASHN yields  $n$ -bit confidentiality and approximately  $n/2$ -bit authenticity and CMT-4 security; FIXM yields  $n$  bits for all three, under the corresponding primitive assumptions (including zmCR for FIXM) and AE privacy and authenticity guarantees.
- **A forkcipher-based committing-AE mode (Section 5).** We introduce FCTR-CMT, built entirely from a single forkcipher, and prove nonce-respecting privacy, authenticity, and CMT-4 security. A domain-separated call derives separate encryption and hashing keys. Counter-mode encryption uses the full  $2n$ -bit output to process  $2n$ -bit message blocks; authentication and commitment apply FHASHN to an injective encoding of the full encryption context.

## 2 Background

In this section we introduce various notations used in our security proofs and briefly define committing security. We give the specification of ForkSkinny and background on differential cryptanalysis.

### 2.1 Notation

We write  $[n]$  for the set  $\{1, \dots, n\}$ ,  $\text{Perm}(n)$  for the set of all permutations on  $\{0, 1\}^n$ ,  $\leftarrow$  for assignment, and  $\leftarrow \$$  for uniform random sampling. We write  $\{0, 1\}^{\geq n}$  for the set of bitstrings of length at least  $n$ . Let  $k$  denote the key length and let  $\mathcal{K} := \{0, 1\}^k$  be the key space.

A forkcipher  $F$  consists of two deterministic algorithms: a forward algorithm  $F : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$  and an inverse algorithm  $F^{-1} : \mathcal{K} \times \mathcal{T} \times \{0, 1\}^n \times \{0, 1\} \rightarrow \{0, 1\}^{2n}$ . The forward algorithm takes a key  $K \in \mathcal{K}$ , a tweak  $T \in \mathcal{T}$ , and a message block  $M \in \{0, 1\}^n$ , and returns  $F(K, T, M) = C_0 \parallel C_1$ . The inverse algorithm takes  $K$ ,  $T$ , one output block  $C_b$ , and a branch index  $b \in \{0, 1\}$ . It returns  $F^{-1}(K, T, C_b, b) = M \parallel C_{1-b}$ .

The advantage of an adversary  $\mathcal{A}$  in distinguishing  $F$  from a random forked permutation is defined using the games in Figure 2 as

$$\text{Adv}_F^{\text{prtfp}}(\mathcal{A}) := \left| \Pr[\mathcal{A}^{\text{prtmfp-real}_F} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{prtmfp-ideal}_F} \Rightarrow 1] \right|.$$

The definition follows the multiforkcipher security notion of [3].

| Game <b>prtmfp-real</b> <sub>F</sub>                                                                        | Game <b>prtmfp-ideal</b> <sub>F</sub>                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $K \leftarrow \$ \mathcal{K}$<br>$b \leftarrow \mathcal{A}^{\mathcal{E}, \mathcal{D}}$<br><b>return</b> $b$ | $K \leftarrow \$ \mathcal{K}$<br><b>for</b> $T \in \mathcal{T}$ <b>do</b> $\pi_{T,0}, \pi_{T,1} \leftarrow \$ \text{Perm}(n)$<br>$b \leftarrow \mathcal{A}^{\mathcal{E}, \mathcal{D}}$<br><b>return</b> $b$ |
| <b>Oracle</b> $\mathcal{E}(T, X)$<br><b>return</b> $F(K, T, X)$                                             | <b>Oracle</b> $\mathcal{E}(T, X)$<br><b>return</b> $\pi_{T,0}(X) \parallel \pi_{T,1}(X)$                                                                                                                    |
| <b>Oracle</b> $\mathcal{D}(T, Y, \beta)$<br><b>return</b> $F^{-1}(K, T, Y, \beta)$                          | <b>Oracle</b> $\mathcal{D}(T, Y, \beta)$<br>$X \leftarrow \pi_{T,\beta}^{-1}(Y)$<br><b>return</b> $X \parallel \pi_{T,1-\beta}(X)$                                                                          |

Fig. 2: Games **prtmfp-real**<sub>F</sub> and **prtmfp-ideal**<sub>F</sub> defining the pseudorandom tweakable-forked-permutation security of  $F$ .

The ideal game in Figure 2 is the fixed-secret-key restriction of the context-indexed ideal forkcipher family from Section 3.1. Specifically, after sampling  $K \leftarrow \$ \mathcal{K}$ , we restrict the ideal family to contexts of the form  $U = (K, T)$ . By Lemma 1, this restriction gives two independent uniform branch permutations for every tweak  $T$ , as in the ideal game of Figure 2.

Both games provide only the external forkcipher interface. Direct access to the internal permutations of the ideal IFI construction is considered separately in Appendix A. In the real game,  $\mathcal{A}$  accesses  $F(K, \cdot, \cdot)$  and  $F^{-1}(K, \cdot, \cdot, \cdot)$  for a uniformly sampled secret key  $K \leftarrow \$ \mathcal{K}$ . In the ideal game, these oracles are replaced by two independent uniform permutations for every tweak. The permutations are sampled lazily when a tweak is queried for the first time.

**Authenticated encryption.** A nonce-based authenticated-encryption scheme with associated data is a pair of deterministic algorithms  $\text{AE} = (\text{AE.Enc}, \text{AE.Dec})$ . For a key  $K \in \{0, 1\}^k$ , nonce  $N \in \{0, 1\}^n$ , associated data  $A \in \{0, 1\}^*$ , and message  $M \in \{0, 1\}^*$ , encryption returns  $C \leftarrow \text{AE.Enc}(K, N, A, M)$ . Decryption returns either a message or  $\perp$ . Correctness requires  $\text{AE.Dec}(K, N, A, \text{AE.Enc}(K, N, A, M)) = M$  for every valid  $(K, N, A, M)$ . A privacy adversary is nonce-respecting, it may not repeat a nonce in its encryption queries. In the authenticity experiment, encryption queries are also nonce-respecting, whereas decryption queries may repeat nonces. A forgery is a query  $(N, A, C)$  such that  $\text{AE.Dec}(K, N, A, C) \neq \perp$  and  $(N, A, C)$  was not previously returned by the encryption oracle.

Let oracles  $\mathcal{E}_k(\cdot) = \text{AE.Enc}_k(\cdot)$ ,  $\mathcal{D}_k(\cdot) = \text{AE.Dec}_k(\cdot)$ . We define the AE privacy against a (nonce-respecting) adversary  $\mathcal{A}$  as

$$\text{Adv}_{\text{AE}}^{\text{priv}}(\mathcal{A}) = \left| \Pr[K \leftarrow \$ \{0, 1\}^k : \mathcal{A}^{\mathcal{E}_K(\cdot)} = 1] - \Pr[\mathcal{A}^{\$}(\cdot) = 1] \right|$$

where  $\$(N, A, M)$  returns a random string of length  $|\mathcal{E}_K(N, A, M)|$ , and  $\mathcal{A}$  is nonce-respecting  $N$ . We define AE authenticity against an adversary  $\mathcal{A}$  as

$$\text{Adv}_{\text{AE}}^{\text{auth}}(\mathcal{A}) = \left| \Pr[K \leftarrow_{\$} \{0, 1\}^k : \mathcal{A}^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)} \text{ forges}] \right|$$

where we say that the adversary *forges* if it queries  $\mathcal{D}$  on  $(N, A, C)$  such that  $(N, A, C)$  was not previously returned by  $\mathcal{E}$  and  $\mathcal{D}(N, A, C) \neq \perp$ . For *auth*,  $\mathcal{A}$  is not allowed to use the same nonce  $N$  for multiple  $\mathcal{E}$  queries (but may repeat nonces during  $\mathcal{D}$  queries).

## 2.2 Committing Security

Depending on the use case, we might require different types of committing security. Bellare and Hoang [8] formalize key committing security (CMT-1) and the stronger context committing security (CMT-4).

**Definition 1 (Committing Security).** *The CMT-1 and CMT-4 advantages of an adversary  $\mathcal{A}$  against AE are defined as follows*

$$\begin{aligned} \text{Adv}_{\text{AE}}^{\text{CMT-1}}(\mathcal{A}) &:= \Pr \left[ \begin{array}{l} (K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2) \leftarrow \mathcal{A} : \\ K_1 \neq K_2 \wedge \\ \text{AE.Enc}(K_1, N_1, A_1, M_1) = \text{AE.Enc}(K_2, N_2, A_2, M_2) \end{array} \right], \\ \text{Adv}_{\text{AE}}^{\text{CMT-4}}(\mathcal{A}) &:= \Pr \left[ \begin{array}{l} (K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2) \leftarrow \mathcal{A} : \\ (K_1, N_1, A_1, M_1) \neq (K_2, N_2, A_2, M_2) \wedge \\ \text{AE.Enc}(K_1, N_1, A_1, M_1) = \text{AE.Enc}(K_2, N_2, A_2, M_2) \end{array} \right]. \end{aligned}$$

## 2.3 Description of ForkSkinny

Forkciphers were introduced by Andreeva et al. for efficient authenticated encryption, particularly of short messages [5]. It is a tweakable primitive takes an  $n$ -bit input and produces a  $2n$ -bit output. ForkSkinny was proposed as a family of forkcipher instantiations based on SKINNY [6]. These instances follow the *iterate-fork-iterate* [5] design. We describe ForkSkinny in Appendix E.1. In this paper, we focus on the variant ForkSkinny-128-384, which takes a 128-bit message  $M$ , a 384-bit tweakkey  $K \parallel T$ , and produces a 256-bit ciphertext  $C = C_0 \parallel C_1$ . We do not consider the 64-bit-message variant of ForkSkinny because it offers a lower generic security against CMT-4 attacks.

## 2.4 Differential Cryptanalysis

Differential cryptanalysis [14] is one of the major statistical techniques used to assess the security of cryptographic primitives. Given a plaintext pair with a chosen difference, we try to predict the output difference. The main idea is to track how differences evolve on a round-by-round basis in a differential characteristic. The probability of this distinguisher is mainly influenced by the number of active S-boxes (S-boxes with a non-zero difference) throughout the cipher.

In the context of hash functions, we are interested in differential characteristics with zero output difference, i.e. collisions, and values that satisfy these differences. As the message block typically takes role of the key, we do not only rely on chance to satisfy a differential characteristic. An attacker can use the additional message control for more powerful, but also more complicated, analysis. Many popular hash functions, like MD4/5 [55,57], SHA-1 [56,53,52], and SHA-2[39,40,24], have been analyzed using differential cryptanalysis.

The complexity of finding optimal or even just good attacks invites computer aided search. For hash functions, dedicated frameworks [39,52] have been used to find collisions. To argue security of a cryptographic primitive we usually rely on bounds on the complexity of parts of certain attacks. Mixed-integer linear programming (MILP) allows formulating the task of finding a high probability differential characteristic as an optimization problem which provides bounds and good differential characteristics [43]. In addition to MILP, Boolean satisfiability (SAT) solvers are often used to find good characteristics [47,44]. Further, SAT solvers can also find concrete values leading to a collision [50,37].

### 3 Forkcipher Collision Resistance

We introduce forkcipher collision resistance (fCR) and zero-message collision resistance (zmCR) and prove matching upper/lower bounds on the generic attacks for both notions. In Section 6, we provide of ForkSkinny-128-384 for these notions.

**Definition 2 ((Zero-Message) Forkcipher Collision Resistance).** *The forkcipher collision resistance (fCR) and zero-message collision resistance (zmCR) advantage of  $\mathcal{A}$  against a forkcipher  $F$  as follows:*

$$\begin{aligned} \text{Adv}_F^{\text{fcr}}(\mathcal{A}) &:= \Pr \left[ \begin{array}{c} ((K, T, M), (K', T', M')) \leftarrow \mathcal{A} : \\ (K, T, M) \neq (K', T', M') \wedge F(K, T, M) = F(K', T', M') \end{array} \right], \\ \text{Adv}_F^{\text{zmcr}}(\mathcal{A}) &:= \Pr \left[ \begin{array}{c} ((K, T), (K', T')) \leftarrow \mathcal{A} : \\ (K, T) \neq (K', T') \wedge F(K, T, 0^n) = F(K', T', 0^n) \end{array} \right]. \end{aligned}$$

Every zmCR collision is an fCR collision, but the converse does not hold. For zmCR, the message inputs are fixed to  $0^n$ , while an fCR adversary may choose both message inputs. fCR enables stronger generic attacks, as shown below.

#### 3.1 Ideal-Forkcipher Model

We consider the ideal version of the iterate-fork-iterate (IFI) forkcipher family indexed by the key/tweak context. Let  $\mathcal{U} := \mathcal{K} \times \mathcal{T}$  denote the context space. For every  $U = (K, T) \in \mathcal{U}$ , sample  $P_U^0, P_U^1, P_U^2 \leftarrow \$ \text{Perm}(n)$  and define the two branch permutations by  $Q_{U,0} = P_U^1$  and  $Q_{U,1} = P_U^2$ .

**Definition 3 (Ideal IFI forkcipher).** *The ideal iterate-fork-iterate forkcipher IFI-FC<sub>n</sub> is defined, for  $U = (K, T) \in \mathcal{U}$  and  $M \in \{0, 1\}^n$ , by*

$$\text{IFI-FC}_n(K, T, M) = Q_{U,0}(P_U^0(M)) \parallel Q_{U,1}(P_U^0(M)).$$

For  $b \in \{0, 1\}$ , its inverse interface is defined by  $\text{IFI-FC}_n^{-1}(K, T, C_b, b) = M \parallel C_{\bar{b}}$ , where  $\bar{b} = 1 - b$  and  $S = Q_{U,b}^{-1}(C_b)$ ,  $M = (P_U^0)^{-1}(S)$ ,  $C_{\bar{b}} = Q_{U,\bar{b}}(S)$ .

We distinguish two oracle interfaces for  $\text{IFI-FC}_n$ : 1. *External interface*: the adversary accesses only the forward and inverse forkcipher algorithms; and 2. *Internal interface*: the adversary additionally accesses  $P_U^j$  and  $(P_U^j)^{-1}$ , where  $j \in \{0, 1, 2\}$ , for adversarially chosen contexts  $U \in \mathcal{U}$ . The generic results in this section concern the *external interface*. Results allowing internal permutation queries are explicitly labeled as such.

**Lemma 1 (Equivalent external representation).** *Under the external interface,  $\text{IFI-FC}_n$  is identically distributed to a family that, independently for every  $U = (K, T) \in \mathcal{U}$ , samples  $\pi_{U,0}, \pi_{U,1} \leftarrow \$ \text{Perm}(n)$  and answers a forward query  $(K, T, M)$  with  $\pi_{U,0}(M) \parallel \pi_{U,1}(M)$ . For  $b \in \{0, 1\}$ , an inverse query  $(K, T, C_b, b)$  is answered with  $M \parallel C_{\bar{b}}$ , where*

$$\bar{b} = 1 - b, \quad M = \pi_{U,b}^{-1}(C_b), \quad C_{\bar{b}} = \pi_{U,\bar{b}}(M).$$

*Proof.* For every  $U \in \mathcal{U}$  and  $b \in \{0, 1\}$ , let  $\pi_{U,b} := Q_{U,b} \circ P_U^0$ . Fix an arbitrary value of  $P_U^0$ . Right-composition by  $P_U^0$  is a bijection on  $\text{Perm}(n)$ . Hence, if  $Q_{U,b}$  is a uniform random permutation, then  $\pi_{U,b}$  is also a uniform random permutation. Moreover,  $Q_{U,0}$  and  $Q_{U,1}$  are sampled independently. Therefore, conditioned on any value of  $P_U^0$ , the permutations  $\pi_{U,0}$  and  $\pi_{U,1}$  are independent and uniform. Independence across distinct contexts follows from the independent sampling of the permutation triples associated with every  $U \in \mathcal{U}$ .

Finally, the forward and inverse interfaces defined by the  $\pi_{U,b}$  permutations agree exactly with the external forward and inverse interfaces of  $\text{IFI-FC}_n$ .  $\square$

Lemma 1 concerns only the external forkcipher interface. Under the internal one, the adversary also observes the common intermediate permutation  $P_U^0$ , hence, the lemma does not identify the corresponding internal transcripts.

### 3.2 Generic Security Bounds

An external forward or inverse query reveals a complete record

$$(U, M, C_0, C_1), \quad \text{IFI-FC}_n(U, M) = C_0 \parallel C_1.$$

Indeed, a forward query directly returns both output blocks, while an inverse query returns the message and the output block on the other branch. Repeated queries reveal no new record.

After  $\mathcal{A}$  outputs its two candidate inputs, we append to the transcript at most two forward evaluations needed to verify them. Below  $q$  denotes the total number of external evaluations, including the verification evaluations.

**Theorem 1 (Generic security of the ideal forkcipher).** *Let  $\mathcal{A}$  make at most  $q < 2^n$  external forward or inverse evaluations of  $\text{IFI-FC}_n$ , including the evaluations needed to verify its output. Then*

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{fcr}}(\mathcal{A}) \leq \frac{\binom{q}{2}}{2^n - q} \quad \text{and} \quad \text{Adv}_{\text{IFI-FC}_n}^{\text{zmc}}(\mathcal{A}) \leq \frac{\binom{q}{2}}{(2^n - q)^2}.$$

*In particular, if  $q \leq 2^{n-1}$ , then*

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{fcr}}(\mathcal{A}) \leq \frac{q^2}{2^n} \quad \text{and} \quad \text{Adv}_{\text{IFI-FC}_n}^{\text{zmc}}(\mathcal{A}) \leq \frac{2q^2}{2^{2n}}.$$

*Proof.* We use the equivalent external representation from Lemma 1. Thus, independently for every context  $U$ , the ideal forkcipher is described by two independent permutations  $\pi_{U,0}, \pi_{U,1} \leftarrow \$ \text{Perm}(n)$ . For a context  $U$ , let  $\mathcal{D}_U$  be the set of message values occurring in records already revealed under  $U$ , and let  $\mathcal{R}_{U,b}$  be the corresponding set of output values on branch  $b$ . Since the branch functions are permutations, then  $r_U = |\mathcal{D}_U| = |\mathcal{R}_{U,0}| = |\mathcal{R}_{U,1}|$  where,  $r_U \leq q$ .

