Generic Committing Attacks

Zero-Padded Ascon is Less Secure than Expected

Nilanjan Datta[1], Hrithik Nandi[1,2], Soumit Pal[3], Yu Sasaki[4], Patrick Struck[5], and Maximiliane Weishäupl[6]

[1] Institute for Advancing Intelligence (IAI), TCG CREST, Kolkata, India
[2] Ramakrishna Mission Vivekananda Educational and Research Institute, India
[3] Indian Statistical Institute, Kolkata, India
[4] NTT Social Informatics Laboratories, Tokyo, Japan and NIST Associate, Gaithersburg, US
[5] University of Konstanz, Germany
[6] University of Regensburg, Germany
nilanjan.datta@tcgcrest.org, hrithiknandi.crypto@gmail.com, soumitpal378@gmail.com, yusk.sasaki@ntt.com, patrick.struck@uni.kn, maximiliane.weishaeupl@ur.de

Abstract. We study generic committing attacks—where ciphertexts decrypt under more than one context, i.e., key, nonce, associated data—for sponge-based authenticated encryption. As our main contribution, we give three new committing attacks which outperform existing attacks. One of our attacks provides a counterexample showing that the previous proof for the committing security of Ascon-like schemes with zero-padding does not extend to all parameter choices: in case of 128-bit tags and 256-bit zero-padding, the existing analysis claims 192-bit security; our attack reduces this by 62 bits down to 130-bit. Our attacks are applicable to the standardized scheme Ascon. As a further contribution, we analyze existing attack strategies for a generic sponge construction with various design features such as key-blinding, zero-padding, and state-update-functions.

1 Introduction

Authenticated encryption with associated data (AEAD) is a cryptographic primitive to enable secure communication between two parties, Alice and Bob, who share a secret key K. In order to securely send a message M to Bob, Alice runs the encryption algorithm which takes as input the key K, a nonce N, optional associated data A, and the message M, and outputs a ciphertext C along with a tag T. The nonce is used to randomize the encryption and is typically required to be unique for every encryption, while the associated data can be considered

* This work started when the authors participated at the GAPS 2025 workshop in Singapore. The authors acknowledge funding by the Cryptology Research Society of India (CRSI), the BMFTR under the project Quant-ID (16KISQ111), and the Hector Foundation II.

as a part of the message that does not need to be confidential, e.g., metadata. The triple (K, N, A) is often referred to as the *context* under which the message M is encrypted.

An AEAD scheme is deemed secure if it provides both confidentiality and integrity—nowadays, these two properties are often covered by a single security notion. Confidentiality ensures that an adversary that does not know the key K, cannot learn anything about the message M from the ciphertext C. Integrity guarantees that an adversary cannot produce a ciphertext C' (along with a nonce and associated data) such that decryption succeeds. These two security properties have in common that the adversary does not know the key shared between the communicating parties. Recent years have shown that certain use-cases of AEAD schemes require the schemes to also be *committing* [15]. Here, the intuitive question is whether ciphertexts only decrypt under a single key. More formally, committing security asks an adversary to find *a single* ciphertext C and *two* contexts $(K, N, A), (K', N', A')$ such that C decrypts validly under both contexts. To formally capture the intuitive description above, the keys from the two contexts are required to be distinct; a notion we will henceforth refer to as $\mathrm{CMT}_{\mathrm{K}}$. However, one can consider various variants of committing security, following a generic framework given by Menda et al. [19]. Throughout this work, we will be concerned with two specific notions of committing security. The first notion is CMT-4 [4], where the sole requirement is that the contexts differ $(K, N, A) \neq (K', N', A')$—unlike for $\mathrm{CMT}_{\mathrm{K}}$ security, where the keys have to be distinct. This notion is the strongest form of committing security and implies all others. Studying CMT-4 security comes with the benefit that schemes achieving it can be widely deployed as it excludes all variations of possible attacks. The second notion we consider is $\mathrm{CMT}^{\star}_{\mathrm{K}}$ [19]. This notion is similar to $\mathrm{CMT}_{\mathrm{K}}$ in that the keys have to be different $K \neq K'$, but at the same time the remaining contexts have to agree, i.e., $(N, A) = (N', A')$, making this a more restricted form of $\mathrm{CMT}_{\mathrm{K}}$. What makes $\mathrm{CMT}^{\star}_{\mathrm{K}}$ security particularly interesting is that it is a weaker notion—therefore potentially easier to provably achieve—that still prevents several real-world attacks like the fast message franking attack [13] and the partitioning oracle attack [17]. Furthermore, the overall importance of committing security as a standard design goal can also be seen from the fact that NIST makes committing security a requirement for applications of the accordion mode [10].

When looking at common design principles for AEAD schemes, sponge-based constructions, obtained from large permutations, are favorable for committing security. Krämer et al. [16] analyzed the finalists from the NIST lightweight cryptography standardization process: out of the 9 analyzed finalists, only 3 (Ascon [12], Isap [11], and Schwaemm [3][7]) achieve CMT-4, while the remaining finalists fall short—in many cases even w.r.t. $\mathrm{CMT}_{\mathrm{K}}$. The commonality of the 3 positive examples is that they are all sponge-based. Among the 6 others are also sponge-based constructions, but certain design choices led to commit-

[7] Note that the result for Schwaemm requires a minimal modification.

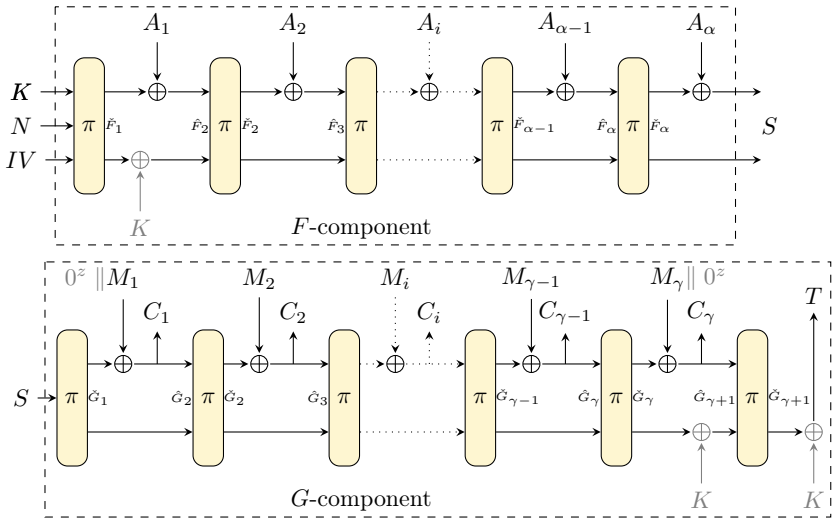

Fig. 1. Illustration of the plain duplex AEAD scheme. We denote the upper part, where the associated data is processed, as the F component, and the lower part, where the message is processed, as the G component. The different variants for key-blinding and zero-padding are depicted in gray.

ting attacks.[8] The winner of the standardization process, Ascon, was independently analyzed by Naito et al. [20]. In contrast to Krämer et al. [16], which only analyzes the mode of operation of Ascon, Naito et al. [20] also analyze the underlying permutation. For Ascon, both works show 64-bit of committing security, which stems from the fact that Ascon has a tag length of 128 bit, and showing committing security in particular requires colliding tags. As argued by Bellare and Hoang [5], 64-bit of committing security might be insufficient for practical use, as committing attacks are offline attacks. A non-invasive way to achieve more committing security is so-called zero-padding, which was also analyzed by Naito et al. [20]. Here, the message M is padded with z zeros before encryption, and decryption verifies that the z zeros are present in the decrypted message—if they are not, the ciphertext is deemed invalid and rejected. The advantage of zero-padding is that it does not require any changes to the AEAD scheme. While zero-padding was originally proposed as a method to turn non-committing AEAD schemes into schemes that achieve CMT_K security [1], Naito et al. [20] show that it enhances CMT-4 security by lifting it from $\frac{\tau}{2}$-bit CMT-4 security to $\frac{\tau+z}{2}$-bit CMT-4 security for Ascon-like schemes. This observation seems intuitive, as the additional zeros can be viewed as having a longer tag.

[8] We stress that committing security was introduced after the call-for-algorithms and was not a requirement by NIST. Hence, the schemes were not designed with any intention to achieve committing security.

1.1 Our Focus

In this work, we study and develop committing attack against sponge-based authenticated encryption schemes. The scheme we consider is illustrated in Fig. 1[9]. We represent the scheme in terms of two components: F and G. The former processes the associated data while the latter processes the message. This representation follows the one given in [16]. The scheme comes with some optional design choices that we consider. Three key-blindings, which refer to the key being XORed to the capacity part of the state:

1. Initial key-blinding (after the first permutation of F)
2. Final key-blinding (before the final permutation of G)
3. Tag key-blinding (after the final permutation of G)

We will typically refer to these key-blinding as Init/Fin/Tag KB, for sake of brevity. Furthermore, we consider the scheme with potential zero-padding, i.e., with z zeros prepended (prefix zero-padding) or appended (suffix zero-padding) to the message before encryption. The standardized AEAD scheme Ascon [24] is a concrete instantiation of this generic construction, where all three key-blindings are used; similarly, other real-world schemes, e.g., Photon-Beetle [2] can be viewed as instantiations. Besides the plain duplex, we also consider the generic construction with so-called state-update-functions. Here, the absorbing and squeezing part is done via some function ξ. The aforementioned example of Photon-Beetle actually deploys such a function.