Conditioned on the current transcript, the unrevealed parts of  $\pi_{U,0}$  and  $\pi_{U,1}$  remain independent uniform completions of the corresponding partial permutations. Consequently, a forward query  $(U, M)$  with  $M \notin \mathcal{D}_U$  samples  $\pi_{U,b}(M) \leftarrow \$ \{0, 1\}^n \setminus \mathcal{R}_{U,b}$  independently for  $b = 0, 1$ .

Similarly, consider an inverse query  $(U, Y, b)$ . Since  $Y \notin \mathcal{R}_{U,b}$ , its preimage is sampled as  $M = \pi_{U,b}^{-1}(Y) \leftarrow \$ \{0, 1\}^n \setminus \mathcal{D}_U$ , after which  $\pi_{U,\bar{b}}(M) \leftarrow \$ \{0, 1\}^n \setminus \mathcal{R}_{U,\bar{b}}$ . If the queried message or output value has already occurred, the query merely returns a previously revealed value and can be ignored.

*fCR case.* For every fixed context  $U$ , the map  $M \mapsto \pi_{U,0}(M) \parallel \pi_{U,1}(M)$  is injective. Hence, an fCR collision necessarily involves two distinct contexts.

Consider the first new record that completes an fCR collision, and condition on the transcript immediately before this record is revealed. Fix a previous record from a distinct context with output  $C_0^* \parallel C_1^*$ . If the new record is produced by a forward query, both output blocks are fresh. The probability that they equal  $C_0^*$  and  $C_1^*$  is at most  $\frac{1}{(2^n - q)^2} \leq \frac{1}{2^n - q}$ . If the new record is produced by an inverse query on branch  $b$ , the adversary may choose the queried block to be  $C_b^*$ . This forces one output block to match, but the output on the other branch remains uniform over at least  $2^n - q$  values. Hence, the probability of matching the complete previous output is at most  $\frac{1}{2^n - q}$ . Before the  $i$ -th new record is revealed, there are at most  $i - 1$  previous records. Therefore, by charging a collision to the first record that completes it and applying the union bound,

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{fcr}}(\mathcal{A}) \leq \sum_{i=1}^q \frac{i-1}{2^n - q} = \frac{\binom{q}{2}}{2^n - q}.$$

*zmCR case.* A zmCR collision consists of two records from distinct contexts, both having message  $0^n$  and the same two output blocks. Again consider the first new record that completes such a collision and fix a previous zero-message record

with output  $C_0^* \parallel C_1^*$ . For a forward query at  $0^n$ , both branch outputs are fresh and independent, unless the corresponding record was already known. Thus, the probability of matching the fixed previous record is at most  $\frac{1}{(2^n - q)^2}$ . Now consider an inverse query on branch  $b$ . The adversary may set its queried output block to  $C_b^*$ . However, for the resulting record to be a zero-message record, the randomly determined preimage must equal  $0^n$ . This occurs with probability at most  $1/(2^n - q)$ . Conditioned on this event, the output of the other, independently sampled branch must equal  $C_b^*$ , which occurs with probability at most another  $1/(2^n - q)$ . If  $0^n$  was already present in  $\mathcal{D}_U$ , then its complete two branch record was already revealed by the earlier external query. Hence, a later query cannot be the first query completing the collision. It follows that, in every relevant case, the conditional probability of matching a fixed previous zero message record is at most  $\frac{1}{(2^n - q)^2}$ . Applying the same first-completion argument gives

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{zmcr}}(\mathcal{A}) \leq \sum_{i=1}^q \frac{i-1}{(2^n - q)^2} = \frac{\binom{q}{2}}{(2^n - q)^2}.$$

For  $q \leq 2^{n-1}$ , we have  $2^n - q \geq 2^{n-1}$ . Thus,  $\frac{\binom{q}{2}}{2^n - q} \leq \frac{q^2}{2^n}$  and  $\frac{\binom{q}{2}}{(2^n - q)^2} \leq \frac{q^2}{2^{2n}}$ .  $\square$

### 3.3 Matching Generic Attacks

We now give attacks matching the bounds from Theorem 1. We assume that the context space  $\mathcal{U} = \mathcal{K} \times \mathcal{T}$  contains at least  $q$  distinct contexts.

**Lemma 2 (Generic fCR attack).** *There is an fCR attack making  $q$  inverse-forkcipher queries with success probability about  $\frac{q^2}{2^n}$ .*

*Proof.* Fix  $C_0$  and choose  $q$  distinct  $U_i = (K_i, T_i)$ . For each context, compute  $(M_i, C_{1,i}) = \text{IFI-FC}_n^{-1}(K_i, T_i, C_0, 0)$ . The values  $C_{1,i}$  are independent uniform  $n$ -bit strings. Therefore, a collision  $C_{1,i} = C_{1,j}$  occurs with probability about  $q^2/2^n$ . Such a collision produces

$$\text{IFI-FC}_n(K_i, T_i, M_i) = C_0 \parallel C_{1,i} = C_0 \parallel C_{1,j} = \text{IFI-FC}_n(K_j, T_j, M_j),$$

which is a valid fCR collision because  $U_i \neq U_j$ .  $\square$

**Lemma 3 (Generic zmCR attack).** *There is a zmCR attack making  $q$  forward-forkcipher queries with success probability about  $\frac{q^2}{2^{2n}}$ .*

*Proof.* Choose  $q$  distinct contexts  $U_i = (K_i, T_i)$  and compute  $Y_i := \text{IFI-FC}_n(K_i, T_i, 0^n)$ . The values  $Y_i$  are independent uniform  $2n$ -bit strings. A birthday collision therefore occurs with probability about  $q^2/2^{2n}$ . If  $Y_i = Y_j$ , then

$$\text{IFI-FC}_n(K_i, T_i, 0^n) = \text{IFI-FC}_n(K_j, T_j, 0^n),$$

which is a valid zmCR collision because  $U_i \neq U_j$ .  $\square$

The fCR attack fixes one output block and searches for a collision in the remaining  $n$ -bit block. The zmCR attack cannot freely choose the message and must instead search for a collision in the complete  $2n$ -bit output. Together with Theorem 1, these attacks show that the generic complexities are about  $2^{n/2}$  queries for fCR and  $2^n$  queries for zmCR.

## 4 Collision-Resistant Pseudorandom Functions

We lift the forkcipher collision resistance notions to fixed-input-length collision-resistant PRFs and then extend them to variable-length inputs through FHASHN. A collision-resistant PRF combines collision resistance with standard and nonce-respecting pseudorandomness definitions. Related notions appear in [8,46].

**Definition 4 (Collision-resistant PRF).** Let  $H : \{0,1\}^k \times D \rightarrow \{0,1\}^\ell$  and let  $\rho : D \rightarrow \{0,1\}^\ell$  be a random function. The collision resistance advantage (hCR) and pseudorandomness (PRF) advantage  $a$  of  $\mathcal{A}$  are defined as follows:

$$\text{Adv}_H^{\text{hcr}}(\mathcal{A}) = \Pr \left[ \begin{array}{l} (K, M), (K', M') \leftarrow \mathcal{A} : \\ (K, M) \neq (K', M') \wedge \\ H(K, M) = H(K', M') \end{array} \right],$$

$$\text{Adv}_H^{\text{prf}}(\mathcal{A}) = \left| \Pr_{K \leftarrow \{0,1\}^k} [\mathcal{A}^{H(K, \cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^\rho \Rightarrow 1] \right|.$$

Nonce-respecting pseudorandomness (nPRF). If  $D \subseteq \{0,1\}^{\geq n}$ , the nPRF advantage is defined by the same experiment, restricted to adversaries that do not repeat the first  $n$  bits of their oracle inputs.

As every nPRF adversary is also a valid PRF adversary,  $\text{Adv}_H^{\text{nprf}}(\mathcal{A}) \leq \text{Adv}_H^{\text{prf}}(\mathcal{A})$ .

*Modelling remark.* Because the key is adversarially chosen in the hCR experiment, a fixed compressing function cannot exclude non-uniform adversaries that hard-code a collision. We therefore interpret hCR as a concrete computational assumption, following the human-ignorance approach [49]. The same applies to fCR and zmCR. A fully formal alternative would introduce an additional randomly sampled public parameter. See Appendix C for further discussion.

### 4.1 Fixed-Length cPRF Constructions

We propose two simple cPRF primitives for fixed-length messages: FIXM and FCPRF. We show their security based on the fCR and zmCR properties of the underlying forkcipher.

**Definition 5 (FCPRF and FixM).** Let  $F : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$  be a forkcipher and  $T \in \{0,1\}^{2n}$  and  $M \in \{0,1\}^n$ . Define

$$\begin{aligned} \text{FixM} : \{0,1\}^n \times \{0,1\}^{2n} &\rightarrow \{0,1\}^{2n}, & \text{FixM}(K, X) &= F(K, X, 0^n), \\ \text{FCPRF} : \{0,1\}^n \times \{0,1\}^{3n} &\rightarrow \{0,1\}^{2n}, & \text{FCPRF}(K, T \| M) &= F(K, T, M). \end{aligned}$$

**Theorem 2 (PRF security of FCPRF and FixM).** *Let  $F : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$  be a forkcipher. For every adversary  $\mathcal{A}$  making at most  $q$  queries, for each of the following inequalities there exists an adversary  $\mathcal{B}$  running in comparable time and making at most  $q$  forkcipher queries:*

$$\begin{aligned} \text{Adv}_{\text{FCPRF}}^{\text{prf}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}) + \frac{q(q-1)}{2^n}, & \text{Adv}_{\text{FixM}}^{\text{prf}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}), \\ \text{Adv}_{\text{FCPRF}}^{\text{nprf}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}), & \text{Adv}_{\text{FixM}}^{\text{nprf}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}). \end{aligned}$$

*Proof sketch.* After replacing  $F$  by its ideal external interface, distinct queries to FixM use distinct tweaks. The same holds for nonce-respecting queries to FCPRF, because the nonce is contained in the tweak. In these cases, the answers are distributed exactly as those of a random function. For unrestricted FCPRF queries, the only difference is that, among queries using the same tweak, neither branch permutation can repeat an output block. A switching argument bounds the probability that a random function exhibits such a within-tweak repetition by  $\frac{2\binom{q}{2}}{2^n} = \frac{q(q-1)}{2^n}$ . The full proof is given in Appendix B.1.  $\square$

**Theorem 3 (hCR security of FCPRF and FixM).** *Let  $F : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^n \rightarrow \{0,1\}^{2n}$  be a forkcipher. For any adversary  $\mathcal{A}$  there are  $\mathcal{B}, \mathcal{C}$  running in about the same time such that*

$$\text{Adv}_{\text{FCPRF}}^{\text{hcr}}(\mathcal{A}) \leq \text{Adv}_F^{\text{fcr}}(\mathcal{B}), \quad \text{Adv}_{\text{FixM}}^{\text{hcr}}(\mathcal{A}) \leq \text{Adv}_F^{\text{zmcr}}(\mathcal{C}).$$

*Proof.* If  $\mathcal{A}$  outputs a hCR collision for FCPRF:  $(K, T \| M) \neq (K', T' \| M')$  and  $F(K, T, M) = F(K', T', M')$ . Then  $\mathcal{B}$  outputs  $(K, T, M), (K', T', M')$ , which is a fCR collision. Similarly, an hCR collision  $(K, X) \neq (K', X')$  for FixM satisfies  $F(K, X, 0^n) = F(K', X', 0^n)$ , and therefore gives a valid zmCR collision.  $\square$

## 4.2 Variable Input Length cPRF Constructions

In this section we introduce the cPRF scheme FHASHN (Figure 3). It is a mode of operation of our prior fixed-length cPRF constructions.

FHASHN iteratively processes the message blocks similar to the Merkle-Damgård construction. We first parse  $M = N \| \dot{M}$ , where  $|N| = n$ , and encode it as  $\dot{M} \| 1 \| 0^p \| \ell \| N \| K$ , where  $\ell = \langle |M| \rangle_n$  and  $p$  is the smallest nonnegative integer making the total length a multiple of  $d$ . This places the padding after  $\dot{M}$  while preserving the final key block and, for  $d = 2n$ , the final block  $N \| K$ . We ensure the last forkcipher call in the loop (via cPRF) uses the FHASHN key  $K$  as its forkcipher key. Further, for the nPRF property, we ensure that the first  $n$  bits of  $M$  (which may serve as a nonce for the nPRF property) are used in the tweak of that last forkcipher call, if FCPRF is used for cPRF.

When instantiating cPRF in Figure 3 with FCPRF, FHASHN structurally resembles the unnamed hash function in FEDT [23], which is also in Merkle-Damgård style. The main contribution of our FHASHN construction is enabling an additional purpose besides hashing, namely its use as a PRF function. For

this, FHASHN uses a different processing order of its input blocks than FEDT. Specifically, in FHASHN, the first input block  $K$  in FHASHN (which is treated as a key in the PRF proof, and as a regular message block in the collision resistance proof) is mapped to the key of the last forkecipher call.

In Figure 4, we illustrate FHASHN using a forkecipher as a cPRF. We now prove the cPRF security properties of FHASHN.

Throughout this subsection, we assume  $n \leq |M| < 2^n$ , so that  $M$  contains an  $n$ -bit nonce prefix and its length has an unambiguous  $n$ -bit representation.

---

**Procedure** FHASHN( $K, M$ )

---

$\ell \leftarrow \langle |M| \rangle_n$      $\parallel$   $n$ -bit encoding of  $|M|$   
 $N \parallel \dot{M} \leftarrow M$      $\parallel$   $|N| = n$   
 $p \leftarrow (-|\dot{M}| - 1 - 3n) \bmod d$      $\parallel$   $0 \leq p < d$   
 $M_1 \parallel \dots \parallel M_B \leftarrow_d \dot{M} \parallel 1 \parallel 0^p \parallel \ell \parallel N \parallel K$   
 $S \leftarrow 0^{2n}$   
**for**  $i = 1$  **to**  $B$  **do**  
     $m_i \parallel K_i \leftarrow M_i$      $\parallel$   $|K_i| = n, |m_i| = d - n$   
     $S \leftarrow \text{cPRF}(K_i, m_i \parallel S)$ .  
**return**  $S$

---

Fig. 3: The FHASHN construction, with key  $K \in \{0, 1\}^n$  and message  $M$  satisfying  $n \leq |M| < 2^n$ . The  $\text{cPRF} : \{0, 1\}^n \times \{0, 1\}^{n+d} \rightarrow \{0, 1\}^{2n}$  uses  $d \in \{n, 2n\}$ .

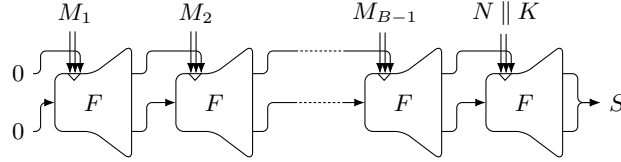


Fig. 4: FHashN, with cPRF is instantiated by FCPRF (and ForkSkinny-128-384).

*Injectivity of the encoding.* For  $M = N \parallel \dot{M}$ , where  $|N| = n$ , define

$$\text{enc}_d(K, M) = \dot{M} \parallel 1 \parallel 0^p \parallel \ell \parallel N \parallel K, \quad \ell = \langle |M| \rangle_n,$$

where  $p \in \{0, \dots, d-1\}$  satisfying  $|\dot{M}| + 1 + p + 3n \equiv 0 \pmod{d}$ . The encoded string is partitioned into  $d$ -bit blocks as in Figure 3. The encoding is injective. Its final  $3n$  bits determine  $\ell$ ,  $N$ , and  $K$ . Since  $n \leq |M| < 2^n$ , the length field uniquely determines  $|M|$  and hence  $|\dot{M}| = |M| - n$ . The first  $|\dot{M}|$  bits therefore recover  $\dot{M}$ , and the padding length  $p$  is uniquely determined. Consequently,  $\text{enc}_d(K, M) = \text{enc}_d(K', M') \implies (K, M) = (K', M')$ . The length field remains at a fixed offset

from the end: it is the third-to-last block when  $d = n$ , and the right  $n$ -bit half of the penultimate block when  $d = 2n$ .

**Theorem 4 (Collision resistance of FHashN).** *If verification of  $\mathcal{A}$ 's output processes at most  $\sigma$  encoded blocks, then there exists  $\mathcal{B}$  making at most  $\sigma$  cPRF evaluations such that*

$$\text{Adv}_{\text{FHashN}}^{\text{hcr}}(\mathcal{A}) \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B}).$$

*Proof.* Suppose  $\mathcal{A}$  outputs  $\text{FHashN}(K, M) = \text{FHashN}(K', M')$  for  $(K, M) \neq (K', M')$ . Let  $Z_1, \dots, Z_B$  and  $Z'_1, \dots, Z'_{B'}$  be their encoded block sequences, with

$$Z_i = m_i \| K_i, \quad S_i = \text{cPRF}(K_i, m_i \| S_{i-1}).$$

Compare the computations backwards, starting from  $S_B = S'_{B'}$ . Whenever  $S_i = S'_j$ , either the two cPRF inputs  $(K_i, m_i \| S_{i-1})$  and  $(K'_j, m'_j \| S'_{j-1})$  differ, yielding an hCR collision, or they agree, implying  $Z_i = Z'_j$  and  $S_{i-1} = S'_{j-1}$ .