For the AEAD scheme, we consider the following three generic committing attacks[10]:

– Initial state (IS) attack: start from an intermediate state and invert the sponge until two valid initial states are found.
– Capacity collision (CC) attack: generate two states that collide in their capacity, which can then be turned into a full-state collision by absorbing proper inputs.
– Tag collision (TC) attack: generate states until colliding tags are found (ensuring the same ciphertexts are trivial by choosing matching message blocks).

1.2 Our Contribution

In the following, we elaborate on our contributions. On the one hand, we give a more systematic analysis of existing attack strategies which overlaps with existing results but also provides new insights. On the other hand, we develop new committing attacks that perform better than existing attacks.

[9] For sake of simplicity, we leave out details such as padding and domain separations, which have only minimal effect on our results.
[10] These attack strategies appear in both [20] and [16].

Revisiting Generic Attack Strategies (cf. Section 3). As our first contribution, we revisit existing attack strategies w.r.t. the above generic sponge construction. Regarding the notions, we consider CMT-4 and $\text{CMT}^{\star}_{\text{K}}$. In the case of CMT-4, the new part of our analysis is that we consider various combinations of the above design choices (key-blindings and zero-paddings), which allows us to derive generic attacks for arbitrary instantiations of the generic sponge-based construction. In the case of $\text{CMT}^{\star}_{\text{K}}$, also the analysis of the base-attacks (IS, CC, and TC) is new to the best of our knowledge.

Two-Stage Tag-Collision Attack (cf. Section 4). Our second and main contribution is a new committing attack that applies to schemes that deploy zero-padding. Our attack is a two-stage tag-collision (2S-TC) attack which reveals that the zero-padding bits are *not* equivalent to having a longer tag. The core observation is that we can attack the zero-padding bits and the tag bits separately: we first generate a (multi-)collision for the zero-padding bits; we then generate colliding tags starting from the initial (multi-)collision. We refer to these two parts of the attack as 2S-TC_1 and 2S-TC_2. Instead of the complexity of $2^{\frac{\tau+z}{2}}$ that the simple collision attack has, our two-stage attack has a complexity of $2^{\frac{z}{2}} + 2^{\tau+1}$ (complexity of 2S-TC_1 plus complexity of 2S-TC_2). In the more general case, i.e., when generating a μ-multi-collision for the zero-padding bits, the complexity of our 2S-TC attack is $2^{\text{mulcoll}(z,\mu)} + 2^{\tau+1-\log(\mu)}$.

For Ascon-128 [12] (using 128-bit of prefix zero-padding), our new 2S-TC attack outperforms the existing TC attack by about 3 bit: $2^{125.26}$ (2S-TC) vs. 2^{128} (TC). Further, for other parameters, our attack can outperform the existing TC attack more significantly: for $\tau = 128$ and $z = 256$, our 2S-TC attack outperforms the previously known attack by 62 bit (2^{130} compared to 2^{192}). More generally, we observe that $z \geq 2\tau$ leads to the most significant improvements of our new attack compared to the TC attack. While one might argue that 2τ bit of zero-padding are overkill, there are parameter settings where this can be practically relevant, such as the NIST standard of Ascon [24], which also specifies a tag-truncated version with tag length down to 64 or even 32.[11] In that case, the best improvement of our 2S-TC attack is already achieved for $z = 128$ (or even $z = 64$). One might again argue that, instead of zero-padding tag-truncated Ascon, a larger tag could be used from the start. However, zero-padding is a non-invasive method that can be applied generically, whereas changing the tag length may interfere with existing protocol specifications.

Finally, our 2S-TC attack reveals that the aforementioned intuition (zero-padding can be viewed as a longer tag) is *not* true. This invalidates the corresponding proof in [20]. Instead of their claimed attack complexity, we obtain the following, where the fourth term stems from our 2S-TC attack:

$$\min\Big(l + \frac{\min(\tau,\kappa)}{2}, \min\left(\frac{c_F}{2}, \frac{c_G + z}{2}\right), \frac{\tau + z}{2}, \log(2^{\text{mulcoll}(z,\mu)} + 2^{\tau+1-\log(\mu)})\Big)\,.$$

[11] NIST recommends 32-bit tags only after a specific risk analysis is performed.

Additional Committing Attacks (cf. Section 5). Our third contribution is two new committing attacks:

1. The first attack is an extension of the IS attack for $\mathrm{CMT}^{\star}_{\mathrm{K}}$. Due to the restrictions of the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ notion, the IS attack is only applicable if some concrete inequality for the parameters holds, namely $2^{r_G} \geq 2^{\frac{\nu}{2}+l}$ for r_G being the rate in the G component, ν the nonce length, and l the IV length. This stems from the fact that we need to generate enough states to apply a preimage attack on the IV and a birthday attack on the nonce. The main idea of the new attack is to use multi-collisions as a tool to generate more states such that an IS attack can be applied even if $2^{r_G} < 2^{\frac{\nu}{2}+l}$.

   More precisely, we prepend a multi-collision attack to the IS attack: start from a fixed intermediate value and XOR all possible 2^{r_G} inputs; when applying the inverse permutation to the resulting 2^{r_G} states, we are likely to see a small multi-collision in the rate part—we denote the size of this multi-collision by μ. Then, we apply the basic IS attack, not just from a single state, but these μ states, yielding $\mu 2^{r_G}$ states. If $\mu 2^{r_G} \geq 2^{\frac{\nu}{2}+l}$ holds, the states suffice to apply the preimage and birthday attacks. We essentially "stretch" the attack to a second round, which is why we refer to it as a two-round initial state (2R-IS) attack. We show that this attack applies to certain parameter choices of the second round CAESAR submission Ketje [6] in Example 4.
2. The second attack is a new meet-in-the-middle (MitM) type attack that applies to the plain duplex construction that uses the tag KB (and neither of the other two). An interesting takeaway is that by adding the tag KB—and no others—to the duplex construction, a new differential propagation emerges, which enables a new attack that outperforms the existing attack strategies i.e., IS, CC, and TC. Moreover, under some additional conditions, the attack even outperforms the standard attacks against the plain duplex construction, i.e., *without* the tag KB. This shows that the tag KB can have a negative effect on committing security. We stress that the NIST LWC finalist Schwaemm [3] uses this configuration of key-blindings. The intuition is as follows. When the construction adopts a large capacity, the cost for IS and CC can be more expensive than that of TC. Namely, the best generic attack is to generate a collision on the zero-padding bits and the tag simultaneously, which requires $\frac{\tau+z}{2}$. Following this nature, the attack first fixes the whole state during the associated data processing, then chooses many associated data values and proceeds with the computation in the forward direction until the tag, where the tag is now an XOR of a τ-bit state and a τ-bit key. A collision on the zero-padding bits is generated only based on the birthday paradox, while the birthday paradox on the τ-bit state is now unnecessary because the difference can be canceled by the tag KB. For this purpose, the attack also chooses many associated data values and proceeds with the computation in the backward direction until the initial state. The two computations in both directions are independent and provide multiple choices of differences to be canceled in the meet-in-the-middle manner.

The meet-in-the-middle attack has complexity $2^{\frac{\tau+z}{4}+\frac{l}{2}+1}$ which can outperform the basic TC attack with complexity $2^{\frac{\tau+z}{2}}$ for certain parameters. Example 5 provides parameters where this attack has complexity 2^{113} which outperforms the best generic attack (with complexity 2^{128}) by 15 bit.

Generic Attacks in Case of State-Update-Functions (cf. Appendix D). As our fourth contribution, we revisit the existing attack strategies when the generic sponge construction deploys a so-called *state update function* for absorbing and squeezing. Here, a function ξ takes as input the rate-part of the state and message (or associated data) and outputs the new rate-part of the state and a ciphertext (or nothing in case the associated data was given as input). The plain duplex construction can be viewed as a specific instantiation of ξ, where the function is given as $\xi(S, I) = (S \oplus I, S \oplus I)$. This function is 2^r-to-1, i.e., for every element in the image of ξ, there are 2^r many inputs. Other examples for ξ exist, for instance, in the NIST LWC finalists Schwaemm, Photon-Beetle, Romulus, and Gift-Cofb. The striking difference is that these state update functions are only 2-to-1 (Photon-Beetle, Gift-Cofb) or 1-to-1 (Schwaemm, Romulus). While the results for CMT-4 are similar to the plain duplex case, the impact on $\text{CMT}^{\star}_{\text{K}}$ security is significant: Neither type of state-update-function, 2-to-1 or 1-to-1, allows for the IS attack. The CC attack is impossible for 1-to-1 functions and can be significantly worse (compared to the duplex case) for 2-to-1 functions. The latter is illustrated for Photon-Beetle in Example 6. The TC attack is the only one that remains feasible for both cases.

1.3 Related Work

The first definition of committing security is due to Farshim et al. [15], though using the term *key-robustness*. CMT-4 security was introduced by Bellare and Hoang [4] while $\text{CMT}^{\star}_{\text{K}}$ security is due to Menda et al. [19] who provide a general framework for committing security, which allows all kinds of variations.