If  $|M| = |M'|$ , the padding lengths and numbers of encoded blocks are equal. In the absence of an underlying collision, the backward comparison therefore makes all encoded blocks equal, contradicting injectivity of the encoding. If  $|M| \neq |M'|$ , the length fields differ and occur at the same fixed offset from the end. The backward comparison therefore encounters distinct blocks before either sequence is exhausted, yielding an underlying hCR collision. The two computations use at most  $\sigma$  evaluations.  $\square$

**Theorem 5 (nPRF security of FHashN).** *Assume cPRF in FHashN uses  $d = 2n$ . For any adversary  $\mathcal{A}$ , there exists  $\mathcal{B}$  that makes one nPRF-oracle query per query of  $\mathcal{A}$  and evaluates all nonfinal cPRF calls directly with*

$$\text{Adv}_{\text{FHashN}}^{\text{nprf}}(\mathcal{A}) \leq \text{Adv}_{\text{cPRF}}^{\text{nprf}}(\mathcal{B}).$$

*Proof.*  $\mathcal{B}$  runs  $\mathcal{A}$  and answers  $\mathcal{A}$ 's oracle queries by computing FHashN manually. Within FHashN, replace the final cPRF call with a call to  $\mathcal{B}$ 's nPRF oracle.  $\mathcal{B}$  returns  $\mathcal{A}$ 's output. Note that in FHashN, the first  $n$  bits of  $M$  ( $N$  in Figure 3) are within the last  $2n$ -bit block,  $M_B$ . The other half of  $M_B$  is  $K$ . Thus, the final cPRF call in FHashN is  $\text{cPRF}(K, N \| S)$  for some  $S$ . Our above replacement of this call with the oracle is possible because  $K$  is not used elsewhere and  $N$  never repeats since  $\mathcal{A}$  is an nPRF adversary. If  $\mathcal{B}$ 's oracle is the real world, then  $\mathcal{B}$  simulates the real world of the nPRF game of FHashN for  $\mathcal{A}$ . If  $\mathcal{B}$ 's oracle is the ideal world (i.e., a random function), the final cPRF call is made with a previously unqueried input (due to  $N$ ). Thus, the FHashN output will be uniform, simulating the ideal world for  $\mathcal{A}$ .  $\square$

We state Theorem 5 for  $d = 2n$  (i.e. cPRF = FCPRF) specifically because this is the case in which the generic route of Corollary 1 is lossy: Theorem 2 incurs a  $q(q-1)/2^n$  birthday term for the ordinary PRF security of FCPRF, which the dedicated argument above avoids by exploiting the nonce directly. For cPRF = FixM ( $d = n$ ), FixM's own PRF and hCR bounds already carry no birthday loss, so Corollary 1 suffices (see the instantiation in that corollary).

**Theorem 6 (PRF security of FHashN).** *For every adversary  $\mathcal{A}$  processing  $\sigma$  encoded blocks over  $q$  queries, there exist adversaries  $\mathcal{B}, \mathcal{C}$  such that*

$$\text{Adv}_{\text{FHashN}}^{\text{prf}}(\mathcal{A}) \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B}) + \text{Adv}_{\text{cPRF}}^{\text{prf}}(\mathcal{C}).$$

*Here,  $\mathcal{B}$  evaluates at most  $\sigma$  underlying calls, while  $\mathcal{C}$  makes  $q$  oracle queries and evaluates the nonfinal calls directly.*

*Proof.* Without loss of generality,  $\mathcal{A}$  makes no repeated outer queries, since repeated queries are answered consistently and provide no additional information. Let  $G_0$  be the real PRF experiment for FHashN.

$G_1$ . Adversary  $\mathcal{C}$  evaluates all nonfinal cPRF calls directly, using only the message, and forwards the final call of every query to its own oracle. If  $\mathcal{C}$ 's oracle is the real oracle  $\text{cPRF}(K, \cdot)$  for the actual FHashN key  $K$ , this is exactly  $G_0$ . If  $\mathcal{C}$ 's oracle is a lazy-sampled random function  $\rho$ , this defines a hybrid game  $G_1$  in which every query's final call is answered by  $\rho$ . Hence

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_{\text{cPRF}}^{\text{prf}}(\mathcal{C}).$$

*Bad event.* For every distinct outer query  $M$ , let  $Z_M = (K_B, m_B \| S_{B-1})$  be the complete key/input pair of the final cPRF call, as in Theorem 4. Let  $E$  be the event that, over the course of  $\mathcal{A}$ 's queries, two distinct queries  $M \neq M'$  satisfy  $Z_M = Z_{M'}$ . By the backward argument of Theorem 4, event  $E$  yields an underlying hCR collision, so  $\Pr[E] \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B})$  for a suitable adversary  $\mathcal{B}$  evaluating at most  $\sigma$  calls.

*Coupling until  $E$ .* Let  $G_2$  denote the ideal PRF experiment for FHashN, i.e.  $\mathcal{A}$  interacts with a fresh uniform random function  $\psi$  on FHashN's outer domain. Couple  $G_1$  and  $G_2$  by using the same underlying randomness for  $\rho$  and  $\psi$  as follows:  $\rho$  is sampled lazily, and each time a query in  $G_1$  reaches a fresh point  $Z_M$  not queried before, sample a fresh uniform value for both  $\rho(Z_M)$  and  $\psi(M)$  jointly. As long as every query in the run reaches a fresh point, i.e. as long as  $E$  has not yet occurred, the two games proceed identically, since both answer with independent fresh uniform values at every new query. The two games can only diverge once  $E$  first occurs (a repeated  $Z_M$  forces  $\rho$  to repeat a prior answer, while  $\psi$  need not). By the difference lemma,

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \Pr[E] \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B}).$$

Combining the two steps proves  $\text{Adv}_{\text{FHashN}}^{\text{prf}}(\mathcal{A}) \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B}) + \text{Adv}_{\text{cPRF}}^{\text{prf}}(\mathcal{C})$ .  $\square$

**Corollary 1 (nPRF security of FHashN via ordinary PRF security).** *For FHashN instantiated with any fixed-length cPRF (any  $d$ ), there exist  $\mathcal{B}, \mathcal{C}$  as in Theorem 6 such that*

$$\text{Adv}_{\text{FHashN}}^{\text{nprf}}(\mathcal{A}) \leq \text{Adv}_{\text{FHashN}}^{\text{prf}}(\mathcal{A}) \leq \text{Adv}_{\text{cPRF}}^{\text{hcr}}(\mathcal{B}) + \text{Adv}_{\text{cPRF}}^{\text{prf}}(\mathcal{C}).$$

*In particular, for  $\text{cPRF} = \text{FixM}$  ( $d = n$ ), combining with Theorem 2 and Theorem 3 gives the following with no birthday-type loss:*

$$\text{Adv}_{\text{FHashN}[\text{FixM}]}^{\text{nprf}}(\mathcal{A}) \leq \text{Adv}_F^{\text{zmcr}}(\mathcal{B}') + \text{Adv}_F^{\text{prtfp}}(\mathcal{C}').$$

*Proof.* The first inequality holds because every nPRF adversary is a valid PRF adversary. The second is Theorem 6. For  $\text{cPRF} = \text{FixM}$ , Theorem 2 gives  $\text{Adv}_{\text{FixM}}^{\text{prf}}(\mathcal{C}) \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{C}')$  and Theorem 3 gives  $\text{Adv}_{\text{FixM}}^{\text{hcr}}(\mathcal{B}) \leq \text{Adv}_F^{\text{zmcr}}(\mathcal{B}')$ , both without an additive loss term. Substituting proves the claim.  $\square$

## 5 Committing Authenticated Encryption

We introduce AEAH-2K, a direct two-key variant of AEAH.

Let  $\text{AE} = (\text{AE.Enc}, \text{AE.Dec})$  be an authenticated-encryption scheme with key space  $\{0, 1\}^{k_{\text{AE}}}$ , and let  $H : \{0, 1\}^{k_H} \times \mathcal{D} \rightarrow \{0, 1\}^t$  be a variable-input-length collision-resistant pseudorandom function. We require  $H$  to satisfy the hCR and PRF of Definition 4; for privacy, its nonce-respecting nPRF security suffices.

The scheme key is the expanded key  $K = K_E \| K_H \in \{0, 1\}^{k_{\text{AE}} + k_H}$ , where  $|K_E| = k_{\text{AE}}$  and  $|K_H| = k_H$ . Let  $\text{enc} : \mathcal{N} \times \{0, 1\}^{k_{\text{AE}}} \times \mathcal{A} \rightarrow \mathcal{D}$  be an injective encoding whose first  $n$  bits are  $N$ . This condition allows us to use the nPRF security of  $H$  in the nonce-respecting privacy analysis.

Our concrete instance sets  $H = \text{FHASHN}$ ,  $\text{enc}(N, K_E, A) = N \| K_E \| A$ ,  $k_H = n$ , and  $t = 2n$ . Since  $N$  and  $K_E$  have fixed lengths, this encoding is injective. We require  $n \leq n + k_{\text{AE}} + |A| < 2^n$ , as required by the domain of FHASHN.

### 5.1 Generic AE (AEaH-2K)

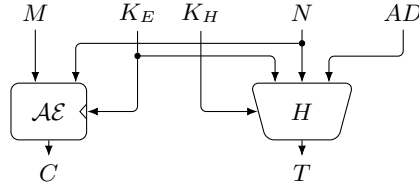


Fig. 5: The generic AEAH-2K construction with expanded key  $K = K_E \| K_H$  and a  $t$ -bit commitment tag.

Figure 5 shows the construction of AEAH-2K and in Appendix D we give the corresponding encryption and decryption algorithms. Encryption applies the underlying AE scheme with empty associated data and appends a commitment tag over the nonce, the AE key, and the external associated data:

$$C \leftarrow \text{AE.Enc}(K_E, N, \epsilon, M), \quad T \leftarrow H(K_H, \text{enc}(N, K_E, A)).$$

Including  $K_E$  in the commitment input is essential in the direct-key formulation. If the tag depended only on  $(K_H, N, A)$ , an adversary could keep  $K_H$  fixed, choose a different AE key  $K'_E$ , and use decryption under  $K'_E$  to construct a second encryption context producing the same ciphertext and tag.

*Relation to AEAH and AEAH-KS.* The construction is based on AEAH [17] and closely related to the key-separated AEAH-KS transform of Minematsu and Inoue [42]. It is, however, not literally a special case of AEAH-KS. In AEAH-KS, a master key  $K$  is expanded as  $(K_E, K_H) \leftarrow G(K)$ , and the tag is computed as  $H(K_H, N, A)$ . Since  $K_E$  is not included in the input of  $H$ , their CMT-4 proof requires right-half collision resistance of  $G$ .

In contrast, AEAH-2K takes  $K_E \| K_H$  directly and computes the tag from  $(K_H, \text{enc}(N, K_E, A))$ . Consequently, a change to either component changes the complete key/input pair of  $H$ , and CMT-4 security reduces directly to the hCR security of  $H$ . Both AEAH-2K and AEAH-KS have standard-model proofs; unlike the original AEAH, neither relies on the random-oracle model.

The direct-key interface eliminates  $G$  as a component of the formal construction and removes the corresponding PRG and right-half collision-resistance terms from the bounds. This simplifies the construction and its assumptions, but does not necessarily reduce its online cost relative to an implementation of AEAH-KS that caches the expanded keys. Moreover, AEAH-2K processes the additional  $K_E$  bits through  $H$ , whereas AEAH-KS hashes only  $N$  and  $A$ .

**Theorem 7 (Privacy of AEaH-2K).** *For every adversary  $\mathcal{A}$ , there exist adversaries  $\mathcal{B}, \mathcal{C}$  running in about the same time as  $\mathcal{A}$  such that*

$$\text{Adv}_{\text{AEaH-2K}}^{\text{PRIV}}(\mathcal{A}) \leq \text{Adv}_H^{\text{nprf}}(\mathcal{B}) + \text{Adv}_{\text{AE}}^{\text{PRIV}}(\mathcal{C}).$$

*If  $\mathcal{A}$  is nonce-respecting, then so are  $\mathcal{B}$  and  $\mathcal{C}$ .*

*Proof intuition.* Since each commitment begins with a fresh nonce, nPRF security of  $H$  lets us replace the tags by independent uniform strings. The tags can be simulated without knowing  $K_E$ , allowing us to replace the ciphertext components by random strings of the same lengths. The full proof is given in Appendix B.2.

**Theorem 8 (Authenticity of AEaH-2K).** *Let  $\mathcal{A}$  make at most  $q_v$  decryption queries. Then there exist  $\mathcal{B}, \mathcal{C}$  with comparable running times such that*

$$\text{Adv}_{\text{AEaH-2K}}^{\text{auth}}(\mathcal{A}) \leq \text{Adv}_H^{\text{prf}}(\mathcal{B}) + \text{Adv}_{\text{AE}}^{\text{auth}}(\mathcal{C}) + \frac{q_v}{2^t}.$$

*If  $\mathcal{A}$  is nonce-respecting in its encryption queries, then so is  $\mathcal{C}$ .*

*Proof intuition.* By PRF security of  $H$ , we replace the commitment function by a random function with  $t$ -bit outputs. For a nonce/associated-data pair absent from encryption queries, a successful forgery requires guessing its hidden tag, contributing at most  $q_v/2^t$ , otherwise, injectivity of  $\text{enc}$  and nonce uniqueness imply that any successful non-replay changes the AE ciphertext under the same nonce, yielding an AE forgery. The full proof is given in Appendix B.3.

**Theorem 9 (CMT-4 security of AEaH-2K).** *For every adversary  $\mathcal{A}$ , there exists an adversary  $\mathcal{B}$  with comparable running time such that*

$$\text{Adv}_{\text{AEaH-2K}}^{\text{CMT-4}}(\mathcal{A}) \leq \text{Adv}_H^{\text{hcr}}(\mathcal{B}).$$

*Proof.* Let  $(K_E \| K_H, N, A, M) \neq (K'_E \| K'_H, N', A', M')$  be the output of  $\mathcal{A}$ :

*Case 1:*  $(K_E \| K_H, N, A) = (K'_E \| K'_H, N', A')$ . The distinctness of the complete tuples implies  $M \neq M'$ , while equality of the ciphertext components gives

$$\text{AE.Enc}(K_E, N, \epsilon, M) = C = \text{AE.Enc}(K_E, N, \epsilon, M').$$

This contradicts correctness of AE, since decryption of the common ciphertext under the same key and nonce cannot return two distinct messages.

*Case 2:*  $(K, N, A) \neq (K', N', A')$ . Equality of the tag components gives

$$\begin{aligned} H(K_H, \text{enc}(N, K_E, A)) &= H(K'_H, \text{enc}(N', K'_E, A')) \text{ and} \\ (K_H, \text{enc}(N, K_E, A)) &\neq (K'_H, \text{enc}(N', K'_E, A')). \end{aligned}$$

Otherwise, injectivity of enc would imply  $K_H = K'_H$ ,  $N = N'$ ,  $K_E = K'_E$ , and  $A = A'$ , contradicting the case assumption. The two pairs therefore form a valid hCR collision for  $H$ , which  $\mathcal{B}$  outputs.  $\square$

*Instantiation with FHASHN.* Setting  $H = \text{FHASHN}$ ,  $k_H = n$ ,  $t = 2n$ , and  $\text{enc}(N, K_E, A) = N \| K_E \| A$  gives

$$\begin{aligned} \text{Adv}_{\text{AEAH-2K}}^{\text{PRIV}}(\mathcal{A}) &\leq \text{Adv}_{\text{FHASHN}}^{\text{nprf}}(\mathcal{B}) + \text{Adv}_{\text{AE}}^{\text{PRIV}}(\mathcal{C}), \\ \text{Adv}_{\text{AEAH-2K}}^{\text{auth}}(\mathcal{A}) &\leq \text{Adv}_{\text{FHASHN}}^{\text{prf}}(\mathcal{B}) + \text{Adv}_{\text{AE}}^{\text{auth}}(\mathcal{C}) + \frac{q_v}{2^{2n}}, \\ \text{Adv}_{\text{AEAH-2K}}^{\text{CMT-4}}(\mathcal{A}) &\leq \text{Adv}_{\text{FHASHN}}^{\text{hcr}}(\mathcal{B}). \end{aligned}$$

The results of Section 4 further reduce these terms to the corresponding properties of the fixed-input-length cPRF and the underlying forkcipher. Instantiating FHASHN with FCPRF gives  $n$ -bit confidentiality (via Theorem 5) and approximately  $n/2$ -bit authenticity and CMT-4 security. Instantiating it with FIXM gives  $n$ -bit confidentiality (via Corollary 1), authenticity, and CMT-4 security, subject to the corresponding pseudorandomness and zmCR assumptions.

*Optional master-key derivation.* The preceding theorems treat  $K_E \| K_H$  as the scheme key. Alternatively, one may derive the pair from a master key  $K_m$  by setting  $(K_E, K_H) = G(K_m)$ . If  $K_m$  is regarded as the scheme key in the security experiments, the privacy and authenticity bounds each acquire an additional PRG-security term for  $G$ . The CMT-4 bound becomes

$$\text{Adv}_{\text{AEAH-2K}^G}^{\text{CMT-4}}(\mathcal{A}) \leq \text{Adv}_G^{\text{cr}}(\mathcal{D}) + \text{Adv}_H^{\text{hcr}}(\mathcal{B}).$$

where CR denotes collision resistance of the complete output of  $G$ . Full-output collision resistance, rather than only right-half collision resistance as in AEAH-KS, suffices because  $K_E$  is included in the commitment input. Pseudorandomness of  $G$  alone does not establish CMT-4 security of this master-key variant.

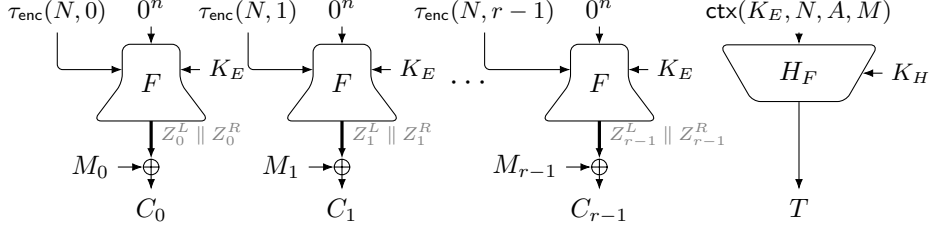


Fig. 6: The forkcipher-based FCTR-CMT.

## 5.2 A Forkcipher-Based Committing-AE Mode

We next propose a dedicated AEAD mode FCTR-CMT that uses only the forkcipher as an underlying primitive. It processes the two forkcipher branches as: 1. the encryption uses a forkcipher-based counter mode that processes message blocks of  $2n$  bits and 2. authenticates in parallel the complete encryption context with FHASHN using also forkcipher calls. We develop the mode with a single precomputable key derivation.

Let  $F$  be a forkcipher and  $H_F : \{0, 1\}^n \times \mathcal{D} \rightarrow \{0, 1\}^{2n}$  be FHASHN[FCPRF $_F$ ] or FHASHN[FXM $_F$ ]. Fix an  $n$ -bit constant  $\Delta_{\text{tag}}$ . Let  $\tau_{\text{kd}} = 1 \parallel 0^{2n-1}$  be the key-derivation tweak and  $\tau_{\text{enc}}(N, j) := 0 \parallel N \parallel \langle j \rangle_{n-1}$  the disjoint encryption tweak ( $0 \leq j < 2^{n-1}$ ), each  $2n$ -bit long. For an  $n$ -bit master key  $K$ , derive the 2 keys  $K_E \parallel K_H$  and the input to  $H_F$  as follows:

$$\begin{aligned} \text{KD}_F(K) &= F(K, \tau_{\text{kd}}, 0^n) = K_E \parallel K_H, \\ \text{ctx}(K_E, N, A, M) &= N \parallel \Delta_{\text{tag}} \parallel K_E \parallel \langle |A| \rangle_n \parallel \langle |M| \rangle_n \parallel A \parallel M. \end{aligned}$$

The inputs need to satisfy  $|A| < 2^n$ ,  $|M| < 2^n$ ,  $5n + |A| + |M| < 2^n$ . The final condition makes the input admissible for FHASHN. The encoding is injective: its fixed-length prefix determines  $N$ ,  $K_E$ ,  $|A|$ , and  $|M|$ , after which the two variable-length strings are parsed uniquely. Moreover, its first  $n$  bits are exactly  $N$ , as required by the nPRF analysis of FHASHN.

Let  $\text{trunc}_s(Z)$  denote the first  $s$  bits of  $Z$ . A message is processed in successive chunks  $M_j$  of at most  $2n$  bits. The last chunk may be partial. For  $M = \epsilon$ , there are no chunks and the encrypted-message part is also  $\epsilon$ .

The key-derivation call can be precomputed once per master key. Given the input lengths, encryption is one pass: the tag computation consumes  $A$  and each plaintext chunk while the same chunk is encrypted. Decryption can also compute the plaintext and tag in one pass.

*Pseudorandomness of the key derivation.* For an adversary  $\mathcal{A}$ , define

$$\text{Adv}_{\text{KD}_F}^{\text{prg}}(\mathcal{A}) := \left| \Pr_{K \leftarrow \mathcal{S}\{0,1\}^n} [\mathcal{A}(F(K, \tau_{\text{kd}}, 0^n)) \Rightarrow 1] - \Pr_{R \leftarrow \mathcal{S}\{0,1\}^{2n}} [\mathcal{A}(R) \Rightarrow 1] \right|.$$

**Lemma 4 (Security of the one-call key derivation).** *For every adversary  $\mathcal{A}$ , there exists  $\mathcal{B}$  making one external forward-forkcipher query such that*

$$\text{Adv}_{\text{KD}_F}^{\text{prg}}(\mathcal{A}) \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}).$$

*Proof.* The adversary  $\mathcal{B}$  queries its forkcipher oracle at  $(\tau_{\text{kd}}, 0^n)$  and gives the resulting  $2n$ -bit string to  $\mathcal{A}$ . In the real game this string is  $\text{KD}_F(K)$ . In the ideal game it is  $\pi_{\tau_{\text{kd}},0}(0^n) \parallel \pi_{\tau_{\text{kd}},1}(0^n)$ , which is uniform in  $\{0,1\}^{2n}$  because the two branch permutations are independent.  $\square$

**Theorem 10 (Privacy of FCTR-CMT).** *Let  $\mathcal{A}$  be a nonce-respecting privacy adversary making  $q_e$  encryption queries containing in total  $\sigma_E$   $2n$ -bit message blocks. Then there exist adversaries  $\mathcal{B}_{\text{kd}}$ ,  $\mathcal{B}_E$ , and  $\mathcal{B}_H$  such that*

$$\text{Adv}_{\text{FCTR-CMT}}^{\text{PRIV}}(\mathcal{A}) \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}_{\text{kd}}) + \text{Adv}_{H_F}^{\text{nprf}}(\mathcal{B}_H) + \text{Adv}_F^{\text{prtfp}}(\mathcal{B}_E).$$

Here  $\mathcal{B}_{\text{kd}}$  makes one external query,  $\mathcal{B}_E$  makes  $\sigma_E$  external forward queries, and  $\mathcal{B}_H$  makes  $q_e$  queries.

*Proof intuition.* Forkcipher pseudorandomness lets us replace the derived keys by two independent uniform keys. Since each hash input begins with a fresh nonce, nPRF security of  $H_F$  then replaces the tags by independent uniform strings. Finally, the forkcipher pseudorandomness replaces the encryption outputs at distinct nonce/counter tweaks by independent uniform pads, achieving privacy as a sum of the three distinguishing advantages. The full proof is given in Appendix B.4.

**Theorem 11 (Authenticity of FCTR-CMT).** *Let  $\mathcal{A}$  make at most  $q_v$  decryption queries. Then there are adversaries  $\mathcal{B}_{\text{kd}}$  and  $\mathcal{B}_H$  such that*

$$\text{Adv}_{\text{FCTR-CMT}}^{\text{auth}}(\mathcal{A}) \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}_{\text{kd}}) + \text{Adv}_{H_F}^{\text{prf}}(\mathcal{B}_H) + \frac{q_v}{2^{2n}}.$$

*Proof intuition.* Forkcipher pseudorandomness and PRF security of  $H_F$  let us replace the derived keys by independent uniform keys and the tag function by a random function. A verification context absent from encryption queries requires guessing a hidden  $2n$ -bit tag, contributing with at most  $q_v/2^{2n}$ . Else, injectivity of  $\text{ctx}$  and deterministic encryption make any successful query an exact replay, yielding the stated bound. The full proof is given in Appendix B.5.

**Theorem 12 (CMT-4 security of FCTR-CMT).** *For every CMT-4 adversary  $\mathcal{A}$ , there are adversaries  $\mathcal{B}_{\text{kd}}$  and  $\mathcal{B}_H$  such that*

$$\text{Adv}_{\text{FCTR-CMT}}^{\text{CMT-4}}(\mathcal{A}) \leq \text{Adv}_F^{\text{zmcr}}(\mathcal{B}_{\text{kd}}) + \text{Adv}_{H_F}^{\text{hcr}}(\mathcal{B}_H).$$

*Proof.* Suppose  $\mathcal{A}$  outputs contexts  $(K, N, A, M) \neq (K', N', A', M')$  with equal ciphertexts. Derive  $K_E \parallel K_H = F(K, \tau_{\text{kd}}, 0^n)$ ,  $K'_E \parallel K'_H = F(K', \tau_{\text{kd}}, 0^n)$ .

First assume that  $K \neq K'$  and  $(K_E, K_H) = (K'_E, K'_H)$ . Then  $F(K, \tau_{\text{kd}}, 0^n) = F(K', \tau_{\text{kd}}, 0^n)$ , which is a zmCR collision.  $\mathcal{B}_{\text{kd}}$  outputs the two contexts.

In every other case, let  $X = \text{ctx}(K_E, N, A, M)$ ,  $X' = \text{ctx}(K'_E, N', A', M')$ . We claim that  $(K_H, X) \neq (K'_H, X')$ . Otherwise  $K_H = K'_H$  and, by injectivity of  $\text{ctx}$ ,  $(K_E, N, A, M) = (K'_E, N', A', M')$ . If  $K = K'$ , this contradicts the distinctness of the two contexts. If  $K \neq K'$ , it gives equality of the two derived key pairs, which is precisely the case we excluded above. Because the tag is the final, fixed-length  $2n$  bits of the external ciphertext, equality of the two external ciphertexts implies  $H_F(K_H, X) = H_F(K'_H, X')$ . Together with  $(K_H, X) \neq (K'_H, X')$ , this is a valid hCR collision, which  $\mathcal{B}_H$  outputs. The two cases prove the claimed bound. The argument also covers empty messages and partial final chunks.  $\square$

Combining Th. 12 with the FHASHN, FCPRF, and FIXM reductions gives:

$$\begin{aligned} \text{Adv}_{\text{FCTR-CMT}[F, H_F^{\text{fCR}}]}^{\text{CMT-4}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{zmcr}}(\mathcal{B}_0) + \text{Adv}_F^{\text{fcr}}(\mathcal{B}_1), \\ \text{Adv}_{\text{FCTR-CMT}[F, H_F^{\text{zmCR}}]}^{\text{CMT-4}}(\mathcal{A}) &\leq \text{Adv}_F^{\text{zmcr}}(\mathcal{C}_0) + \text{Adv}_F^{\text{zmcr}}(\mathcal{C}_1). \end{aligned}$$

For privacy, the  $\text{nPRF}$  term  $\text{Adv}_{H_F}^{\text{nprf}}(\mathcal{B}_H)$  in Theorem 10 is instantiated differently depending on the choice of  $H_F$ . When  $H_F = \text{FHASHN}[\text{FCPRF}_F]$  ( $d = 2n$ ), Theorem 5 bounds this term directly by  $\text{Adv}_F^{\text{prtfp}}(\mathcal{B})$ . When  $H_F = \text{FHASHN}[\text{FIXM}_F]$  ( $d = n$ ), Corollary 1 applies instead, giving

$$\text{Adv}_{H_F}^{\text{nprf}}(\mathcal{B}_H) \leq \text{Adv}_F^{\text{zmcr}}(\mathcal{B}'_0) + \text{Adv}_F^{\text{prtfp}}(\mathcal{B}'_1)$$

with no additional birthday-type loss. In both cases  $\text{Adv}_{\text{FCTR-CMT}}^{\text{PRIV}}(\mathcal{A})$  term reduces entirely to the  $\text{prtfp}$  ( $\text{zmCR}$  for  $\text{FIXM}$  instantiation) assumptions on  $F$ .

For authenticity, Theorem 6 reduces the  $H_F$  PRF term in Theorem 11 to the hCR and PRF properties of the selected fixed-length construction, for either choice of  $H_F$ ; combined with Theorem 2 and Theorem 3, this reduces further to  $\text{prtfp}/\text{fCR}$  (for  $\text{FCPRF}$ ) or  $\text{prtfp}/\text{zmCR}$  (for  $\text{FIXM}$ ) assumptions on  $F$ .

Consequently, all construction-level security claims are in the standard model and reduce to the explicit pseudorandomness, fCR, and zmCR assumptions.

## 6 Cryptanalysis of ForkSkinny-128-384 Collisions

Since our proofs rely on forkcipher collision resistance and zero-message collision resistance (fCR and zmCR, Figure 7a and 7b) of forkciphers, we analyze the security of ForkSkinny-128-384 as a concrete instance under these notions. Due to Lemma 2 and 3, ForkSkinny-128-384 can achieve at most 64-bits of collision resistance and 128 bits of fixed-message collision resistance.

While many cryptanalysis techniques, like integral, linear, and algebraic cryptanalysis, primarily find distinguishers for key-recovery attacks, differential cryptanalysis lends itself as a convenient and successful technique for collision attacks. In particular, a differential characteristic with a non-zero input difference and a zero output difference, is a natural way to analyze collisions.

In related work, SKINNY-based hash functions have been analyzed in related work using differential cryptanalysis. The sponge-based Skinny-Hash [7]

has been analyzed [51,54]. SKINNY in Matyas-Meyer-Oseas hashing mode has been analyzed with rebound attacks and SAT solvers [36,26,54]. Romulus-H, i.e., SKINNY-128-384 in Hirose double block length hashing mode [31], has been analyzed with SAT solvers [44].

In Section 6.1, we present our differential cryptanalysis ForkSkinny-128-384 for finding collisions. In Section 6.2, we adapt the method to analyze the fixed-message collision resistance of ForkSkinny-128-384. Furthermore, in Appendix F and G, we present additional analysis of ButterKnife and ZIP-AES, respectively.

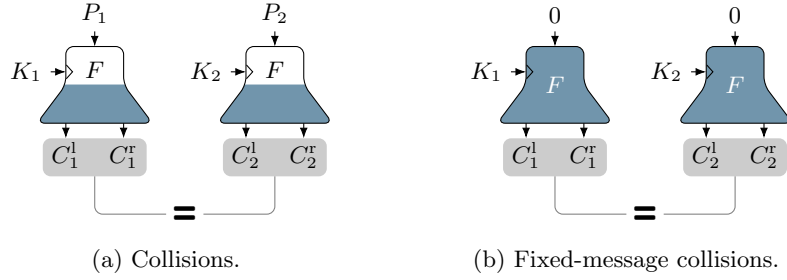


Fig. 7: Attack setup for finding (fixed-message) collisions. The blue color highlights the parts we need to analyze to find collisions.

### 6.1 Collisions for ForkSkinny-128-384

When looking for collisions without restrictions on the message input, we observe that the rounds before the forking point do not contribute to the security of the construction. In particular, for any values  $X, X'$  at the forking point that collide with tweakeys  $TK, TK'$ , we can find colliding messages  $M, M'$  via decryption.

To find such a colliding pair, there are three main techniques to consider: rebound attacks [41], guess-and-determine attacks [39,40], and MILP/SAT-based methods [37,44]. Rebound attacks are most competitive for ciphers with high diffusion, they are less suitable for SKINNY-based ciphers with comparatively slow diffusion. Guess-and-determine attacks require developing dedicated solvers tailored to the analyzed cipher. Further, both rebound attacks and guess-and-determine attacks typically fix a bitwise characteristic before satisfying values are obtained, which in the case of ForkSkinny requires special attention due to the dependencies at the branching point. To avoid these limitations, we use MILP and SAT modeling in a 2-step approach [37,44]. First, we use MILP to find cellwise differential characteristics with zero output difference. In this step we optimize for a minimal number of active S-boxes, allowing us to state lower bound on the number of active S-boxes. Second, we use SAT to simultaneously model the bitwise characteristic and an assignment of values following the characteristic. A solution for this SAT problem is a colliding pair  $X, X', TK, TK'$ .

In the design of ForkSkinny, the branch constant  $BC$  is XORED to the right branch after branching. This prevents the cancellation of the S-boxes in the reconstruction scenario [5]. With the branch constant the operations  $\mathcal{S}(S^{-1}(x))$  are replaced by  $\mathcal{S}(c_i \oplus S^{-1}(x))$  for 16 different  $c_i$ . We observe that these constants are chosen in a suboptimally, as the resulting effective S-boxes are cryptographically weaker than the SKINNY S-box. Their maximum differential probabilities (MDP) are distributed as follows: 10 have the weakest possible MDP of 1; 5 have MDP  $3/4$ ; 1 has MDP  $1/2$ . We show the difference distribution tables of these effective S-boxes in Figure 13 in Appendix E.3. Furthermore, all effective S-boxes have linearity 256 as they all contain linear component functions. Therefore, we do not count these S-boxes as active for optimizing and bounding probabilities.

*Description of the MILP Model.* Our MILP model is based on the model from [44]. We create binary variables  $X_i^l[r, c]/X_i^r[r, c]$  to indicate activity of the S-box row  $r$  column  $c$  in the  $i$ -th round after the forking point in the left/right branch. Similarly, we use  $Y_i^l[r, c]/Y_i^r[r, c]$  to denote activity after the tweakkey addition. We create binary variables  $TK_i[r, c]$  to denote the activity of the tweakkey cells; we use one list of tweakkeys for both branches. We constrain the tweakkey variables according to the tweakkey schedule and allow for up to 2 cancellation per tweakkey lane every 15 updates, i.e., 30 rounds. Similarly, we constrain the variables  $X, Y$  according to the tweakkey addition and MixColumns. We use Gurobi to optimize for a minimal number of active S-boxes not counting the S-boxes immediately after the forking point, i.e., we minimize  $\sum_{r>0} X_i^l[r, c] + X_i^r[r, c]$ .

*Description of the SAT Model.* Since in ForkSkinny many bitwise characteristics are invalid, i.e., they permit no valid assignment, we use a SAT model to simultaneously find a bitwise characteristic and a corresponding assignment like in [37]. This way a solution to the SAT model also directly yields a collision. To create this model, we formulate the set  $\{x, y, \Delta x, \Delta y : y = \mathcal{S}(x), \Delta y = \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \Delta x)\}$  in conjunctive normal form (CNF). We use the Espresso logic minimizer to find a CNF with 192 clauses, which we repeat for each active S-box in the cellwise characteristic. For inactive S-boxes, we create CNF that tracks  $y = \mathcal{S}(x)$ . While we find some collisions with these 2 models, we propose heuristics to find collisions for more rounds.

*Heuristic Improvements to the MILP Model.* Optimizing strictly for number of active S-boxes is useful to define a bound on the probability; however, we find that necessary cancellations in the tweakkey schedule for optimal solutions, severely impact the satisfiability of cellwise characteristics. While active S-Boxes restrict the possible values for each transition, tweakkey differences cancellations reduce the solution space by  $2^8$  each. Experiments highlight this trade-off as our models yield the best practical results for the optimization goal of  $\min 0.7 \cdot C_{TK} + 0.3 \cdot \sum_i S_i$ , where  $S_i$  denotes the activity of S-boxes and  $C_{TK}$  denotes the total count of tweakkey cancellations. Furthermore, we use the callback feature of Gurobi to filter out impossible cellwise characteristics: For each intermediate solution, we use a SAT solver to verify if a bitwise characteristic exists for the

cellwise propagation. We only accept the intermediate solution if the SAT solver returns SAT and reject it otherwise. To limit the runtime, we terminate the SAT solver after a timeout, which we initially set to 5 seconds and increase it after every 10th rejected solutions to account for longer solving times later in the optimization process. With the basic MILP model, we find collisions for up to  $37 = 25 + 6 + 6$  rounds, which we improve to  $47 = 25 + 11 + 11$  rounds.