Several works have shown that the absence of committing security (in various forms) can lead to real-world vulnerabilities: Dodis et al. [13] developed the *fast message franking attack* against the Facebook message franking protocol which allowed a malicious user to send contents that the recipient cannot report. Len et al. [17] showed an improved key-recovery attack via the *partitioning oracle attack*. Albertini et al. [1] discuss, among other, the *Subscribe-with-Google attack*.

There are many transforms that turn AEAD schemes into committing ones or those that are only weakly committing into stronger ones. Bellare and Hoang [4] provide transforms to achieve CMT_{K} security as well as achieving CMT-4 from CMT_{K} security. Struck and Weishäupl [22] provide modified versions that also apply in the setting of side-channel leakage. Additional transforms are given in [8,9]. Zero-padding was proposed as a method to achieve CMT_{K} security by Albertini et al. [1], though it was not claimed to work for all schemes and needs individual analysis. Krämer et al. [16] give examples for which zero-padding does not lead to CMT_{K} security, while Naito et al. [20] analyze the effect of zero-padding to strengthen CMT-4 security.

2 Preliminaries

In this section, we introduce the notation and revisit the definitions of authenticated encryption with associated data (AEAD) and committing security.

Notation. We denote by $\{0,1\}^*$ the set of all finite binary strings and by $\{0,1\}^n$ the set of strings of length exactly n for any natural number n. For a string $X \in \{0,1\}^*$, $|X|$ denotes its bit length. The concatenation of two strings X and Y is written as $X \parallel Y$. For two equal length strings X and Y, we write $X \oplus Y$ to denote their bit-wise XOR. For a string $X \in \{0,1\}^*$ with $|X| \geq k$, we define $\lceil X \rceil_k$ to denote the first k bits of X and $\lfloor X \rfloor_k$ to denote the last k bits of X. For any string $X \in \{0,1\}^*$, we define $\langle X \rangle_n$ to be the n bits binary representation of X. The sponge-based AEAD schemes studied in this work are built from a public, fixed-size permutation $\pi : \{0,1\}^b \to \{0,1\}^b$. All other notations that we rely on will be defined in their respective sections. Table 7 summarizes the notation.

Authenticated Encryption with Associated Data. An AEAD scheme over the message space $\mathcal{M}$, nonce space $\mathcal{N}$, key space $\mathcal{K}$, associated data space $\mathcal{A}$, cipher text space $\mathcal{C}$ and tag space $\mathcal{T}$ is a tuple of algorithms (Enc, Dec) defined as follows

- $\textsc{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$, i.e., $(K, N, A, M) \mapsto (C, T)$
- $\textsc{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}$, i.e., $(K, N, A, C, T) \mapsto M/\perp$

Here, $\perp$ signifies that the ciphertext-tag pair is invalid and consequently rejected. Additionally, the correctness condition is imposed as follows,

$$\textsc{Dec}(K, N, A, \textsc{Enc}(K, N, A, M)) = M \text{ for any } (K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}\,.$$

The *tidyness* property requires that re-encrypting a decrypted message yields the original ciphertext:

$$(M = \textsc{Dec}(K, N, A, C)) \Rightarrow (C = \textsc{Enc}(K, N, A, M))\,.$$

Throughout this work, we consider $\mathcal{K} = \{0,1\}^\kappa$, $\mathcal{N} = \{0,1\}^\nu$, $\mathcal{T} = \{0,1\}^\tau$, and $\mathcal{A} = \mathcal{M} = \mathcal{C} = \{0,1\}^*$. We call the triple (K, N, A) a *context*.

Generic Sponge Construction. The pictorial description and the pseudocode for the generic AEAD scheme based on a plain duplex sponge are given in Fig. 1 and Fig. 3, respectively. It consists of two components: F and G. The former component, F, processes the context. It starts by setting the initial state of the sponge to be the concatenation of key K, nonce N, and (optionally) IV. It then absorbs the associated data, r_F bit in each round, until the final state S is obtained. The latter component, G, processes the message. It sets the initial state to S and then, in each round, absorbs r_G bits of the message and outputs the first r_G bits of the state as the ciphertext. After the final permutation, the last τ bits are outputted as the tag T. The pictorial description in Fig. 1 displays both the key-blinding and the zero-padding in gray; the pseudocode in Fig. 3 does not show these for sake of simplicity.

Game CMT-4	Game $\mathrm{CMT}^{\star}_{\mathrm{K}}$
$(K, N, A, M), (K', N', A', M') \leftarrow \mathcal{A}()$	$(K, N, A, M), (K', N', A', M') \leftarrow \mathcal{A}()$
if $(K, N, A) = (K', N', A')$	**if** $K = K' \vee (N, A) \neq (N', A')$
return 0	**return** 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$	$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(C', T') \leftarrow \textsc{Enc}(K', N', A', M')$	$(C', T') \leftarrow \textsc{Enc}(K', N', A', M')$
return $[\![(C, T) = (C', T')]\!]$	**return** $[\![(C, T) = (C', T')]\!]$

Fig. 2. Security games CMT-4 and $\mathrm{CMT}^{\star}_{\mathrm{K}}$.

Committing Security. CMT-4 and $\mathrm{CMT}^{\star}_{\mathrm{K}}$ security are defined via the security games shown in Fig. 2. Both require an adversary to output two tuples of context-message pairs that encrypt to the same ciphertext. The difference between the two security notions lies in the requirements towards the adversary's output: CMT-4 security requires merely different context, i.e., $(K, N, A) \neq (K', N', A')$, whereas $\mathrm{CMT}^{\star}_{\mathrm{K}}$ requires the keys to be distinct while the rest of the contexts agree, i.e., $K \neq K'$ and $(N, A) = (N', A')$.

Throughout this work, we state attack complexities in terms of the expected number of calls to the permutation π and/or its inverse. We ignore constant factors, in particular when an attack involves inverting multiple rounds of the sponge, i.e., walking backwards through several permutation calls, then each inversion adds only a constant multiplicative overhead, which we omit for readability. The complexities are often given as base-2 logarithms.

3 Existing Attack Classes Against the Generic Duplex

In this section, we consider the generic sponge-based AEAD construction described above and apply existing committing attacks. We start the section by introducing zero-padding, key-blinding, and the existing attack classes in Section 3.1. Subsequently, in Section 3.2 and Section 3.3, we cover CMT-4 attacks against the plain duplex and the zero-padded duplex, respectively. Similarly, in Section 3.4 and Section 3.5, we study $\mathrm{CMT}^{\star}_{\mathrm{K}}$ attacks against the plain duplex and the zero-padded duplex, respectively. Lastly, in Section 3.6, we give a short summary of the results in the case a state-update-function is used, while the detailed analysis can be found in Appendix D.

3.1 Design Features and Existing Attack Classes

For the generic construction on top of the duplex construction, we consider the following modifications.

Zero-Padding. The message M is padded with the z-bit constant 0^z. Depending on whether the padding is applied at the beginning of the message or at the end, we have two variants:

$\mathsf{Duplex}(K, N, A, M)$

1 : $S \leftarrow F(K, N, A)$
2 : $(C, T) \leftarrow G(S, M)$
3 : **return** (C, T)

$G(S, M)$

1 : $M_1, \ldots, M_\gamma \xleftarrow{r_G} M$
2 : $\check{G}_1 \leftarrow \pi(S)$
3 : **for** $i = 1$ **to** γ
4 : $\quad \hat{G}_{i+1} \leftarrow \check{G}_i \oplus (M_i \parallel 0^{c_G})$
5 : $\quad C_i \leftarrow \left\lceil \hat{G}_{i+1} \right\rceil_{r_G}$
6 : $\quad \check{G}_{i+1} \leftarrow \pi(\hat{G}_{i+1})$
7 : $C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\gamma \rceil_{|M|}$
8 : $T \leftarrow \left\lfloor \check{G}_{\gamma+1} \right\rfloor_\tau$
9 : **return** (C, T)

$F(K, N, A)$

1 : $A_1, \ldots, A_\alpha \xleftarrow{r_F} A$
2 : $\check{F}_1 \leftarrow \pi(K \parallel N \parallel IV)$
3 : **for** $i = 1$ **to** $\alpha - 1$
4 : $\quad \hat{F}_{i+1} \leftarrow \check{F}_i \oplus (A_i \parallel 0^{c_F})$
5 : $\quad \check{F}_{i+1} \leftarrow \pi(\hat{F}_{i+1})$
6 : $S \leftarrow \check{F}_\alpha \oplus (A_\alpha \parallel 0^{c_F})$
7 : **return** S

$F^{-1}(S, A)$

1 : $A_1, \ldots, A_\alpha \xleftarrow{r_F} A$
2 : $\check{F}_\alpha \leftarrow S \oplus (A_\alpha \parallel 0^{c_F})$
3 : **for** $i = \alpha - 1$ **to** 1
4 : $\quad \hat{F}_{i+1} \leftarrow \pi^{-1}(\check{F}_{i+1})$
5 : $\quad \check{F}_i \leftarrow \hat{F}_{i+1} \oplus (A_i \parallel 0^{c_F})$
6 : $K \parallel N \parallel IV \leftarrow \pi^{-1}(\check{F}_1)$
7 : **return** (K, N, IV)

Fig. 3. Pseudocode for Generic duplex construction. The description of F^{-1} is used in some attacks but not used by the construction itself.