*Results.* The basic MILP models leads to bounds on the number of active S-boxes for differential characteristics with zero output difference. Together with the maximum differential probability of  $2^{-2}$  of the SKINNY S-box, we bound the probability of single characteristics in Table 1. With the improved MILP model, we search for concrete collisions (see Table 2). We show an exemplary cellwise characteristic in Figure 14 in Appendix E.4. Note that our successful collision attacks use more active S-boxes than stated by the bound as cellwise characteristics with a minimal number of active S-boxes are not necessarily optimal for finding collisions. Furthermore, we observe that the SAT model consistently uses S-box transitions with lower probability than the ‘optimal’  $2^{-2}$ . This is because there are only few transitions with optimal probability and the SAT model does not consider the probability of the transitions. With the differential, MILP, and SAT models considered here, we found no attack on the full-round construction below the  $2^{64}$  generic fCR security.

Table 1: Bounds for collision attacks on ForkSkinny-128-384.  $\#\mathcal{S}$ : minimum number of active S-boxes.  $w_{\text{bnd}} = -\log_2(p)$ : associated probability bound.

| $r_0 = r_1$      | 10 | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  | 24  | 25  | 26  | 27  |
|------------------|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $\#\mathcal{S}$  | 46 | 52  | 62  | 70  | 80  | 88  | 77  | 87  | 96  | 98  | 105 | 117 | 118 | 130 | 136 | 144 | 152 | 160 |
| $w_{\text{bnd}}$ | 92 | 104 | 124 | 140 | 160 | 176 | 154 | 174 | 192 | 196 | 210 | 234 | 236 | 260 | 272 | 288 | 304 | 320 |

Table 2: Experimental results for ForkSkinny-128-384 collisions.  $\#\mathcal{S}$ : number of active S-boxes in characteristic.  $p_{\text{char}}$ : probability of the characteristic used for collision. Italics indicates average values. Time: time of SAT model on a single CPU core. \*: only for modified round constants (using  $RC_0$  after forking point).

| Rounds       | $\#\mathcal{S}$ | $p_{\text{char}}$ | $\#\text{Vars}$ | $\#\text{Clauses}$ | Time          | Result     |
|--------------|-----------------|-------------------|-----------------|--------------------|---------------|------------|
| (25, 12, 12) | 74              | $2^{-362}$        | 18 528          | 123 576            | > 40 days     | Timeout    |
| (25, 11, 11) | 65              | $2^{-389}$        | 17 536          | 112 666            | $\approx 1$ h | Collision* |
| (25, 10, 10) | 55              | $2^{-270}$        | 15 776          | 103 446            | $\approx 3$ s | Collision  |
| (25, 9, 9)   | 48              | $2^{-323}$        | 14 400          | 93 836             | $\approx 3$ s | Collision  |
| (25, 8, 8)   | 32              | $2^{-232}$        | 13 024          | 82 276             | < 1 s         | Collision  |
| (25, 7, 7)   | 28              | $2^{-212}$        | 11 648          | 72 796             | < 1 s         | Collision  |

*Satisfying values.* We note the success of the attack, specifically the configuration that covers 47 rounds, depends on the used round constants: for this particular configuration, we only find satisfying values for a modified round constants, where the round constant intended for the first round is used right after the forking point. In Table 4 in Appendix E.4, we present the satisfying values of the two most interesting configurations, covering 47 and 45 rounds respectively.

## 6.2 Fixed-Message Collisions for ForkSkinny-128-384

So far, we have seen bounds and concrete collisions when the message can be freely chosen. When the message is instead fixed to a constant like 0, attacks are much harder: the generic fixed-message collision attack from Lemma 3 takes about  $2^{128}$  time. Now, we adapt our analysis to the fixed-message case.

The fixed message requires a zero difference at the input. Therefore, we need to also model the rounds before the forking point and constrain a zero difference at the input. We apply this change to both our MILP model.

*Branch Constant.* For the S-boxes at the forking point, we need to track 3 differences now. For values  $x$ ,  $x \oplus \Delta x$  before the fork, we get differences  $\Delta y_1 = \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \Delta x)$  and  $\Delta y_0 = \mathcal{S}(x \oplus c) \oplus \mathcal{S}(x \oplus c \oplus \Delta x)$ , where  $c$  denotes a byte of  $BC$ . Since  $\Delta y_0$  and  $\Delta y_1$  are closely related as we saw earlier, we count a nonzero  $(\Delta x, \Delta y_0, \Delta y_1)$  as one active S-box for optimizing and bounding probabilities.

*Results.* In Table 3 (full version in Table 5 in Appendix E.4), we list lower bounds on the probability of single differential characteristics with zero output difference. While the bounds for (19, 19, 19) rounds do not exclude the possibility of a rebound attack, we observe a consistently higher number of active S-Boxes compared to the free-message case. Since the message is fixed, only 384 degrees of freedom remain, leaving significantly fewer possible satisfying values. The large remaining margin to the specified (25, 31, 31) rounds together with the gap between bounds and practical attacks in the free-message case, makes the existence of a practical attack for the specified rounds highly unlikely. We note that our bounds only directly bound attacks based on single differential characteristics. Still, based the factors outline above, we believe full-round attacks with complexity below  $2^{128}$  are not possible.

Table 3: Probability bounds on differential characteristics for fixed-message collision attacks on ForkSkinny-128-384 stated as  $-\log_2(p)$ .  $r_i$ : number of rounds before the forking point.  $r_0 = r_1$ : number of rounds after the forking point.

| $r_0 = r_1$ | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $r_i = 17$  | 166 | 182 | 196 | 202 | 202 | 212 | 220 | 236 | 240 | 266 | 276 | 292 | 296 | 332 | 342 | 356 |
| $r_i = 18$  | 170 | 190 | 202 | 200 | 206 | 220 | 224 | 246 | 250 | 270 | 280 | 302 | 308 | 332 | 352 | 360 |
| $r_i = 19$  | 180 | 194 | 204 | 200 | 208 | 222 | 232 | 256 | 260 | 276 | 294 | 312 | 310 | 342 | 354 | 366 |

## 7 Conclusion

We introduced the forkcipher collision-resistance notions  $\text{fCR}$  and  $\text{zmCR}$  and established matching classical generic query bounds in a unified external ideal-forkcipher model. We then cryptanalyzed **ForkSkinny-128-384** under both notions and our results support the security of the primitive.

We used these notions to construct fixed-length collision-resistant PRFs and the variable-input-length construction **FHASHN**. Instantiating the commitment component of our direct-two-key **AEAH-2K** transform with **FHASHN** yields a standard model context-committing AE scheme. In contrast to **AEAH-KS**, **AEAH-2K** includes the AE key in the commitment input and thus avoids a separate right-half collision-resistance assumption on a key-expansion function when the expanded key is used directly.

We additionally introduced **FCTR-CMT**, a dedicated forkcipher-based AEAD mode whose encryption path processes message chunks of up to  $2n$  bits using the expanding forkcipher output and whose commitment path applies **FHASHN** to the complete encryption context. We established nonce-respecting privacy, authenticity, and CMT-4 security for these constructions through black-box reductions to explicit pseudorandomness,  $\text{fCR}$ , and  $\text{zmCR}$  assumptions, without relying on a random oracle or ideal cipher.

Developing compact and independently analyzable assumptions for authenticated encryption modes is an important direction for future work, as is identifying other efficient primitives that realize the collision-resistant PRF interface.

*Acknowledgments.* This research was funded in part by the Austrian Science Fund (FWF) SFB project SPyCoDe (10.55776/F85) and the European Research Council (ERC) starting grant #101165216. Elena Andreeva was funded in part by the Vienna Science and Technology Fund (WWTF) CrossPings, grant number 10.47379/ICT25075.

## References

1. Abdalla, M., Bellare, M., Neven, G.: Robust encryption. In: TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer (2010). [https://doi.org/10.1007/978-3-642-11799-2\\_28](https://doi.org/10.1007/978-3-642-11799-2_28)
2. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmiege, S.: How to abuse and fix authenticated encryption without key commitment. In: 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10–12, 2022. pp. 3291–3308. USENIX Association (2022), <https://www.usenix.org/conference/usenixsecurity22/presentation/albertini>
3. Andreeva, E., Bhati, A.S., Preneel, B., Vizár, D.: 1, 2, 3, fork: Counter mode variants based on a generalized forkcipher. IACR Transactions on Symmetric Cryptology **2021**(3), 1–35 (2021). <https://doi.org/10.46586/tosc.v2021.i3.1-35>
4. Andreeva, E., Cogliati, B., Lallemand, V., Minier, M., Purnal, A., Roy, A.: Masked iterate-fork-iterate: A new design paradigm for tweakable expanding pseudorandom function. In: ACNS 2024. LNCS, vol. 14584, pp. 433–459. Springer (2024). [https://doi.org/10.1007/978-3-031-54773-7\\_17](https://doi.org/10.1007/978-3-031-54773-7_17)

5. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: Forkcipher: A new primitive for authenticated encryption of very short messages. In: ASIACRYPT 2019. LNCS, vol. 11922, pp. 153–182. Springer (2019). [https://doi.org/10.1007/978-3-030-34621-8\\_6](https://doi.org/10.1007/978-3-030-34621-8_6)
6. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer (2016). [https://doi.org/10.1007/978-3-662-53008-5\\_5](https://doi.org/10.1007/978-3-662-53008-5_5)
7. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: SKINNY-AEAD and skinny-hash. IACR Transactions on Symmetric Cryptology **2020**(S1), 88–131 (2020). <https://doi.org/10.13154/tosc.v2020.is1.88-131>
8. Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. In: EUROCRYPT 2022. LNCS, vol. 13276, pp. 845–875. Springer (2022). [https://doi.org/10.1007/978-3-031-07085-3\\_29](https://doi.org/10.1007/978-3-031-07085-3_29)
9. Bellare, M., Hoang, V.T.: Succinctly-committing authenticated encryption. In: CRYPTO 2024. LNCS, vol. 14923, pp. 305–339. Springer (2024). [https://doi.org/10.1007/978-3-031-68385-5\\_10](https://doi.org/10.1007/978-3-031-68385-5_10)
10. Bellare, M., Krovetz, T., Rogaway, P.: Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. In: EUROCRYPT ’98. LNCS, vol. 1403, pp. 266–280. Springer (1998). <https://doi.org/10.1007/bfb0054132>
11. Berti, F., Standaert, F.X., Levi, I.: Authenticity in the presence of leakage using a forkcipher. IACR Communications in Cryptology **1**(4), 13 (2024). <https://doi.org/10.62056/abksr-10k>
12. Bhati, A.S., Andreeva, E., Müller, S., Vizár, D.: Sonikku: Gotta speed, keed! A family of fast and secure macs. In: CANS 2025. LNCS, vol. 16351, pp. 120–141. Springer (2025). [https://doi.org/10.1007/978-981-95-4434-9\\_6](https://doi.org/10.1007/978-981-95-4434-9_6)
13. Bhati, A.S., Pohle, E., Abidin, A., Andreeva, E., Preneel, B.: Let’s go eevee! A friendly and suitable family of AEAD modes for iot-to-cloud secure computation. In: CCS 2023. pp. 2546–2560. ACM (2023). <https://doi.org/10.1145/3576915.3623091>
14. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer (1993). <https://doi.org/10.1007/978-1-4613-9314-6>
15. Campbell, P.: GLEVIAN and VIGORNIAN: Robust beyond-birthday AEAD modes. Cryptology ePrint Archive, Paper 2023/1379 (2023), <https://eprint.iacr.org/2023/1379>
16. Chan, J., Rogaway, P.: On committing authenticated-encryption. In: ESORICS 2022. LNCS, vol. 13555, pp. 275–294. Springer (2022). [https://doi.org/10.1007/978-3-031-17146-8\\_14](https://doi.org/10.1007/978-3-031-17146-8_14)
17. Chen, S., Karadžić, V.: Committing authenticated encryption: Generic transforms with hash functions. In: Advances in Cryptology – EUROCRYPT 2025. Lecture Notes in Computer Science, vol. 15601, pp. 93–123. Springer (2025). [https://doi.org/10.1007/978-3-031-91107-1\\_4](https://doi.org/10.1007/978-3-031-91107-1_4)
18. Chen, S., Guo, J., List, E., Shi, D., Zhang, T.: Scrutinizing the security of AES-based hashing and one-way functions. In: ASIACRYPT 2025. LNCS, vol. 16245, pp. 157–188. Springer (2025). [https://doi.org/10.1007/978-981-95-5018-0\\_6](https://doi.org/10.1007/978-981-95-5018-0_6)
19. Chen, Y.L., Davidson, M., Dworkin, M., Kelsey, J., Sasaki, Y., Turan, M.S., Thompson, A., Mouha, N., Chang, D.: Requirements for cryptographic accordions. Tech. Rep. NIST Interagency Report (IR) 8552 (2025). <https://doi.org/https://doi.org/10.6028/nist.ir.8552>

20. Chou, T., Orlandi, C.: The simplest protocol for oblivious transfer. In: LATIN-CRYPT 2015. LNCS, vol. 9230, pp. 40–58. Springer (2015). [https://doi.org/10.1007/978-3-319-22174-8\\_3](https://doi.org/10.1007/978-3-319-22174-8_3)
21. Daemen, J., Hoffert, S., Mella, S., Assche, G.V., Keer, R.V.: Shaking up authenticated encryption. In: EuroS&P 2025. pp. 861–882. IEEE (2025). <https://doi.org/10.1109/eurosp63326.2025.00054>
22. Daemen, J., Rijmen, V.: Plateau characteristics. IET Information Security **1**(1), 11–17 (2007). <https://doi.org/10.1049/iet-ifs:20060099>
23. Datta, N., Dutta, A., List, E., Mandal, S.: FEDT: forkcipher-based leakage-resilient beyond-birthday-secure AE. IACR Communications in Cryptology **1**(2), 21 (2024). <https://doi.org/10.62056/akgy186bm>
24. Dobraunig, C., Eichlseder, M., Mendel, F.: Analysis of SHA-512/224 and SHA-512/256. In: ASIACRYPT 2015. LNCS, vol. 9453, pp. 612–630. Springer (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_25](https://doi.org/10.1007/978-3-662-48800-3_25)
25. Dodis, Y., Grubbs, P., Ristenpart, T., Woodage, J.: Fast message franking: From invisible salamanders to encryption. In: CRYPTO 2018. LNCS, vol. 10991, pp. 155–186. Springer (2018). [https://doi.org/10.1007/978-3-319-96884-1\\_6](https://doi.org/10.1007/978-3-319-96884-1_6)
26. Dong, X., Guo, J., Li, S., Pham, P.: Triangulating rebound attack on AES-like hashing. In: CRYPTO 2022. LNCS, vol. 13507, pp. 94–124. Springer (2022). [https://doi.org/10.1007/978-3-031-15802-5\\_4](https://doi.org/10.1007/978-3-031-15802-5_4)
27. Farshim, P., Orlandi, C., Rosie, R.: Security of symmetric primitives under incorrect usage of keys. IACR Transactions on Symmetric Cryptology **2017**(1), 449–473 (2017). <https://doi.org/10.13154/tosc.v2017.i1.449-473>
28. Flórez-Gutiérrez, A., Grassi, L., Leander, G., Sibleyras, F., Todo, Y.: General practical cryptanalysis of the sum of round-reduced block ciphers and ZIP-AES. In: ASIACRYPT 2024. LNCS, vol. 15492, pp. 280–311. Springer (2024). [https://doi.org/10.1007/978-981-96-0947-5\\_10](https://doi.org/10.1007/978-981-96-0947-5_10)
29. Grubbs, P., Lu, J., Ristenpart, T.: Message franking via committing authenticated encryption. In: CRYPTO 2017. LNCS, vol. 10403, pp. 66–97. Springer (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_3](https://doi.org/10.1007/978-3-319-63697-9_3)
30. Hoang, V.T., Menda, S.: Robust AE with committing security. In: ASIACRYPT 2024. LNCS, vol. 15492, pp. 312–342. Springer (2024). [https://doi.org/10.1007/978-981-96-0947-5\\_11](https://doi.org/10.1007/978-981-96-0947-5_11)
31. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. IACR Transactions on Symmetric Cryptology **2020**(1), 43–120 (2020). <https://doi.org/10.13154/tosc.v2020.i1.43-120>
32. Kang, J., Hong, S., Chang, D., Yung, M.: HOP-1 and HOP-2: new re-keying schemes for symmetric ciphers. IACR Communications in Cryptology **2**(4), 28 (2026). <https://doi.org/10.62056/av18ksuc2>
33. Krämer, J., Struck, P., Weishäupl, M.: Committing AE from sponges security analysis of the NIST LWC finalists. IACR Transactions on Symmetric Cryptology **2024**(4), 191–248 (2024). <https://doi.org/10.46586/tosc.v2024.i4.191-248>
34. Lambæk, M.: Breaking and fixing private set intersection protocols. IACR Cryptology ePrint Archive, Paper 2016/665 (2016), <http://eprint.iacr.org/2016/665>
35. Len, J., Grubbs, P., Ristenpart, T.: Partitioning oracle attacks. In: 30th USENIX Security Symposium, USENIX Security 2021, August 11–13, 2021. pp. 195–212. USENIX Association (2021), <https://www.usenix.org/conference/usenixsecurity21/presentation/len>