- Prefix Zero-Padding: Here the padded message is $M_{\text{zp}} = 0^z \parallel M$.
- Suffix Zero-padding: Here the padded message is $M_{\text{zp}} = M \parallel 0^z$.

For constructions with zero-padding, apart from the regular verification check, the verified-decryption algorithm additionally checks whether the extracted message ends (or begins) with z zeros.

Key-Blinding. In duplex constructions with key-blinding, the underlying key K is additionally XORed into the capacity part of some specified intermediate state. Depending on the position of the key-blinding, we consider the following three cases:

- The key is XORed after the first permutation of F (Init KB):

$$\check{F}_1 \leftarrow \pi(K \parallel N \parallel IV) \oplus (0^{r_F} \parallel K)\,.$$

- The key is XORed before the final permutation of G (Fin KB):

$$\hat{G}_{\gamma+1} \leftarrow \hat{G}_{\gamma+1} \oplus (0^{r_G} \parallel K)\,.$$

- The key is XORed after the final permutation of G (Tag KB):

$$T \leftarrow \left\lfloor \check{G}_{\gamma+1} \right\rfloor_\tau \oplus K\,.$$

We consider both the duplex and zero-padded duplex constructions under all combinations of the three key-blinding mechanisms. We denote these by $\mathsf{Duplex}[b_1, b_2, b_3]$ and $\mathsf{ZP\text{-}Duplex}[b_1, b_2, b_3]$, respectively, where $[b_1, b_2, b_3]$ specifies which key-blindings are enabled. In particular, $b_i = 1$ indicates that the corresponding key-blinding is applied, and $b_i = 0$ otherwise; here b_1, b_2, and b_3 correspond to Init KB, Fin KB, and Tag KB, respectively. We use the notation $b_i = \star$ to indicate that b_i can take the values both 0 and 1. For example, $\mathsf{Duplex}[\star, 1, 1]$ represents both $\mathsf{Duplex}[0, 1, 1]$ and $\mathsf{Duplex}[1, 1, 1]$. By a slight abuse of notation, we write $\mathsf{Duplex}[\star, \star, \star]$ to cover all constructions except $\mathsf{Duplex}[0, 0, 0]$.

Existing Attack Classes. In this section, we apply three generic attack strategies against the committing security of the above-mentioned classes of duplex constructions. These attacks were previously identified in the literature and form the baseline for our analysis. Let the duplex sponge construction be as depicted in Fig. 1 and Fig. 3. The state after processing the context (K, N, A) via the F component is denoted by $S = F(K, N, A)$. The subsequent G component processes a (possibly zero-padded) message M and outputs a ciphertext C and a tag T. Now we discuss a high-level overview of the attack strategies:

- *Initial State (IS) Attack.* This attack aims to invert the sponge construction by varying contexts until a valid initial state is reached (matching the specified IV). Typically, there are two approaches: *forward-backward* and *backward-backward*. In this first approach, an intermediate state S is computed using a forward computation, and we aim to find one context corresponding to S that inverts the F component to produce a valid initial state. In the latter approach, the adversary fixes any intermediate state and aims to obtain two different contexts which result in this chosen intermediate state.
- *Capacity Collision (CC) Attack.* The adversary searches for two contexts that lead to a collision in the capacity part of an internal state. Such a collision in the capacity bits can be easily extended to a full-state collision by absorbing a matching input block. From that point onward, the computations are identical, yielding the same ciphertext and tag.
- *Tag Collision (TC) Attack.* The adversary directly targets the tag, generating many contexts $(K^{(i)}, N^{(i)}, A^{(i)})$ and corresponding message blocks M_i until a tag collision occurs. Of course, the message blocks have to be chosen carefully so that the corresponding ciphertext blocks match, which can always be done for a duplex sponge.

3.2 CMT-4 Results for Duplex Sponges

Here, we describe three attack strategies against the CMT-4 security of the duplex sponge construction, namely the initial-state collision attack, the capacity collision attack, and the tag-collision attack. The corresponding results are summarized in Table 1.

Table 1. CMT-4 results for duplex sponge construction. The symbol † indicates that if the construction uses Init KB, the attack is possible only when $r_F \geq \frac{\kappa}{2} + l$.

	Cost		
	IS	CC	TC
Duplex$[0,0,0]$	l	$\min\left(\frac{c_F}{2}, \frac{c_G}{2}\right)$	$\frac{\tau}{2}$
Duplex$[\star,\star,\star]$	$\frac{\kappa}{2} + l^{\dagger}$	$\min\left(\frac{c_F}{2}, \frac{c_G}{2}\right)$	$\frac{\tau}{2}$

CMT-4 Initial State Attack. The CMT-4 initial state (IS) attack works by computing some intermediate state and then going backwards from this state to a different, valid initial state. A simple choice would be the state S that is fed from F to G. First, pick an arbitrary context (K, N, A) and compute S; then, invert S using different associated data A' until an initial state of the form $K' \parallel N' \parallel IV$ is found. For the attack to succeed, we merely need to hit the correct IV when inverting F, yielding a complexity of l. Since the associated data are arbitrary in length, there is also no restriction on how many initial states the attack can generate. The pseudocode of the attack can be found in Appendix E (cf. Algorithm 4).

Key-Blinding. Deploying any key-blinding affects the attack. Instead of finding initial states, we can look for initial states with the same keys, thereby avoiding the effect of later key-blindings. In case of initial key-blinding, the options are limited, as the attacker has to choose the first round of F: after XORing the first associated data block but before XORing the key-blinding. Then, inverting this state for different associated data blocks until an initial state with the same key (and obviously the correct IV) is found. Similarly, the attack extends to the case of final key-blinding and tag key-blinding. The difference is that we can choose any intermediate state of F as the key-blinding only occurs in G. For all three variants, this yields a complexity of $\kappa+l$. Instead of performing the attack by first computing an intermediate state in the forward direction, followed by inverting to find a second initial state (we will henceforth refer to this as a *forward-backward* strategy), we can choose an arbitrary intermediate state and invert multiple times to find different initial states yielding this intermediate state. We call this *backward-backward* strategy. For this strategy, we require only a collision for the keys, which yields a complexity of $\frac{\kappa}{2} + l$. Since this complexity is better than that of the previous strategy, the backward-backward attack is the best approach here. Note that for initial key-blinding, the options are limited, as the attacker has to choose the first round of F, thus, only works if $r_F \geq \frac{\kappa}{2} + l$. For final and tag key-blinding, since the associated data are arbitrary in length, there is no restriction on how many initial states the attack can generate. For completeness, the pseudocode of the attack is given in Appendix E (cf. Algorithm 5) and the formal result (with and without key-blinding) is summarized in Theorem 1.

It is easy to see that the same attack algorithm works for constructions having multiple key-blinding (at least two of initial, final or tag key-blinding). As argued previously, if the combination contains initial key-blinding, the attack works if $r_F \geq \frac{\kappa}{2} + l$. Otherwise, i.e., construction having final and tag key-blinding, we do not have any restriction.

CMT-4 Capacity Collision Attack. The CMT-4 capacity collision (CC) attack works by obtaining intermediate states which agree in their capacity. This partial collision is then turned into a full-state collision via the absorption of the current block. The adversary targets an intermediate round of F or G and generates different states ($\check{F}_i$ or $\check{G}_i$) using different contexts. After turning the capacity collision into a full-state collision by absorbing fitting inputs, the remaining computations will be identical, thereby also yielding identical tags. In case the adversary targets an intermediate state of G, it needs to ensure that the prior ciphertext blocks are equal which can be done by choosing appropriate message blocks. Depending on whether a state in F or G is targeted, the (logarithmic) complexity is $\frac{c_F}{2}$ or $\frac{c_G}{2}$, yielding an overall complexity of $\min(\frac{c_F}{2}, \frac{c_G}{2})$.

For completeness, the attack algorithm and the security theorem are formally presented in Appendix E (cf. Algorithm 6 and Theorem 5).

Key-Blinding. The attack easily extends to any of the three key-blindings by fixing some key in advance and trying to obtain a capacity collision by varying nonce and associated data. This ensures that equal states remain equal since the same key is XORed in the capacity. In case of initial/final key-blinding, there is no need to choose a fixed key if the capacity collision targets any state *after* the key-blinding takes place.[12] In case of tag key-blinding, note that choosing different keys thwarts the attack altogether. Since the states are equal in any round after the capacity collision, they are equal after the final permutation. By XORing different keys to the same state, the tags will be distinct.

CMT-4 Tag Collision Attack. The CMT-4 tag collision (TC) attack works by choosing arbitrary contexts until a collision of the tags is found. More precisely, for different contexts, the F component is computed. Then, G is computed on the state obtained from F, where the message blocks are chosen to ensure that some arbitrary but fixed ciphertext is hit—this is possible due to the duplex structure by setting the message as the XOR of the rate part of the state and the ciphertext block. By generating enough contexts like this, there will be two contexts (and messages) that result in the same tag, thereby yielding a CMT-4 attack. The complexity is $\frac{\tau}{2}$. The pseudocode of the attack and the security result is deferred to Appendix E (cf. Algorithm 7 and Theorem 9).

[12] For initial key-blinding, one could target the first round of F, which requires $\kappa + \nu \geq \frac{c_F}{2}$ to find a capacity collision in the first round. For final key-blinding, the capacity collision can only target the final round of G.

Table 2. CMT-4 results for duplex sponge construction with zero-padding. The symbol † indicates that if the construction uses Init KB, the attack is possible only when $r_F \geq \frac{\kappa}{2} + l$.

	Cost			
	IS	CC		TC
		Prefix	Suffix	
ZP-Duplex$[0,0,0]$	l	$\min\left(\frac{c_F}{2}, \frac{c_G+z}{2}\right)$	$\min\left(\frac{c_F}{2}, \frac{c_G}{2}\right)$	$\frac{\tau+z}{2}$
ZP-Duplex$[\star,\star,\star]$	$\frac{\kappa}{2} + l^{\dagger}$	$\min\left(\frac{c_F}{2}, \frac{c_G+z}{2}\right)$	$\min\left(\frac{c_F}{2}, \frac{c_G}{2}\right)$	$\frac{\tau+z}{2}$

Key-Blinding. The attack is unaffected by any key-blinding (and combinations of key-blinding) as the key-blinding only affects the capacity part of the states, which allows to still find the matching messages that yield the target ciphertext.

3.3 CMT-4 Results for Duplex Sponges with Zero-Padding

In this section, we present the CMT-4 results for duplex sponge constructions with zero-padding. The results for prefix zero-padding and suffix zero-padding are summarized in Table 2.

CMT-4 Initial State Attack. The initial state attacks described on the duplex sponge in Section 3.2 still apply when the scheme uses zero-padding, irrespective of where the zeros are padded. The reason is that the attack targets the first component F and ensures that the states outputted by F agree. By using the same (zero-padded) message, the states will agree throughout the computation of G, yielding identical ciphertexts and tags. For completeness, we describe the formal security result in the appendix (cf. Theorem 2).

CMT-4 Capacity Collision Attack. The capacity collision attack described for duplex sponge still works when the scheme deploys zero-padding. When targeting an intermediate state in F, the attack works with the same complexity $\frac{c_F}{2}$. When targeting a state in G, the attack complexity differs depending on the position of the zeros. In the prefix case, the complexity increases to $\frac{c_G+z}{2}$ as the states, additionally, have to agree in the bits where the zero-padding is applied to ensure the same ciphertexts. In the suffix case, the complexity remains $\frac{c_G}{2}$ but the capacity collision can no longer be obtained for the final round (where the zero-padding is absorbed).[13] We summarize the discussion in the appendix (cf. Theorem 6).

[13] Strictly speaking, the attack can be applied to the final round but then the complexity would be $\frac{c_G+z}{2}$ just as for the prefix case.

Table 3. $\mathrm{CMT}^{\star}_{\mathrm{K}}$ results for duplex sponge construction. For each attack, the first column (Cost) denotes the cost of the attack while the second column (DoF) denotes the degree of freedom that the adversary has. Note that the degree of freedom must exceed the cost for the attack to be effective. A — is used to indicate that no non-trivial attack was found.

	IS		CC		TC	
	Cost	DoF	Cost	DoF	Cost	DoF
$\mathsf{Duplex}[0,0,0]$	$\frac{\nu}{2}+l$	r_G	$\frac{c_G}{2}$	κ	$\frac{\tau}{2}$	κ
$\mathsf{Duplex}[1,0,0]$	$\frac{\nu}{2}+l+\kappa$	r_G	$\frac{c_G}{2}$	κ	$\frac{\tau}{2}$	κ
$\mathsf{Duplex}[\star,1,0]$	$\frac{\nu}{2}+l+\kappa$	$r_G+\kappa$	$\frac{c_G}{2}$	κ	$\frac{\tau}{2}$	κ
$\mathsf{Duplex}[\star,\star,1]$	—	—	—	—	$\frac{\tau}{2}$	κ

CMT-4 Tag Collision Attack. In case of zero-padding—regardless of prefix or suffix—the attack still applies. However, the adversary loses control over the parts of the message that are zero-padded. This worsens the complexity to $\frac{\tau+z}{2}$ as the states additionally have to agree on the z bits where the zeros are XORed. The resulting theorem is given in the appendix (cf. Theorem 10).

3.4 $\mathrm{CMT}^{\star}_{\mathrm{K}}$ Results of Duplex Sponges

Here, we describe three attack strategies against the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ security of the duplex sponge construction, namely the initial-state collision attack, the capacity collision attack, and the tag-collision attack. The corresponding results are summarized in Table 3.

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Initial State Attack. Regarding the initial state attack for $\mathrm{CMT}^{\star}_{\mathrm{K}}$, there are two attack strategies: (1) *forward-backward* and (2) *backward-backward*. The former is the one we already described for the CMT-4 attack: start from an initial state, compute some intermediate state, and invert trying to find a different, valid initial state. The latter works by picking an intermediate state and inverting twice, trying to obtain different, valid initial states.

There are some crucial differences for the IS attack regarding $\mathrm{CMT}^{\star}_{\mathrm{K}}$ compared to CMT-4. First of all, we can no longer target any intermediate state of F. The reason is that $\mathrm{CMT}^{\star}_{\mathrm{K}}$ requires *the same* associated data for both outputs, which would result in only one initial state, hence making the entire contexts equal. Instead, we need to target a state of G. The best option is the first round (when absorbing the first message block)—for any later round, inverting would require a collision in the rate to guarantee equal ciphertexts. This first-round state can be inverted using different first message blocks, yielding different states which can then be inverted using some arbitrary but fixed associated data to obtain many initial states. The complexity of the attack depends on the strategy. In the forward-backward case, the intermediate state is generated honestly via the forward computation. This forward computation fixes a

Algorithm 1 A $\mathrm{CMT}^{\star}_{\mathrm{K}}$ Attack exploiting Initial State Collision.

function $\mathrm{CMT}^{\star}_{\mathrm{K}}$`_IS_Adversary`( )
  Choose an associated data A
  Choose a state $\hat{G}_2$
  for $i = 1$ to q **do**
    Choose one message block $M_1^{(i)}$
    $\check{G}_1^{(i)} \leftarrow \hat{G}_2 \oplus M_1^{(i)} \parallel 0^{c_G}$
    $S \leftarrow \pi^{-1}(\check{G}_1^{(i)})$
    $K^{(i)} \parallel N^{(i)} \parallel IV^i \leftarrow F^{-1}(S, A)$
  if $\exists i, j \in [q]\colon N^{(i)} = N^{(j)} \wedge IV^i = IV^{(j)} = IV$ **then**
    return $(K^{(i)}, N^{(i)}, A, M_1^{(i)} \parallel M)$ and $(K^{(j)}, N^{(j)}, A, M_1^{(j)} \parallel M)$
  return $\perp$

nonce that the backward computation (when inverting using different messages) has to hit in the initial state. On top of that, the IV also has to be hit, which yields an overall complexity of $\nu + l$. In the backward-backward case, there is no longer a fixed nonce to be hit but merely a collision to be found, which results in a complexity of $\frac{\nu}{2} + l$ and thus in the better attack. The degree of freedom is r_G and the attack requires $r_G \geq \frac{\nu}{2} + l$ to succeed. The pseudocode of the attack is given in Algorithm 1.

In the following, we discuss the differences of the attack when the scheme uses key-blinding.

Initial Key-Blinding. Key-blinding after the first permutation in F affects the attack. Due to the XOR of the key, we can no longer take any initial state for which nonces agree, and the correct IV is hit. We additionally need to hit the correct key that we used before inverting the very first permutation. Using the *forward-backward* strategy, this yields a complexity of $\kappa + \nu + l$, which equals the state size n. Since the degree of freedom r_G is smaller than n, this attack strategy can never succeed. In contrast, for the *backward-backward* strategy, the complexity is $\kappa + \frac{\nu}{2} + l$, which yields the better attack. The attack algorithm is depicted in Algorithm 2. The adversary's degrees of freedom come from the key (κ bits) and the first message block (r_G bits). To generate enough trials to hit the IV (l bits) and achieve a nonce collision requiring about $2^{\nu/2}$ candidates after IV filtering), the total number of trials is $2^{\kappa + r_G}$. The attack requires: $\kappa + r_G \geq l + \nu/2 + \kappa$, which basically implies the rate of the G component $r_G \geq l + \nu/2$.