36. Li, S., Liu, G., Pham, P.: Rebound attacks on skinny hashing with automatic tools. In: NSS 2022. LNCS, vol. 13787, pp. 649–666. Springer (2022). [https://doi.org/10.1007/978-3-031-23020-2\\_37](https://doi.org/10.1007/978-3-031-23020-2_37)
37. Liu, F., Isobe, T., Meier, W.: Automatic verification of differential characteristics: Application to reduced gimli. In: CRYPTO 2020. LNCS, vol. 12172, pp. 219–248. Springer (2020). [https://doi.org/10.1007/978-3-030-56877-1\\_8](https://doi.org/10.1007/978-3-030-56877-1_8)
38. Menda, S., Len, J., Grubbs, P., Ristenpart, T.: Context discovery and commitment attacks – how to break CCM, EAX, siv, and more. In: EUROCRYPT 2023. LNCS, vol. 14007, pp. 379–407. Springer (2023). [https://doi.org/10.1007/978-3-031-30634-1\\_13](https://doi.org/10.1007/978-3-031-30634-1_13)
39. Mendel, F., Nad, T., Schl  ffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer (2011). [https://doi.org/10.1007/978-3-642-25385-0\\_16](https://doi.org/10.1007/978-3-642-25385-0_16)
40. Mendel, F., Nad, T., Schl  ffer, M.: Improving local collisions: New attacks on reduced SHA-256. In: EUROCRYPT 2013. LNCS, vol. 7881, pp. 262–278. Springer (2013). [https://doi.org/10.1007/978-3-642-38348-9\\_16](https://doi.org/10.1007/978-3-642-38348-9_16)
41. Mendel, F., Rechberger, C., Schl  ffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Gr  stl. In: FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer (2009). [https://doi.org/10.1007/978-3-642-03317-9\\_16](https://doi.org/10.1007/978-3-642-03317-9_16)
42. Minematsu, K., Inoue, A.: Do not mix models: Revisiting generic transforms for committing authenticated encryption. IACR Cryptology ePrint Archive, Paper 2026/240 (2026), <https://eprint.iacr.org/2026/240>
43. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer (2011). [https://doi.org/10.1007/978-3-642-34704-7\\_5](https://doi.org/10.1007/978-3-642-34704-7_5)
44. Nageler, M., Pallua, F., Eichlseder, M.: Finding collisions for round-reduced Romulus-h. IACR Transactions on Symmetric Cryptology **2023**(1), 67–88 (2023). <https://doi.org/10.46586/tosc.v2023.i1.67-88>
45. Naito, Y., Sasaki, Y., Sugawara, T.: Committing security of Ascon: Cryptanalysis on primitive and proof on mode. IACR Transactions on Symmetric Cryptology **2023**(4), 420–451 (2023). <https://doi.org/10.46586/tosc.v2023.i4.420-451>
46. Naito, Y., Sasaki, Y., Sugawara, T.: Committing wide encryption mode with minimum ciphertext expansion. IACR Transactions on Symmetric Cryptology **2025**(1), 44–69 (2025). <https://doi.org/10.46586/tosc.v2025.i1.44-69>
47. Ni, J., Li, Y., Liu, F., Wang, G.: Practical key collision on AES and kiasu-bc. IEEE Transactions on Information Theory **71**(12), 9732–9752 (2025). <https://doi.org/10.1109/tit.2025.3623726>
48. Rahman, M., Kundu, S., Isobe, T.: On the key-commitment properties of forkcipher-based aeads. IACR Communications in Cryptology **1**(4), 32 (2024). <https://doi.org/10.62056/ayfhp2fgx>
49. Rogaway, P.: Formalizing human ignorance. pp. 211–228 (2006)
50. Sakamoto, K., Ito, R., Isobe, T.: Parallel SAT framework to find clustering of differential characteristics and its applications. In: SAC 2023. LNCS, vol. 14201, pp. 409–428. Springer (2023). [https://doi.org/10.1007/978-3-031-53368-6\\_20](https://doi.org/10.1007/978-3-031-53368-6_20)
51. Song, X., Jiang, W., Li, Z., Liu, L., Wu, S.: New collision paths for round-reduced SKINNY-hash. China Communications **17**(6), 145–152 (2020). <https://doi.org/10.23919/jcc.2020.06.012>
52. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: CRYPTO 2017. LNCS, vol. 10401, pp. 570–596. Springer (2017). [https://doi.org/10.1007/978-3-319-63688-7\\_19](https://doi.org/10.1007/978-3-319-63688-7_19)

53. Stevens, M., Karpman, P., Peyrin, T.: Freestart collision for full SHA-1. In: EUROCRYPT 2016. LNCS, vol. 9665, pp. 459–483. Springer (2016). [https://doi.org/10.1007/978-3-662-49890-3\\_18](https://doi.org/10.1007/978-3-662-49890-3_18)
54. Toyama, K., Sakamoto, K., Ito, R., Taka, K., Taiyama, K., Isobe, T.: Automated tool for finding practical collisions on SPN-based hash functions. In: ACISP 2026. LNCS, vol. 16792, pp. 3–39. Springer (2026). [https://doi.org/10.1007/978-981-92-3012-9\\_1](https://doi.org/10.1007/978-981-92-3012-9_1)
55. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer (2005). [https://doi.org/10.1007/11426639\\_1](https://doi.org/10.1007/11426639_1)
56. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer (2005). [https://doi.org/10.1007/11535218\\_2](https://doi.org/10.1007/11535218_2)
57. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer (2005). [https://doi.org/10.1007/11426639\\_2](https://doi.org/10.1007/11426639_2)

## A Internal Permutation Access to the Ideal Forkcipher

We consider the ideal IFI forkcipher from Definition 3 under the stronger internal interface. In this setting, the adversary may directly query  $P_U^j$  and  $(P_U^j)^{-1}$ , for  $j \in \{0, 1, 2\}$ , for adversarially chosen contexts  $U \in \mathcal{U}$ . We count each forward or inverse evaluation of an internal permutation as one query, including the evaluations needed to verify the adversary’s output.

**Lemma 5 (fCR security with internal permutation access).** *Let  $\mathcal{A}$  make at most  $q$  internal-permutation queries to  $\text{IFI-FC}_n$ . If  $q \leq 2^{n-1}$ , then*

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{fcr}}(\mathcal{A}) \leq \frac{2q^2}{2^n}.$$

*Proof.* Let **bad** be the event that the output of a fresh internal permutation query equals an input or output value that appeared in an earlier query. Before the  $i$ -th query, at most  $2(i-1)$  such values have appeared. The output of the new query is uniform over at least  $2^n - q$  available values. Therefore,

$$\begin{aligned} \Pr[\text{bad}] &\leq \sum_{i=1}^q \frac{2(i-1)}{2^n - q} \\ &\leq \frac{2q^2}{2^n}. \end{aligned}$$

It remains to show that an fCR collision implies **bad**. Suppose that two distinct inputs produce the same output:  $\text{IFI-FC}_n(K, T, M) = \text{IFI-FC}_n(K', T', M') = C_0 \parallel C_1$ . Let  $U = (K, T)$  and  $U' = (K', T')$ , and let  $S = P_U^0(M)$  and  $S' = P_{U'}^0(M')$ . The collision requires all four relations

$$\begin{aligned} Q_{U,0}(S) &= C_0, & Q_{U,1}(S) &= C_1, \\ Q_{U',0}(S') &= C_0, & Q_{U',1}(S') &= C_1. \end{aligned}$$

Consider the last query needed to establish these four relations. If it is a forward query, its output is either  $C_0$  or  $C_1$ , which has already appeared in a previous query. If it is an inverse query, its output is either  $S$  or  $S'$ , which has already appeared in the query to the other branch of the same context. In either case, **bad** occurs.

Thus, an fCR collision cannot occur unless **bad** occurs, and hence

$$\text{Adv}_{\text{IFI-FC}_n}^{\text{fer}}(\mathcal{A}) \leq \Pr[\text{bad}] \leq \frac{2q^2}{2^n}.$$

□

## B Extended Proofs

### B.1 Proof of Theorem 2

We assume without loss of generality that  $\mathcal{A}$  makes no redundant queries. We first give the proof for FCPRF. Let  $G_0$  be the real PRF experiment for FCPRF. In  $G_1$ , replace the forkcipher by the ideal external forkcipher from Figure 2. A **prtfp** adversary  $\mathcal{B}$  runs  $\mathcal{A}$ , forwards every construction call to its forkcipher oracle, and returns  $\mathcal{A}$ 's output bit. Hence,

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}). \quad (1)$$

In  $G_1$ , a query  $(T, M)$  is answered by  $\pi_{T,0}(M) \parallel \pi_{T,1}(M)$ , where the two permutations are independent for every tweak  $T$ . We then couple this game with a random function returning two  $n$ -bit blocks. The two games agree until, for some two distinct queries using the same tweak, the random function repeats the first or the second output block. Let  $q_T$  denote the number of queries using tweak  $T$ . By the union bound,

$$\Pr[\text{bad}] \leq \frac{2}{2^n} \sum_{T \in \mathcal{T}} \binom{q_T}{2} \leq \frac{2 \binom{q}{2}}{2^n} = \frac{q(q-1)}{2^n}.$$

Together with (1), this proves

$$\text{Adv}_{\text{FCPRF}}^{\text{prf}}(\mathcal{A}) \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}) + \frac{q(q-1)}{2^n}.$$

For FIXM, every distinct construction query is a distinct tweak, and the ideal forkcipher is evaluated only once under each tweak. Consequently, its responses are distributed exactly as those of a random function.

The same argument applies to nonce-respecting FCPRF queries: the first  $n$  input bits belong to the tweak, so nonce uniqueness implies tweak uniqueness. Finally, nonce-respecting FIXM queries also use distinct tweaks. In all three cases, the ideal forkcipher and random function games are identical, leaving only the corresponding **prtfp** term.

## B.2 Proof of Theorem 7

We use a sequence of games and write  $G_i \Rightarrow 1$  for the event that  $\mathcal{A}$  returns 1 in game  $G_i$ .

$G_0$ . This is the real **priv** experiment for AEAH-2K, with  $K = K_E \| K_H \leftarrow \$\{0, 1\}^{k_{AE} + k_H}$ .

$G_1$ . Replace  $H(K_H, \text{enc}(N, K_E, A))$  by  $\rho(\text{enc}(N, K_E, A))$ , where  $\rho$  is a lazy-sampled random function from  $\mathcal{D}$  to  $\{0, 1\}^t$ . Since the first  $n$  bits of every encoded input are exactly  $N$ , the reduction  $\mathcal{B}$  is nonce-respecting. Hence,

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_H^{\text{prf}}(\mathcal{B}).$$

$G_2$ . Replace the output of  $\text{AE.Enc}(K_E, N, \epsilon, M)$  by a random string of the same length. This gives

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{\text{AE}}^{\text{PRIV}}(\mathcal{C}).$$

Although  $\mathcal{C}$  does not know  $K_E$ , it can simulate  $\rho$  using a table indexed by  $(N, A)$ . Indeed,  $K_E$  is fixed throughout the experiment, and injectivity of  $\text{enc}$  ensures that distinct pairs  $(N, A)$  give distinct encoded inputs for this fixed  $K_E$ .

In  $G_2$ , the AE outputs are random. Since nonces do not repeat, the inputs to  $\rho$  in encryption queries are distinct, so the tag outputs are also independent and uniform. Thus,  $G_2$  is identical to the ideal **priv** experiment.

## B.3 Proof of Theorem 8

Let  $G_0$  be the real authenticity experiment. In  $G_1$ , replace  $H(K_H, \text{enc}(N, K_E, A))$  by  $\rho(\text{enc}(N, K_E, A))$ , where  $\rho$  is a lazy-sampled random function with  $t$ -bit outputs. A PRF reduction gives

$$|\Pr[G_0] - \Pr[G_1]| \leq \text{Adv}_H^{\text{prf}}(\mathcal{B}).$$

Consider a decryption query  $(N, A, C \| T)$  in  $G_1$  and write  $X = \text{enc}(N, K_E, A)$ . If no encryption query previously evaluated  $\rho$  at  $X$ , then  $\rho(X)$  is hidden from  $\mathcal{A}$ . For  $q_v$  verification attempts, the probability of guessing it is at most  $q_v/2^t$ .

It remains to consider a successful query for which an encryption query previously evaluated  $\rho$  at the same  $X$ . Injectivity of  $\text{enc}$  implies equality of  $N$ ,  $K_E$ , and  $A$ . Since encryption queries are nonce-respecting, there is a unique previous encryption query with nonce  $N$ ; let its output be  $C_0 \| T$ . A successful forgery cannot be the replay  $C_0 \| T$ , so  $C \neq C_0$ . Nevertheless,  $\text{AE.Dec}(K_E, N, \epsilon, C) \neq \perp$ , and therefore  $(N, \epsilon, C)$  is a valid forgery against AE. Combining the three cases proves the theorem.

## B.4 Proof of Theorem 10

We use three games. Game  $G_0$  is the real privacy experiment. In  $G_1$ , replace  $(K_E, K_H) = \text{KD}_F(K)$  by two independent uniform  $n$ -bit keys. By Lemma 4,

$$|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_F^{\text{prtfp}}(\mathcal{B}_{\text{kd}}).$$

In  $G_2$ , replace  $H_F(K_H, \text{ctx}(K_E, N, A, M))$  by a lazy-sampled random function. The first  $n$  bits of every hash input equal  $N$ . Hence the nonce-respecting condition makes the resulting reduction  $\mathcal{B}_H$  nonce-respecting, and

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{H_F}^{\text{npf}}(\mathcal{B}_H).$$

In particular, every tag in  $G_2$  is a fresh uniform string.

In  $G_3$ , we replace the forkcipher under  $K_E$  by its ideal external interface. We have all tweaks  $\tau_{\text{enc}}(N, j)$  distinct and nonce uniqueness resulting in distinct encryption queries returning independent uniform  $2n$ -bit ciphertexts. Then,

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \text{Adv}_F^{\text{prfp}}(\mathcal{B}_E).$$

Since the tag is independent and uniform in  $G_2$ , then  $G_3$  returns an independent uniform string of length  $|M| + 2n$ . Summing the three differences proves our claim.

### B.5 Proof of Theorem 11

We first replace the derived pair  $(K_E, K_H)$  by two independent uniform keys, which is bound by at most  $\text{Adv}_F^{\text{prfp}}(\mathcal{B}_{\text{kd}})$  by Lemma 4. Next, we replace  $H_F(K_H, \cdot)$  by a lazy-sampled random function  $\rho$  deriving a bound by at most  $\text{Adv}_{H_F}^{\text{prf}}(\mathcal{B}_H)$ .

We then consider a decryption query  $(N, A, C||T)$ . The decryption algorithm first determines  $M$  from  $(K_E, N, C)$  and then  $X = \text{ctx}(K_E, N, A, M)$ . If  $X$  did not occur in an earlier encryption query,  $\rho(X)$  is unknown to  $\mathcal{A}$ . The probability of a successful guess over at most  $q_v$  verification attempts is at most  $q_v/2^{2n}$ .

It remains to examine the case in which  $X$  equals an earlier encryption query. The injectivity of  $\text{ctx}$  implies that the earlier query used the same  $(N, A, M)$  and produced the same  $C$ , and hence the correct tag is the same value  $T$ . The successful decryption query is therefore an exact replay and is not an authenticity forgery. These two cases prove the bound.

## C Discussion on Modelling Collision Resistance

The collision resistance property hCR is similar to hash function collision resistance and thus faces similar challenges. Many real-world hash functions, such as SHA-2, are unkeyed. As such, it is natural to also represent them as unkeyed functions in the security model. Unfortunately, no unkeyed hash function can formally achieve collision resistance. This limitation arises from how security is defined for arbitrary adversaries. Since hash functions go from a large input space to a small output space, collisions must exist. By considering arbitrary adversaries, we also capture adversaries that make no computations and only output a fixed collision. This formal modeling therefore disregards the practical difficulty of actually finding the collision. For this problem, two main solutions have been proposed.

- By the human ignorance approach by Rogaway, we just ignore the fact that no hash function can satisfy this unkeyed collision resistance notion. This keeps the theoretical representation of the hash function close to the real world implementation. The notion still allows writing proofs that reduce the security of a complex scheme to that of the underlying hash function.
- The alternative is to model hash functions as taking a secret key that is set at the start of the experiment. Because adversaries must be able to provide collisions for any key they receive, they cannot simply output a single colliding message pair. This overcomes the formal problem but leaves a significant gap between the real-world function (that is supposed to be secure in practice without a secret key) and the formal model. Protocols based on these hash functions then exhibit the same gap since they sample and potentially distribute the hash key. Real-world implementations are then supposed to ignore these hash keys and all related operations, which may be confusing to the practitioners.

Notably, while our  $H$  function is keyed, the distinction between key and message is only relevant for the PRF property. For the purpose of hCR, the key behaves like an additional message block and can be chosen by the adversary. As such, hCR also faces the aforementioned challenges as modeling collision resistance for (unkeyed) hash functions. We opted for an approach similar to the human ignorance approach by Rogaway. Our hCR notion cannot be satisfied by any fixed  $H$  function. However, it is perfectly suitable for reduction proofs and avoids the confusion of introducing a second key for hashing that just exists in the formal model.

## D Generic AE procedures

In Figure 8 we give the encryption and decryption algorithms of AEAH-2K. In Figure 9 we give the encryption and decryption of the dedicated AEAD mode FCTR-CMT.

| <b>Procedure</b> $\text{Enc}(K, N, A, M)$         | <b>Procedure</b> $\text{Dec}(K, N, A, Y)$          |
|---------------------------------------------------|----------------------------------------------------|
| $K_E \  K_H \leftarrow K$                         | <b>if</b> $ Y  < t$ <b>then return</b> $\perp$     |
| $C \leftarrow \text{AE.Enc}(K_E, N, \epsilon, M)$ | $C \  T \leftarrow Y \quad //  T  = t$             |
| $T \leftarrow H(K_H, \text{enc}(N, K_E, A))$      | $K_E \  K_H \leftarrow K$                          |
| <b>return</b> $C \  T$                            | $T' \leftarrow H(K_H, \text{enc}(N, K_E, A))$      |
|                                                   | <b>if</b> $T \neq T'$ <b>then return</b> $\perp$   |
|                                                   | <b>return</b> $\text{AE.Dec}(K_E, N, \epsilon, C)$ |

Fig. 8: Encryption and Decryption procedures of AEAH-2K construction shown in Figure 5.

| FCTR-CMT.Enc( $K, N, A, M$ )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | FCTR-CMT.Dec( $K, N, A, Y$ )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $K_E \parallel K_H \leftarrow F(K, \tau_{\text{kd}}, 0^n)$<br>$C \leftarrow \epsilon$<br>$r \leftarrow \lceil  M /2n \rceil$<br><b>for</b> $j = 0$ <b>to</b> $r - 1$ <b>do</b><br>$M_j \leftarrow j\text{-th at-most-}2n\text{-bit chunk of } M$<br>$Z_j \leftarrow F(K_E, \tau_{\text{enc}}(N, j), 0^n)$<br>$C_j \leftarrow M_j \oplus \text{trunc}_{ M_j }(Z_j)$<br>$C \leftarrow C \parallel C_j$<br>$X \leftarrow \text{ctx}(K_E, N, A, M)$<br>$T \leftarrow H_F(K_H, X)$<br><b>return</b> $C \parallel T$ | <b>if</b> $ Y  < 2n$ <b>then</b><br><b>return</b> $\perp$<br>$C \parallel T \leftarrow Y$<br>$K_E \parallel K_H \leftarrow F(K, \tau_{\text{kd}}, 0^n)$<br>$M \leftarrow \epsilon; \quad r \leftarrow \lceil  C /2n \rceil$<br><b>for</b> $j = 0$ <b>to</b> $r - 1$ <b>do</b><br>$C_j \leftarrow \text{chunk}_{2n}(C, j)$<br>$Z_j \leftarrow F(K_E, \tau_{\text{enc}}(N, j), 0^n)$<br>$M_j \leftarrow C_j \oplus \text{trunc}_{ C_j }(Z_j)$<br>$M \leftarrow M \parallel M_j$<br>$T' \leftarrow H_F(K_H, \text{ctx}(K_E, N, A, M))$<br><b>if</b> $T \neq T'$ <b>then</b><br><b>return</b> $\perp$<br><b>return</b> $M$ |

Fig. 9: Encryption and decryption of FCTR-CMT shown in Figure 6. If  $M = \epsilon$  or  $C = \epsilon$ , respectively, the corresponding loop is empty.

## E Detailed Analysis of ForkSkinny

In Appendix E.1, we give a description of the ForkSkinny primitive. In Appendix E.2, we outline the specification of the SKINNY round function. In Appendix E.3, we show the differential properties of the effective S-boxes in the ForkSkinny reconstruction scenario. In Appendix E.4, we list the full table for our bounds on single characteristics in the fixed-message case.

### E.1 Detailed description of ForkSkinny

The overall structure of ForkSkinny is shown in Figure 10. The round function  $\mathcal{R}$  of ForkSkinny is closely based on the one from SKINNY. Following the *iterate-fork-iterate* principle, ForkSkinny first processes  $M$  with  $r_{\text{init}} = 25$  iterations of  $\mathcal{R}$ . The fork-step, consists of copying the internal state and adding a branch constant  $BC$  to the state that leads to  $C_1$ . To both of the resulting branches ForkSkinny-128-384 applies  $r_0 = 31$  and  $r_1 = 31$  iterations of  $\mathcal{R}$  to produce the outputs  $C_0$  and  $C_1$ , respectively. The tweakkey  $TK$  is processed by the tweakkey update function  $TKS$  before being added to the state. We refer to the tweakkey after  $i$  applications of  $TKS$  as  $TK_i$ . For the initial rounds, the tweakkey gets initialized by  $TK_0 = K \parallel T$ . The tweakkey at the input of the first branch is  $TK_{r_{\text{init}}}$  and of the second branch  $TK_w$ , with  $w = r_{\text{init}} + r_0$ .

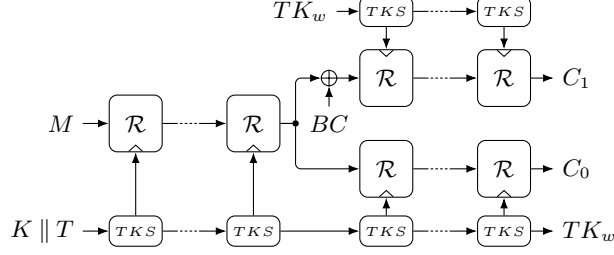


Fig. 10: ForkSkinny encryption.

*Round function.* ForkSkinny-128 views the state as a 4 by 4 matrix of 8-bit words. The round function  $\mathcal{R}$  applies the following operations to the state:

$$\mathcal{R} = \text{MixColumns} \circ \text{ShiftRows} \circ \text{AddRoundTweakey} \circ \text{AddConstant} \circ \text{SubCells}.$$

Compared to **Skinny**, all operations (see Section E.2) except that the round tweakeys and constants for the  $C_1$  branch are continued versions of those of the  $C_0$  branch. The constants used in **AddConstant** are sourced from a 7-bit LFSR, with the update function

$$(rc_6 \parallel rc_5 \parallel \dots \parallel rc_0) \rightarrow (rc_5 \parallel \dots \parallel rc_0 \parallel rc_6 \oplus rc_5 \oplus 1).$$

The LFSR is initialized with the 7 bits set to 0 and the update function is applied at each round. In each round  $i$ , a round constant state  $RC_i$ , defined as

$$RC_i = \begin{pmatrix} c_0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

is added to the state, where  $(c_0, c_1, c_2) = (0 \parallel 0 \parallel 0 \parallel 0 \parallel rc_3 \parallel rc_2 \parallel rc_1 \parallel rc_0, 0 \parallel 0 \parallel 0 \parallel 0 \parallel rc_6 \parallel rc_5 \parallel rc_4, 0x2)$ .

*Branch Constant.* At the beginning of the branch leading to  $C_1$ , ForkSkinny-128 adds the constant  $BC$  to the state with

$$BC = \begin{pmatrix} 01 & 02 & 04 & 08 \\ 10 & 20 & 41 & 82 \\ 05 & 0a & 14 & 28 \\ 51 & a2 & 44 & 88 \end{pmatrix}.$$

## E.2 Description of SKINNY-128-384

We illustrate the round function of SKINNY-128-384 in Figure 11.

The **SubCells** operations applies an 8-bit S-box (DDT shown in Figure 12a) to each cell of the state. After adding the round constant (see Section 2.3), the round

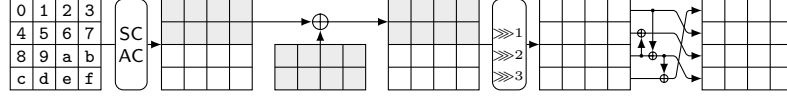


Fig. 11: One round of Skinny-128-384.

tweakey  $tk$  is added to the upper half of the state. The **ShiftRows** operation applies a word based permutation  $P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12]$ . Finally, we apply the **MixColumns** operation, which can be seen as a bit-sliced matrix multiplication with

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

The key  $k$  and the tweak  $t$  are partitioned as  $TK1||TK2||TK3 = k||t$ . After applying the tweakey update function  $TKS$ , we add the upper half of  $tk = TK1 \oplus TK2 \oplus TK3$  to the state. The  $TKS$  separately permutes  $TK1$ ,  $TK2$ , and  $TK3$  by the word based permutation  $PT = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ . Afterward, distinct LFSRs are applied to the first and second row of  $TK2$  and  $TK3$ . For further details, refer to the design document [6].

### E.3 Properties of Effective S-boxes in ForkSkinny-128

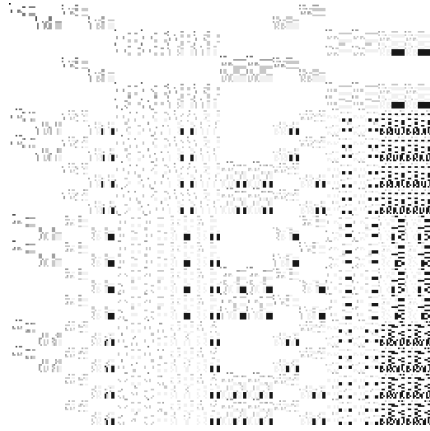
In Figure 13, we show the difference distribution tables of the effective S-boxes that arise due to ForkSkinny’s branch constant when reconstructing one ciphertext from the other. In Figure 12, we show the difference distribution tables of the SKINNY S-box and the AES S-box for comparison.

### E.4 Extra Tables and Figures for ForkSkinny-128-384 Collisions

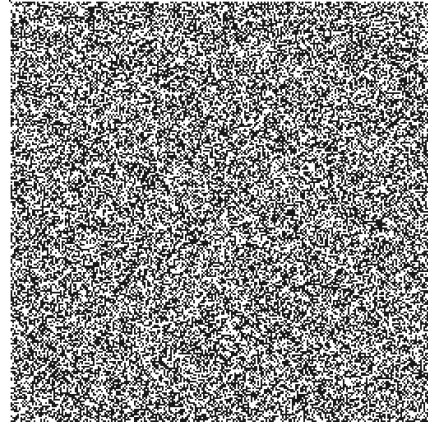
In Table 4, we display the satisfying values for our collision attacks on ForkSkinny-128-384. These results are for the case where we can arbitrarily choose the message. In Table 5, we list the full table of lower bounds on the probability of single differential characteristics with zero output difference. In Figure 14, we show the cellwise characteristic used for the collision attack on ForkSkinny-128-384 with (25, 11, 11) rounds.

Table 4: Satisfying values for collision attacks on ForkSkinny-128-384. \*: only for modified round constants, i.e., when starting the round constant schedule after the forking point.

| Rounds       | (25, 10, 10)                       | (25, 11, 11)*                    |
|--------------|------------------------------------|----------------------------------|
| $M$          | cfe090a908a0eb94ff461fc31e634b54   | —                                |
| $\Delta M$   | 9fd784ad19ea3ee9f46fa9d66ba0ed3c   | —                                |
| $F$          | 8a3ae0a9da685fa8a1f30afae22182ad   | 04169d836eba486b32530a5fdb0eface |
| $\Delta F$   | 00d27c59d20000a2f10000d2593d0059   | 00006d00006b00002000770091270000 |
| $TK1$        | 6831dc16325ebc1d284294420dd14cd7   | 134cc107fae146d3047a5b884ee56eb8 |
| $\Delta TK1$ | 00000000000000000000000000000000   | 00ed0000000000000000000000000000 |
| $TK2$        | 3f2b73924b05c5400bbf85c08653415f   | 4a910124b2b89d661bc0d1d445611277 |
| $\Delta TK2$ | 0000000000000000000000000000f0000  | 001c0000000000000000000000000000 |
| $TK3$        | 1a29b566109c1ae40fd334037b3ab28a   | 23e32035979f02527ea1cc292f019ebb |
| $\Delta TK3$ | 00000000000000000000000000007f0000 | 00b70000000000000000000000000000 |



(a) DDT of SKINNY S-box.



(b) DDT of AES S-box.

Fig. 12: Difference distribution tables of the SKINNY and AES S-boxes. White pixels indicate impossible transitions, while darker pixels indicate transitions of higher probability.

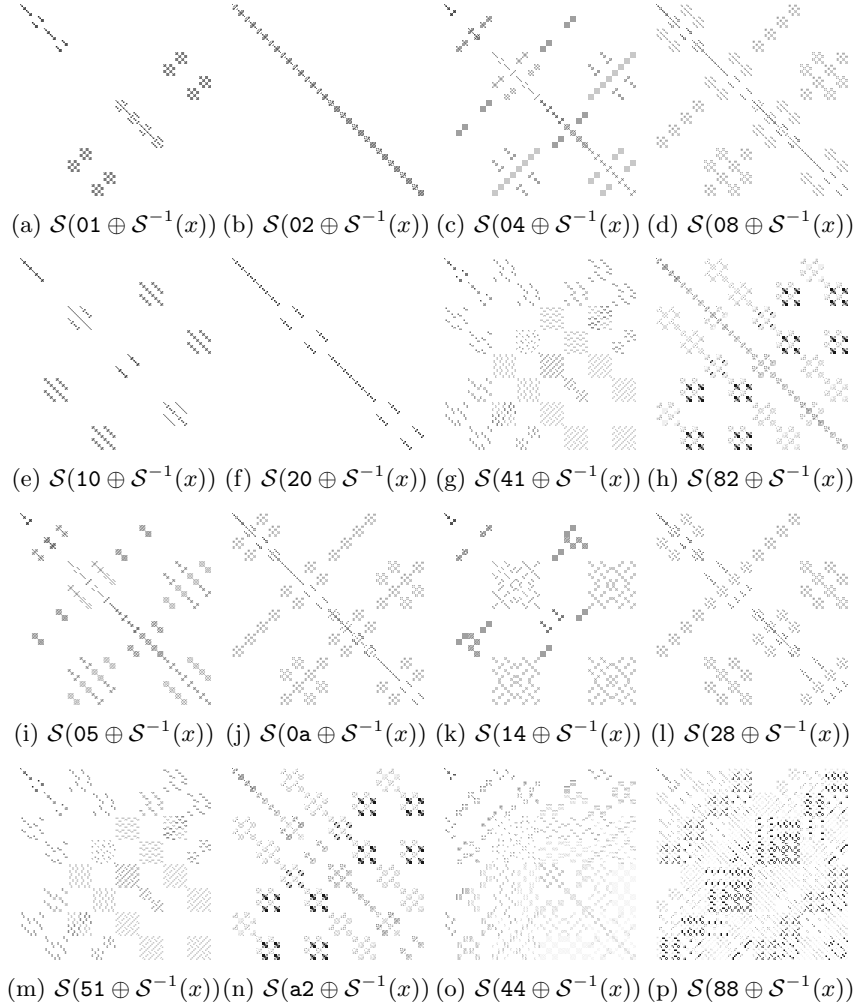


Fig. 13: Difference distribution tables for effective S-boxes due to branch constant  $BC$  in ForkSkinny reconstruction scenario. White pixels indicate impossible transitions, while darker pixels indicate transitions of higher probability.  $\mathcal{S}$  denotes the SKINNY S-box.

Table 5: Probability bounds on differential characteristics for fixed-message collision attacks on ForkSkinny-128-384 stated as  $-\log_2(p)$ . Rows state  $r_i$ , the number of rounds before the forking point. Columns state  $r_0 = r_1$ , the number of rounds after the forking point.

| $r_0 = r_1$ | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  |
|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| $r_i = 3$   | 44  | 64  | 84  | 94  | 106 | 130 | 144 | 162 | 176 | 196 | 206 | 198 | 202 | 222 | 238 | 244 |
| $r_i = 4$   | 56  | 72  | 90  | 108 | 114 | 136 | 150 | 172 | 186 | 206 | 210 | 202 | 208 | 230 | 240 | 254 |
| $r_i = 5$   | 64  | 82  | 96  | 110 | 126 | 144 | 158 | 178 | 192 | 206 | 204 | 212 | 218 | 236 | 246 | 256 |
| $r_i = 6$   | 78  | 84  | 108 | 122 | 136 | 154 | 166 | 190 | 202 | 186 | 206 | 220 | 220 | 238 | 254 | 268 |
| $r_i = 7$   | 80  | 90  | 114 | 132 | 136 | 158 | 178 | 198 | 202 | 198 | 216 | 220 | 226 | 248 | 264 | 274 |
| $r_i = 8$   | 84  | 108 | 122 | 134 | 152 | 168 | 186 | 202 | 206 | 194 | 220 | 230 | 230 | 256 | 274 | 286 |
| $r_i = 9$   | 96  | 114 | 128 | 144 | 158 | 180 | 194 | 204 | 202 | 204 | 226 | 232 | 236 | 260 | 280 | 290 |
| $r_i = 10$  | 108 | 122 | 136 | 156 | 164 | 184 | 206 | 204 | 206 | 210 | 230 | 242 | 246 | 268 | 290 | 296 |
| $r_i = 11$  | 114 | 128 | 140 | 162 | 170 | 192 | 204 | 202 | 212 | 214 | 234 | 248 | 252 | 280 | 298 | 310 |
| $r_i = 12$  | 118 | 132 | 158 | 170 | 184 | 202 | 202 | 204 | 206 | 224 | 242 | 258 | 260 | 278 | 306 | 320 |
| $r_i = 13$  | 188 | 144 | 162 | 176 | 190 | 202 | 204 | 214 | 220 | 234 | 252 | 264 | 264 | 294 | 312 | 316 |
| $r_i = 14$  | 136 | 154 | 168 | 182 | 206 | 204 | 202 | 216 | 228 | 242 | 262 | 274 | 276 | 298 | 318 | 328 |
| $r_i = 15$  | 152 | 160 | 178 | 196 | 194 | 200 | 210 | 224 | 230 | 248 | 264 | 284 | 278 | 312 | 330 | 338 |
| $r_i = 16$  | 154 | 166 | 182 | 204 | 198 | 204 | 218 | 228 | 242 | 256 | 272 | 284 | 298 | 318 | 332 | 348 |
| $r_i = 17$  | 166 | 182 | 196 | 202 | 202 | 212 | 220 | 236 | 240 | 266 | 276 | 292 | 296 | 332 | 342 | 356 |
| $r_i = 18$  | 170 | 190 | 202 | 200 | 206 | 220 | 224 | 246 | 250 | 270 | 280 | 302 | 308 | 332 | 352 | 360 |
| $r_i = 19$  | 180 | 194 | 204 | 200 | 208 | 222 | 232 | 256 | 260 | 276 | 294 | 312 | 310 | 342 | 354 | 366 |

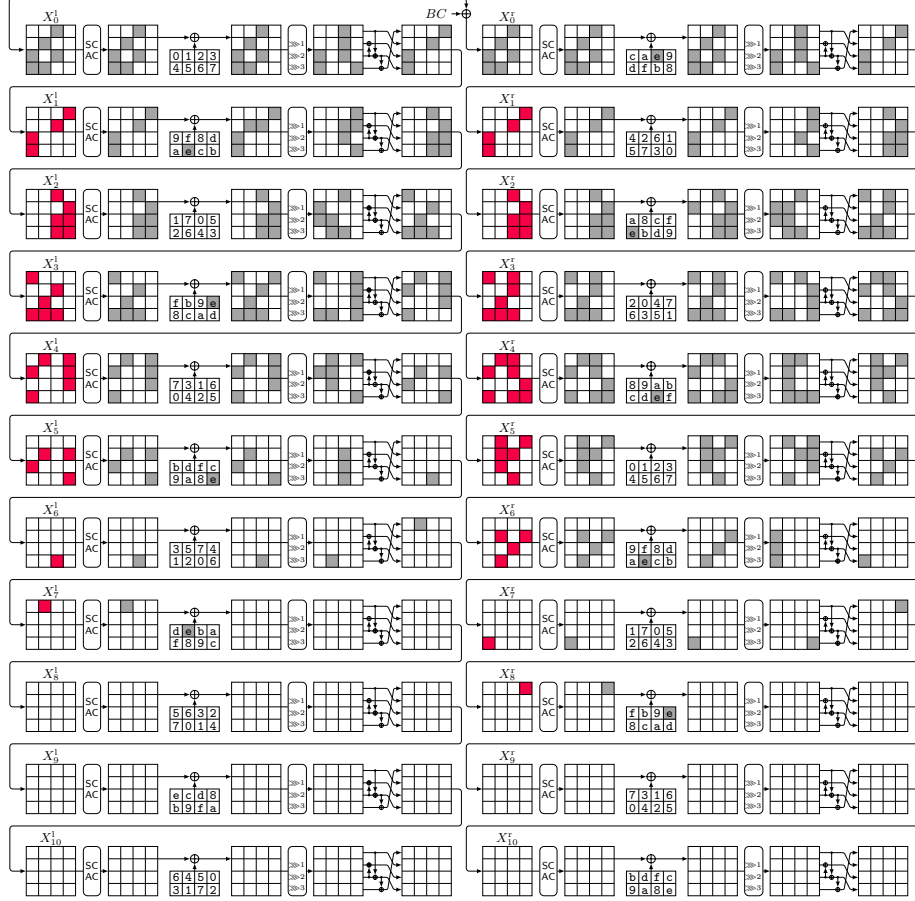


Fig. 14: Cellwise differential characteristic used for collision for ForkSkinny with (25, 11, 11) rounds with 65 active S-boxes.