Final Key-Blinding. Here, we fix the state after key-blinding and before the final permutation, and then apply a *backward-backward* strategy. The resulting attack complexity is $\kappa + \frac{\nu}{2} + l$. The corresponding attack algorithm is analogous to the one for initial key-blinding described in Algorithm 2, with the boxed instructions applied. Note that one could think about an alternate strategy by

Algorithm 2 A $\mathrm{CMT}^{\star}_{\mathrm{K}}$ Attack exploiting IS Collision on Sponge Duplex with initial and final key-blinding. The boxed statements are applicable only for the variant using final key-blinding.

function $\mathrm{CMT}^{\star}_{\mathrm{K}}$`_IS_Adversary_InitialKB( )` `FinalKB`
  Choose one associated data block A_1
  Choose a state $\hat{G}_2$
  for $i = 1$ to q **do**
    $M_1^{(i)} \leftarrow \langle i \rangle_{r_G}$
    for $j = 1$ to q' **do**
      $\check{G}_1^{(i)} \leftarrow \hat{G}_2 \oplus M_1^{(i)} \parallel 0^{c_G}$ $\quad$ $\boxed{\check{G}_1^{(i)} \leftarrow \hat{G}_2 \oplus M_1^{(i)} \parallel K^j}$
      $S^{(i)} \leftarrow \pi^{-1}(\check{G}_1^{(i)}) \oplus (0^{r_F} \parallel K^j)$ $\quad$ $\boxed{S^{(i)} \leftarrow \pi^{-1}(\check{G}_1^{(i)})}$
      $(K, N^{(j)}, IV^{(j)}) \leftarrow F^{-1}(S_j^{(i)}, A_1)$
      if $K = K^{(j)}$ and $IV^{(j)} = IV$ **then**
        Store $(K^{(j)}, N^{(j)}, M_1^{(i)})$ in a list $\mathcal{L}$
  if $\exists (K, N, M), (K', N', M') \in \mathcal{L}$: $N = N'$ **then**
    return (K, N, A_1, M) and (K', N', A_1, M')
  return $\perp$

fixing a state before the final key-blinding and proceeding similarly; however, that would require a tag collision.

Initial and Final Key-Blinding. When key-blinding is applied at both the initial and final, the same attack as in the key-blinding final case applies. Specifically, we fix a state after the key-blinding and before the final permutation and then employ a *backward-backward* strategy. The resulting attack complexity remains unchanged, namely $\kappa + \frac{\nu}{2} + l$.

Tag Key-Blinding. Having the key-blinding on the final tag thwarts the attack altogether. Since the states are identical from the first ciphertext block onwards, they remain the same up to the state after the final permutation. To compute the tag, two different keys are XORed to this state, which ensures that the tags will always be different.

For completeness, we summarize the above discussion in the appendix (cf. Theorem 3.)

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Capacity Collision Attack. The $\mathrm{CMT}^{\star}_{\mathrm{K}}$ capacity collision (CC) attack works like the CMT-4 attack except for a few differences. Since $\mathrm{CMT}^{\star}_{\mathrm{K}}$ requires the same associated data, a capacity collision in F can no longer be turned into a full-state collision. Instead, the attack has to target an intermediate state in G which yields a complexity of $\frac{c_G}{2}$. Note further that the only degree of freedom that the adversary has is the choice of the key—the nonce, just as

the associated data, has to be fixed in advance. The attack therefore requires $\kappa \geq \frac{c_G}{2}$ to succeed.

In the following, we discuss the differences of the attack when the scheme uses key-blinding.

Initial Key-Blinding. The attack still applies if initial key-blinding is used.

Final Key-Blinding. In case of final key-blinding, the attack needs to target the final round of G: the state *after* the key-blinding is XORed but *before* the final message block is XORed.[14]

Tag Key-Blinding. By the same argument as for CMT-4, the attack does not work when tag key-blinding is used.

We summarize this discussion in the appendix (cf. Theorem 7).

$\mathbf{CMT}^{\star}_{\mathbf{K}}$ Tag Collision Attack. The $\mathrm{CMT}^{\star}_{\mathrm{K}}$ tag collision (TC) attack works exactly like the CMT-4 attack, except that the adversary can no longer vary nonce and associated data. Instead, values for these two context components have to be fixed in advance. The complexity of the attack remains to be $\frac{\tau}{2}$ but the only degree of freedom for the adversary is the choice of the key. This means that $\kappa \geq \frac{\tau}{2}$ has to hold in order for the attack to be applicable.

The argumentation for key-blinding works the same, yielding the same complexity, as for CMT-4, and again requires κ to be greater than this complexity for the attack to succeed. We summarize the above discussion in the appendix (cf. Theorem 11).

3.5 $\mathbf{CMT}^{\star}_{\mathbf{K}}$ Results of Duplex Sponges with Zero-Padding

Here, we present the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ results for duplex sponge constructions with zero-padding. The results for prefix zero-padding and suffix zero-padding are summarized in Table 4 and Table 5, respectively.

$\mathbf{CMT}^{\star}_{\mathbf{K}}$ Initial State Attack. All attacks proceed essentially in the same manner as in the case without zero-padding; the only difference is that the degree of freedom is reduced. For the prefix zero-padding case, since the first z bits of the first message block are fixed to 0, we no longer have full control over the first message block. Consequently, we can generate at most $2^{r_G - z}$ distinct states (or $2^{r_G + \kappa - z}$ in the case of final key-blinding), instead of 2^{r_G} (or $2^{r_G + \kappa}$, respectively).

All attacks for the case without zero-padding remain unaffected by suffix zero-padding. This is because the attacks rely on finding distinct initial states

[14] Note that the attack might further be affected by the padding of the final message block. In case of a simple 10^* padding, the adversary can minimize the effect by choosing the message length to only have a single bit of padding.

Table 4. $\mathrm{CMT}^{\star}_{\mathrm{K}}$ results for duplex sponge construction with prefix zero-padding.

	IS		CC		TC	
	Cost	DoF	Cost	DoF	Cost	DoF
ZP-Duplex$[0,0,0]$	$\frac{\nu}{2}+l$	r_G-z	$\frac{c_G+z}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[1,0,0]$	$\frac{\nu}{2}+l+\kappa$	r_G-z	$\frac{c_G+z}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[\star,1,0]$	$\frac{\nu}{2}+l+\kappa$	$r_G-z+\kappa$	$\frac{c_G+z}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[\star,\star,1]$	—	—	—	—	$\frac{\tau+z}{2}$	κ

Table 5. $\mathrm{CMT}^{\star}_{\mathrm{K}}$ results for duplex sponge construction with suffix zero-padding.

	IS		CC		TC	
	Cost	DoF	Cost	DoF	Cost	DoF
ZP-Duplex$[0,0,0]$	$\frac{\nu}{2}+l$	r_G	$\frac{c_G}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[1,0,0]$	$\frac{\nu}{2}+l+\kappa$	r_G	$\frac{c_G}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[\star,1,0]$	$\frac{\nu}{2}+l+\kappa$	$r_G-z+\kappa$	$\frac{c_G+z}{2}$	κ	$\frac{\tau+z}{2}$	κ
ZP-Duplex$[\star,\star,1]$	—	—	—	—	$\frac{\tau+z}{2}$	κ

that collide in the first round of message absorption. Accordingly, the degree of freedom remains the same as in the case without zero-padding. However, when key-blinding is present at the final stage, the degree of freedom is reduced, since the last z bits of the final message block are fixed to 0.

For completeness, we summarize the above discussion in the appendix (cf. Theorem 4).

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Capacity Collision Attack. The capacity-collision attack proceeds analogously to the case without zero-padding. However, due to ZP-prefix, the first z bits of the first message block are fixed to 0, and consequently, the state after absorbing the first message block must also agree on these z bits. Hence, the effective collision search space has size c_G+z. The degree of freedom remains unchanged, as it arises solely from the key.

In the suffix case, both the attacks and their complexities are exactly the same as in the setting without zero-padding when no key-blinding is applied or when key-blinding is applied only at the initial position. If key-blinding is applied at the final position, then due to ZP-suffix, the last z bits of the final message block are fixed to 0. Consequently, the state after absorbing the last message block must also agree on these z bits in addition to matching the capacity. Hence, the effective collision search space has size c_G+z.

For completeness, we provide the corresponding security result in the appendix (cf. Theorem 8).

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Tag Collision Attack. For both ZP-prefix and ZP-suffix, the attacks proceed exactly as in the case without zero-padding. The only difference is that the state after absorbing the first message block (for ZP-prefix) or the last message block (for ZP-suffix) must additionally agree on the z bits, besides matching the tag. Consequently, the attack complexity becomes $(\tau+z)/2$, while the degree of freedom remains unchanged, as it arises solely from the key. For completeness, we provide the corresponding security result in the appendix (cf. Theorem 12).

3.6 Results for Sponges with State-Update-Functions

Several of the NIST LWC finalists deploy so-called state-update-functions, which are applied in alternating manner with the block-cipher or permutation used by the schemes. Our focus is on sponge-based schemes with state-update-functions. A state-update-function ξ maps an input state and an input value to an output state alongside an output value: In the G component, the input state is the rate part of the current state and the input value is a message block, while the output state is an updated rate part and the output value is a ciphertext block. The same is true for the F component, with the exception that associated data blocks are the input values instead of message blocks, and the output value is discarded. An illustration of a duplex sponge with state-update-function is given in Fig. 9 with the corresponding pseudocode being described in Fig. 10. While the details of the state update functions differ among the NIST LWC finalists, we observe that all of them are either invertible or 2-to-1.

We consider the three generic attack strategies (IS, CC, and TC) also for duplex sponges with state-update-functions, and take zero-padding and key-blinding into account. The detailed results can be found in Appendix D. The main insight of the analysis is that the usage of state-update-functions can restrict or even prevent certain attacks—especially in the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ case. This is due to the fact that for $\mathrm{CMT}^{\star}_{\mathrm{K}}$ attacks, we cannot target the F component. While this was also a disadvantage in the standard duplex setting, it becomes even more pronounced if state-update-functions are used: in the F component, the output values of the state-update-function are discarded, which makes attacking the F component easier than the G component. In particular, the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ capacity collision attack is not possible for invertible state-update-functions. Moreover, the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ initial state attack is impossible for both invertible and 2-to-1 state-update-functions.