## F Cryptanalysis of ButterKnife Collision Security

ButterKnife is a tweakable expanding pseudo-random function proposed by Andreeva et al. at ACNS 2024 [4]. ButterKnife’s construction employs a Davies-Meyer-like feed-forward mechanism, making it non-trivial to find collisions in the output. Truncating ButterKnife’s output to 2 blocks fits our interface for a fixed-length cPRF making it a candidate for constructing a context committing AE. However, based on the following cryptanalysis we conclude that we are not confident in the collision security of ButterKnife to propose this construction. Note that our analysis does not affect the classical security notions of ButterKnife.

### F.1 Brief Description of ButterKnife

ButterKnife is based on the tweakable block cipher Deoxys-BC-256 which in turn is based on the AES round function with an adapted key schedule that. Deoxys-BC-256 processes an 128-bit plaintext block and a 256-bit tweakkey using the AES round functions and a linear tweakkey schedule based on LFSRs and cell shuffles. ButterKnife first applies 7 Deoxys-BC rounds to the input, remembers the intermediate value at the forking point and creates up to 8 branches. Each branch applies another 8 Deoxys-BC rounds that use the same tweakkeys but differ in the used round constants. Finally, the remembered intermediate value is XORED onto each output block. We depict the structure in Figure 15. A full specification can be found in the design document [4].

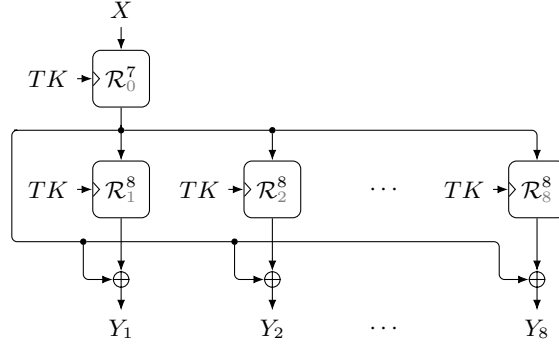


Fig. 15: Structure of ButterKnife producing up to 1024 bits of output from 128 bit input and 256 bit tweakkey.  $\mathcal{R}_c^r$  denotes  $r$  rounds of Deoxys-BC using branch constant  $c$ .

## F.2 Rebound Attacks on 1-Branch and 2-Branch ButterKnife

We present a practical rebound attack to find collisions for **ButterKnife** with 1 output branch. We use a MILP model find a cellwise differential characteristic, that we use to bound the total number of S-boxes in a differential attack. Independent, of the MILP model, we use a SAT model to find a bitwise characteristic that is suitable for finding collisions. Then, we utilize a SAT solver to find a bitwise characteristic. Finally, we implement the collision attack on full round 1-branch **ButterKnife** and on 2-branch **ButterKnife** reduced to 3 rounds (instead of 7) after the forking point.

*MILP Model.* We use a MILP model to bound the number of active S-boxes. We create variables for the input and output cells of the S-boxes, as well as the cells before tweakey addition. To model the tweakey schedule, we allow up to 1 cancellation in each tweakey lane. In contrast to **ForkSkinny**, 12 of 16 S-boxes after the forking point have identical values. The other 4 S-boxes have only differ due to the fixed branch constant. In both cases, we only count the S-boxes in the round after the forking point once.

*SAT Model.* We use a SAT model to find bitwise differential characteristics suitable for rebound attacks on **ButterKnife** when considering 1 or 2 output blocks. We encode the DDT using the **espresso** logic minimizer. For the DDT of the 4 related S-boxes in the round following the forking point, we encode a heuristically reduced version of the  $8 \rightarrow 16$  bit DDT. This DDT contains many entries with values 0, 2, 4 and the trivial entry with 256. To encode the DDT using **espresso**, we specify that all values  $\geq 4$  must be covered in the CNF, all entries with 0 must be absent, and the other entries with value 2 may be present or absent. This way, we get a relatively compact CNF with only 30 clauses.

*Results.* We display the bounds on the number of active S-boxes for differential characteristics for **ButterKnife** with 2 output blocks in Table 6. We show a concrete rebound attack for full-round **ButterKnife** with 1 output block in Figure 16. We can solve the inbound phase using Gaussian elimination. In particular, the solution set of the active S-boxes forms an affine space (as the AES S-box is planar [22]) and the tweakey schedule is linear. We implement this attack and are able to quickly find a concrete collision after repeating the inbound phase  $2^{14}$  times.

While the concrete attack does not extend to the 2-branch case, we believe collision attacks with below-generic attack complexity are possible. In particular, in Table 6, we can see that for the specified attacks, we need at least 28 active S-boxes for a differential characteristic that leads to a collision. While this corresponds to a probability bound of  $2^{-168}$ , many of these S-boxes can be solved in the inbound phase, as **ButterKnife** offers 384 bits of freedom (256 bit tweakey and 128 bit input). Hence, we do not claim security against collision for **ButterKnife**. Note that this analysis does not affect **ButterKnife**'s classical security.

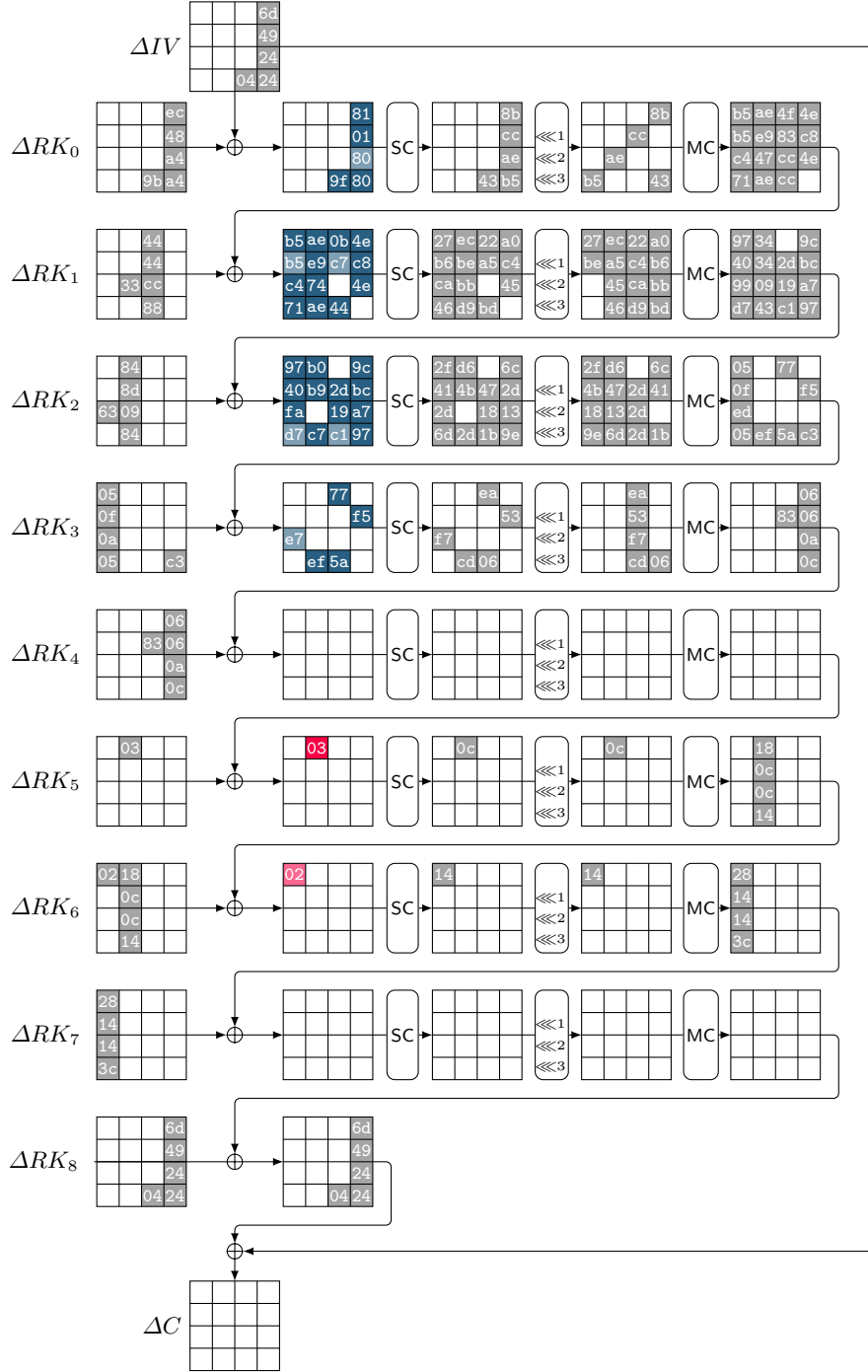


Fig. 16: Rebound attack on full-round ButterKnife with 1 output block with outbound phase probability  $2^{-13}$ . Inbound phase is solved by Gaussian elimination. ■/■: active S-box in inbound phase with  $p = 2^{-6} / 2^{-7}$ . ■/■: active S-box in outbound phase with  $p = 2^{-6} / 2^{-7}$ .

Table 6: Lower bounds on the number of active S-boxes in differential characteristics for **ButterKnife** with 2 output blocks.  $r$  denotes the number of rounds per branch after the forking point. Bold text indicates the specified number of rounds.

| $r$   | 4 | 5 | 6  | 7  | <b>8</b>  | 9  | 10 | 11 | 12 | 13 | 14 |
|-------|---|---|----|----|-----------|----|----|----|----|----|----|
| $\#S$ | 4 | 8 | 16 | 22 | <b>28</b> | 36 | 44 | 50 | 56 | 60 | 68 |

## G Cryptanalysis of ZIP-AES Collision Security

ZIP-AES is a pseudo-random function proposed by Flórez-Gutiérrez et al. [28], that instantiates the ZIP cipher construction with the AES-128 blockcipher. The input is processed in parallel by two branches and their output is XORED to produce the final output of the cipher. This mechanism ensures non-invertibility and prevents trivial collisions. While ZIP-AES does not directly fit our interface for a fixed-length cPRF, it nevertheless appears promising in providing collision resistance and may consequently serve as a candidate for constructing a context committing AE. However, based on the following cryptanalysis and existing collision results on similar constructions we conclude that we are not confident in the collision security of ZIP-AES.

### G.1 Brief Description of ZIP-AES

ZIP-AES is characterized by the sum-of-PRP paradigm, proposed by Bellare et al. [10] and utilized by the ZIP cipher construction [28]. In its essence, the sum-of-PRP paradigm is a way to turn PRPs into PRFs by making them non-invertible. For ZIP-AES specifically, the AES-128 primitive, which is specified with 10 rounds, is split into two halves  $AES_5$  and  $AES_5^{-1}$ , each of 5 rounds, with an additional MixColumns at the end of the final round of each half. While the AES-128 key schedule remains unchanged and generates the keys for both branches from the same initial key, the round keys are used in ascending order in both branches. Consequently, the inverted branch uses the round keys in reversed order. To accomodate for the fact that on each branch a whitening-key operation is performed at the beginning of the first round, an additional placeholder key  $0^{128}$  is added to the round keys. During the encryption, an input  $P$  is split, and both copies of the input are processed independently on the two branches to produce two intermediate outputs which are then XORED to produce the final output  $C = AES_5(P) \oplus AES_5^{-1}(P)$ . We illustrate the ZIP-AES encryption in Figure 17, where  $r_{\text{left}}$  and  $r_{\text{right}}$  denote the rounds of the inverted and forward branches, respectively.

### G.2 Rebound Attacks on ZIP-AES

Following the attack methodology presented in Section F.2 to ZIP-AES, we obtain a practical collision for 5 rounds. We construct a SAT model that encodes the

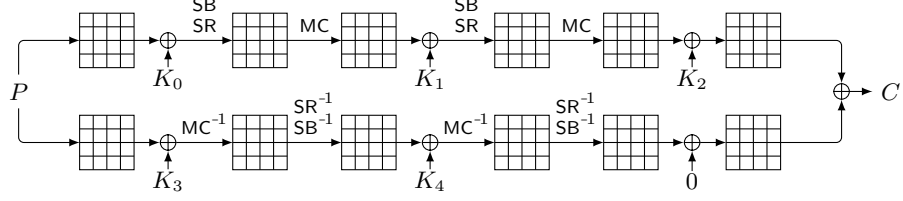


Fig. 17: Specification of ZIP-AES, illustrated for  $(r_{\text{left}}, r_{\text{right}}) = (2, 2)$ .

ZIP-AES primitive to find bitwise characteristics suitable for rebound attacks. We then use the obtained characteristics to perform a rebound attack on round-reduced ZIP-AES and obtain practical collisions.

*SAT Model.* To minimize the overhead of adjusting the attack for ZIP-AES, we construct the SAT model so that its resulting characteristics resemble those of the Davies-Meyer construction of the **ButterKnife** branches. We achieve this by selecting the right-branch output as the cipher’s input and modeling its counter-clockwise path through the cipher. This yields an equivalent representation of the ZIP-AES primitive, where the input  $P$  encodes the state connecting the two encryption blocks  $\text{AES}_5$  and  $\text{AES}_5^{-1}$ .

*Results.* We show the concrete rebound attack for 5-round ZIP-AES in Figure 18. While there is still a large margin to the specified rounds of ZIP-AES, the negligible attack complexity of  $2^6$  and existing theoretical results on similar constructions covering up to 7 rounds [18], indicate that practical collisions for more than 5 rounds may very much be possible. We believe that further analysis is required to make a claim about the collision resistance of ZIP-AES.

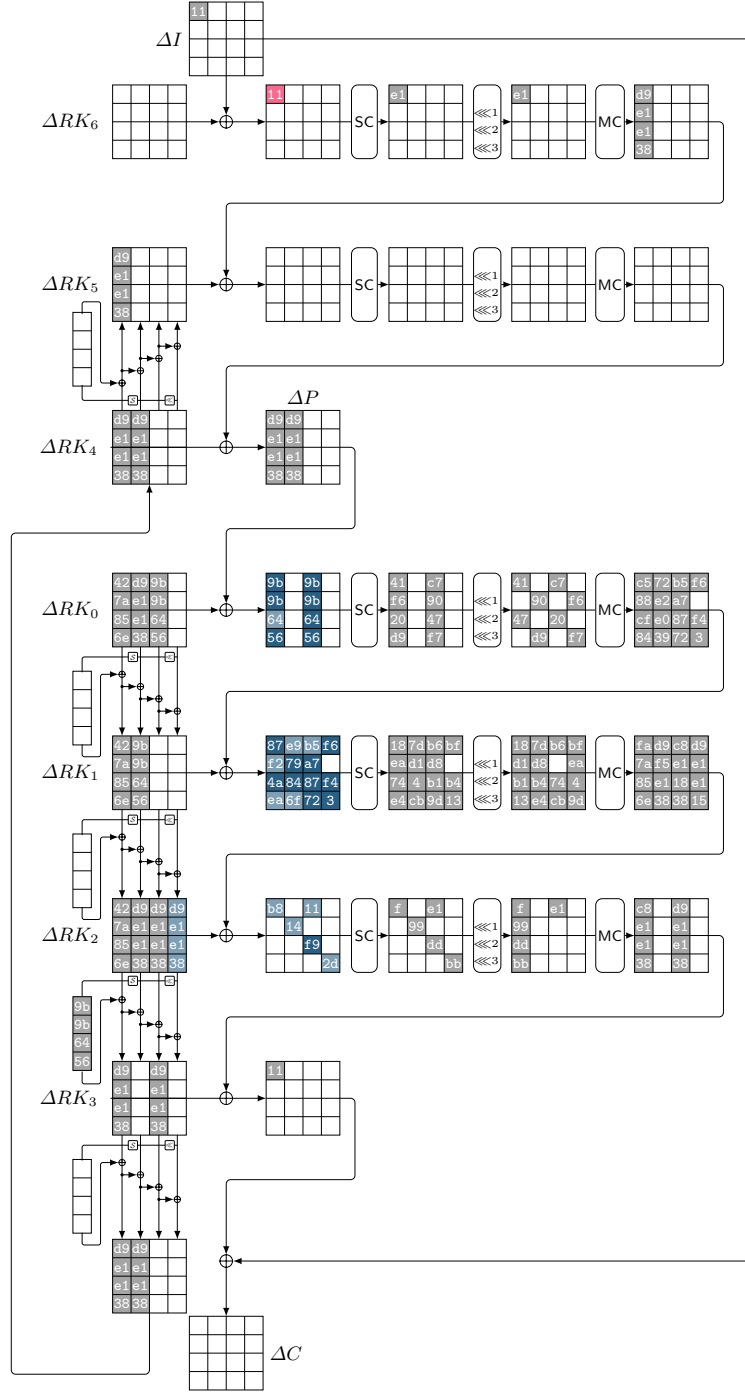


Fig. 18: Rebound attack on ZIP-AES-3-2 with outbound phase probability  $2^{-6}$ . Inbound phase is solved by Gaussian elimination. ■/■: S-box in inbound phase with  $p = 2^{-6} / 2^{-7}$ . ■/■: S-box in outbound phase with  $p = 2^{-6} / 2^{-7}$ .