4 Two-Stage Tag-Collision Attack

As shown in Section 3.3, when considering the currently known strategies, the tag collision attack against a zero-padded duplex sponge has complexity $\frac{\tau+z}{2}$. This complexity seems reasonable as the zero-padding can be viewed as an increase in the tag length. In this section, however, we present an improved attack which debunks this intuition. In Section 4.1, we describe this new attack and in Section 4.2, we apply it to Ascon.

4.1 Description of the Attack

Consider a duplex sponge with prefix zero-padding. The idea of our new attack is to first generate a collision on the z rate-bits affected by the zero-padding followed by generating a tag collision using the initial collision on the zero-padding bits. For the second part (the tag collision) we exploit the property of the duplex sponge that one can always hit any desired ciphertext block by absorbing the matching message block. We refer to this attack as a two-state tag collision (2S-TC) attack. The two parts of the attack are referred to as 2S-TC$_1$ and 2S-TC$_2$. We give the pseudocode of the attack in Algorithm 3.

In the first step of the attack (2S-TC$_1$), we choose arbitrary contexts (K, N, A) until a collision on the z rate-bits affected by the zero-padding is found. Thereby, we obtain same ciphertext bits for these two states since the zero-padding does not change these bits.[15] For the second step of the attack (2S-TC$_2$), we consider the first round in G that does not contain any zero-padding, denoted by the index i. In the previous round $i-1$, we have two states $\check{G}_{i-1}^{(1)}$ and $\check{G}_{i-1}^{(2)}$ forming a collision, i.e., these states agree in the rate bits where the zero-padding is applied. By XORing appropriate message bits to these two states, we obtain $\hat{G}_{i-1}^{(1)}$ and $\hat{G}_{i-1}^{(2)}$, which agree in their first r_G bits. Then, we compute the states $\check{G}_{i}^{(1)} \leftarrow \pi(\hat{G}_{i-1}^{(1)})$ and $\check{G}_{i}^{(2)} \leftarrow \pi(\hat{G}_{i-1}^{(2)})$, and fix an arbitrary ciphertext block C_i. We choose the message blocks $M_i^{(1)} \leftarrow \left\lceil \check{G}_i^{(1)} \right\rceil_{r_G} \oplus C_i$ and $M_i^{(2)} \leftarrow \left\lceil \check{G}_i^{(2)} \right\rceil_{r_G} \oplus C_i$, which guarantees that the i-th ciphertext blocks for the two states are the same. This second step can be repeated over multiple rounds of message absorption by fixing ciphertext blocks $C_{i+1}, \ldots, C_\gamma$ and choosing the messages analogously as described for C_i. Finally, we check for a collision in the tags $T^{(1)}$ and $T^{(2)}$ and, if no collision is found, we repeat the second step with a different C_i (and potentially different $C_{i+1}, \ldots, C_\gamma$).

Attack Complexity and Requirements. The core property of the 2S-TC attack is that the second part can be repeated *without* repeating the first part. The complexity of the first part of the attack is $2^{\frac{z}{2}}$ as we generate a collision on z bits. For the second part of the attack, the complexity is 2^τ. This stems from the fact that for each ciphertext block, we need to make 2 queries to π and find a tag collision with probability $\frac{1}{2^\tau}$. We repeat this for 2^τ values of the ciphertext, which yields a total cost of $2^{\tau+1}$. The complexity of the entire attack is $2^{\frac{z}{2}} + 2^{\tau+1}$.

Comparison to Simple Tag-Collision Attack. Our new two-stage attack outperforms the simple tag-collision attack if $2^{\frac{z}{2}} + 2^{\tau+1} < 2^{\frac{z+\tau}{2}}$. This leaves the question, of which parameters this is the case. Note that $2^{\frac{z}{2}} + 2^{\tau+1} < 2^{\max(\frac{z}{2}, \tau+1)+1}$, i.e., if $\max(\frac{z}{2}, \tau+1) + 1 < \frac{z+\tau}{2}$ holds, the above requirement

[15] Note that in case z is not a multiple of r, we fix a value for the remaining ciphertext bits and choose the message bits accordingly.

is fulfilled. To determine the cases when our attack outperforms the simple tag-collision attack, we distinguish the following two cases:

1. If $\max(\frac{z}{2}, \tau+1) = \frac{z}{2}$, the inequality $\frac{z}{2}+1 < \frac{z+\tau}{2}$ holds as long as $2 < \tau$. Thus, our attack outperforms the existing one if $4 < 2\tau + 2 < z$.
2. If $\max(\frac{z}{2}, \tau+1) = \tau+1$, the inequality $\tau+2 < \frac{z+\tau}{2}$ holds as long as $\frac{\tau}{2}+2 < \frac{z}{2}$. Thus, our attack outperforms the existing one if $\tau + 4 < z \leq 2\tau + 2$.

In total, we observe that our attack outperforms the simple tag-collision attack as long as the length of the zero-padding is greater than $\tau + 4$.

Additionally, in cases where the simple tag-collision attack outperforms the IS attack and CC attack, our 2S-TC attack is the overall best known committing attack. This occurs, for example, when l and the capacities (c_F and c_G) are large compared to τ and z. The following example describes such an instance.

Example 1. Consider a state size of 512 bits, $r_F = r_G = 128$, $c_F = c_G = 384$, $l = 256$, $\nu = 128$, $\kappa = 128$, $\tau = 128$, and $z = 256$. Then, the complexity of an IS attack is $2^l = 2^{256}$, the complexity of a CC attack is $2^{\min\{\frac{c_F}{2}, \frac{c_G+z}{2}\}} = 2^{192}$, and the complexity of a TC attack is $2^{\frac{\tau+z}{2}} = 2^{192}$. Our new attack has complexity $2^{\frac{z}{2}} + 2^{\tau+1} = 2^{128} + 2^{129} \approx 2^{129.58}$, which improves the previously best attacks by roughly 62 bits.

Improving the Attack via Multi-Collisions. We make the following modification to the two-stage tag-collision attack: Instead of producing only two states that collide in the z bits where the zero-padding is applied, we generate a multi-collision of size μ. While this makes the first part of the attack more costly, the complexity of the second part decreases from $2^{\tau+1}$ to $2^{\tau+1-\log(\mu)}$. This stems from the fact that we now have more than two states, namely μ, for producing the tag collision. Then, for each ciphertext block, we need to make μ queries to π and find a tag collision with probability $\frac{\mu^2}{2^{\tau+1}}$. We repeat this for $\frac{2^{\tau+1}}{\mu^2}$ values of the ciphertext, which yields a total cost of $\frac{2^{\tau+1}}{\mu}$ for the second part of the attack. In total, the complexity is $2^{\mathrm{mulcoll}(z,\mu)} + 2^{\tau+1-\log(\mu)}$, where $\mathrm{mulcoll}(z, \mu)$ denotes the (base-2 logarithmic) complexity to find a μ-multi-collision in z bits.

The multi-collision strategy can improve our attack in cases where $\tau+1 > \frac{z}{2}$. Note that otherwise, i.e., if $\tau + 1 \leq \frac{z}{2}$, the first summand from the previous attack complexity $2^{\frac{z}{2}} + 2^{\tau+1}$ is already the dominant one such that the above multi-collision approach cannot improve the attack. In the setting $\tau+1 > \frac{z}{2}$, our previous attack yields the complexity $2^{\frac{z}{2}}+2^{\tau+1}$. By determining the value of μ for which $2^{\mathrm{mulcoll}(z,\mu)}$ and $2^{\tau+1-\log(\mu)}$ are most balanced, the multi-collision variant of the attack can achieve a complexity smaller than $2^{\frac{z}{2}} + 2^{\tau+1}$. To compute the multi-collision complexity, we use the result from Suzuki et al. [23], which shows that, with a probability of approximately 2^{-1}, $2^{\mathrm{mulcoll}(z,\mu)}$ is given by

$$(\mu!)^{1/\mu} \cdot 2^{\frac{\mu-1}{\mu}z} + \mu - 1\,. \tag{1}$$

We give a comparison of the complexities for the existing and new tag collision attacks in Fig. 4 and Fig. 5. While these figures show the same plots, the second

one covers a wider range of values for z; Fig. 4 allows to see where our new attack outperforms the simple TC attack, while Fig. 5 shows that our new attack eventually also gets more complex for $z > 2\tau$ (while still outperforming the TC attack significantly). For both figures, the choice of μ is not depicted. Appendix C shows plots where the attack complexities are given for various values of μ s.t. one can read off the optimal choices for μ. In summary, using multi-collisions, we obtain two improvements:

1. The multi-collision variant of the 2S-TC attack outperforms the $\mu = 2$ case for all $z < 2(\tau + 1)$. For $z \geq 2(\tau + 1)$, the best complexity is obtained by choosing $\mu = 2$.
2. We can extend the range of zero-padding lengths, for which our new two-stage approach outperforms the standard TC attack. The 2S-TC attack with $\mu = 2$ improved the TC attack for $z > \tau + 4$. Using multi-collision, we can relax the lower bound for z for which our 2S-TC attack outperforms the TC one—see Fig. 4 where the red (solid) line and the green (dash-dotted) line intersect.

In the following we present two examples illustrating the above improvements. The first example shows that the multi-collision strategy can also yield further improvements in cases where the new two-stage tag-collision attack with $\mu = 2$ already outperformed the standard tag collision attack.

Example 2. Consider a state size of 512 bits, $r_F = r_G = 128$, $c_F = c_G = 384$, $l = 256$, $\nu = 128$, $\kappa = 128$, $\tau = 128$, and $z = 140$. Then, the complexity of an IS attack is $2^l = 2^{256}$, the complexity of a CC attack is $2^{\min\{\frac{c_F}{2}, \frac{c_G+z}{2}\}} = 2^{192}$, and the complexity of a TC attack is $2^{\frac{\tau+z}{2}} = 2^{134}$. The two-stage tag-collision attack with $\mu = 2$ yields complexity $2^{\frac{z}{2}} + 2^{\tau+1} = 2^{70} + 2^{129} < 2^{130}$, while the multi-collision attack yields the minimal attack complexity 2^{127} for $\mu = 2^3$.

Next, we present an example where our previous two-stage tag-collision attack (with $\mu = 2$) does not outperform the standard attack (as $\tau = z$); however, the new multi-collision one does.

Example 3. Suppose that the sizes of the state, capacity, and IV are sufficiently large. For example, the state size is 512 bits, $r = 128$, $c_F = c_G = 384$, $l = 256$, $\nu = 128$, $\kappa = 128$, $\tau = 128$, and $z = 128$. Then, the complexity of an IS attack is $2^l = 2^{256}$, the complexity of a CC attack is $2^{\min\{\frac{c_F}{2}, \frac{c_G+z}{2}\}} = 2^{192}$, and the complexity of a TC attack is $2^{\frac{\tau+z}{2}} = 2^{128}$. Hence, a TC attack is the most efficient attack in the existing analysis, while our attack can be even faster. For $\tau = 128$, the complexities of Eq. (1), $\tau + 1 - \log \mu$, and their sum for some μ are listed in Table 6. As it shows, the attack complexity is minimal when $\mu = 15$, with a total of $2^{125.26}$.

Modifications. First, note that the two-stage attack is not affected by any key-blinding. Secondly, when considering suffix instead of prefix zero-padding, the

Table 6. Complexity of our new two-stage attack 2S-TC (the individual parts 2S-TC$_1$ and 2S-TC$_2$ as well as the overall complexity) for various values of μ.

μ	Complexity		
	2S-TC$_1$ (mulcoll(z,μ))	2S-TC$_2$ ($\tau + 1 - \log \mu$)	Overall
11	118.66	125.54	125.55
13	120.66	125.30	125.35
15	122.15	125.09	125.26
17	123.31	124.91	125.32
19	124.25	124.75	125.52

Algorithm 3 2S-TC Attack.

function `2S_TC_Adversary`( )
  for $i_1 = 1$ to q_1 **do** ▷ First stage (2S-TC$_1$)
    $S^{(i_1)} \leftarrow F(K^{(i_1)}, N^{(i_1)}, A^{(i_1)})$
    $\check{G}_1^{(i_1)} \leftarrow \pi(S^{(i_1)})$
  if $\exists j_1, k_1 \in [q_1]\colon \left\lceil \check{G}_1^{(j_1)} \right\rceil_z = \left\lceil \check{G}_1^{(k_1)} \right\rceil_z$ **then** ▷ Zero-padding collision
    Let $S^{(j_1)}$ and $S^{(k_1)}$ be the states resulting in $\check{G}_1^{(j_1)}$ and $\check{G}_1^{(k_1)}$
    for $i_2 = 1$ to q_2 **do** ▷ Second stage (2S-TC$_2$)
      Choose a fresh ciphertext $C^{(i_2)}$
      Choose message $M^{(i_2,1)}$ s.t. $G(S^{(j_1)}, M^{(i_2,1)}) = (C^{(i_2)}, T^{(i_2,1)})$
      Choose message $M^{(i_2,2)}$ s.t. $G(S^{(k_1)}, M^{(i_2,2)}) = (C^{(i_2)}, T^{(i_2,2)})$
      if $T^{(i_2,1)} = T^{(i_2,2)}$ **then** ▷ Tag collision
        $(K, N, A, M) \leftarrow (K^{(j_1)}, N^{(j_1)}, A^{(j_1)}, M^{(i_2,1)})$
        $(K', N', A', M') \leftarrow (K^{(k_1)}, N^{(k_1)}, A^{(k_1)}, M^{(i_2,2)})$
        return $(K, N, A, M), (K', N', A', M')$
  return $\perp$

above attacks (with and without multi-collisions) are also applicable, however, only in a restricted form. Since the zero-padding affects the very last round, we are restricted in the number of times we can repeat the second part of the attack, i.e., searching for a tag collision for different ciphertexts. We can only repeat the attack 2^{r-z} times since we can only choose 2^{r-z} different ciphertexts for the last round. In particular, the attack can only work if $z < r$.

While we describe the 2S-TC attack as a CMT-4 attack, it can also be adapted to work as a CMT$^\star_{\mathrm{K}}$ attack. Instead of varying the contexts (K, N, A) until a multi-collision is found, we only vary the key and keep nonce and associated data fixed. Thus, the CMT$^\star_{\mathrm{K}}$ attack is only applicable if the key length is sufficient to generate enough states to find a μ multicollision in z bits.

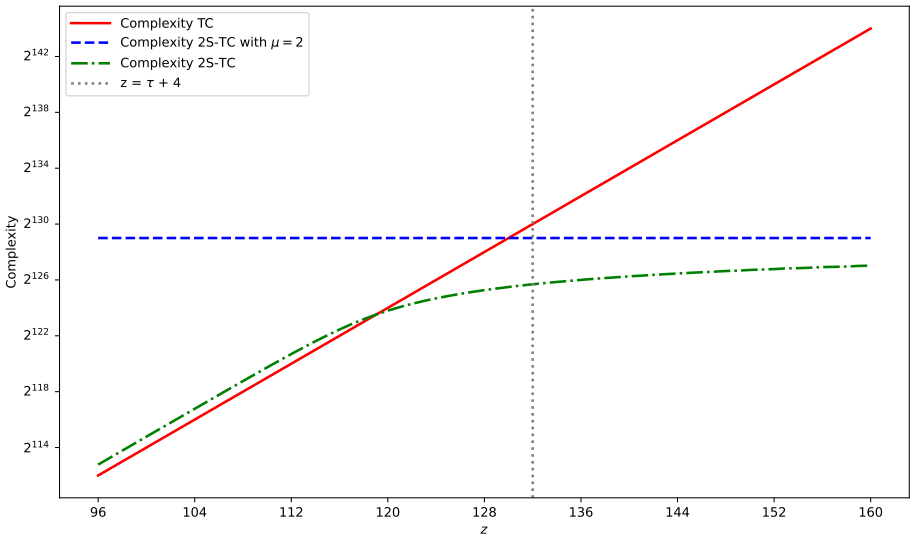

Fig. 4. Complexity of the different attack strategies for tag length $\tau = 128$ and varying zero-padding length $z \in [96, 160]$. The red (solid) curve corresponds to the standard tag-collision attack, while the other two curves depict our new two-stage tag-collision attacks: the blue (dashed) curve depicts the attack for $\mu = 2$, while the green (dash-dotted) curve uses the value for μ that minimizes the complexity (note that the exact value of μ is not shown in the picture). The gray (dotted) line marks the point where the zero-padding length is $\tau + 4$: for all values of z to the right of this line, the 2S-TC attack with $\mu = 2$ is better than the standard TC attack—as described in Section 4. Note that the actual intersection of the red (solid) and blue (dashed) line is further to the left. This deviation stems from us using the estimation $2^{\frac{z}{2}} + 2^{\tau+1} < 2^{\max(\frac{z}{2},\tau+1)+1}$ in our computations. In the case that $\mu \neq 2$ is used, i.e., the green (dash-dotted) line, our two state attack outperforms for even smaller values of z.

4.2 Application to Ascon

In this section, we apply our new 2S-TC attack to the generic sponge-based AEAD scheme Ascon [12] and its standardized version [24].

Applicability of the 2S-TC Attack. The 2S-TC attack is not affected by any of the key-blindings and thus can be applied to the Ascon-like structure as well as the simple duplex construction. With the Init KB and Tag KB, the IS attack complexity is $l + \frac{\min(\tau,\kappa)}{2}$, hence the previous generic attack complexity is $\min(l + \frac{\min(\tau,\kappa)}{2}, \min(\frac{c_F}{2}, \frac{c_G+z}{2}), \frac{\tau+z}{2})$. Our attack updates the generic attack complexity as

$$\min\left(l + \frac{\min(\tau,\kappa)}{2}, \min\left(\frac{c_F}{2}, \frac{c_G+z}{2}\right), \frac{\tau+z}{2}, \log(2^{\mathrm{mulcoll}(z,\mu)} + 2^{\tau+1-\log(\mu)})\right).$$

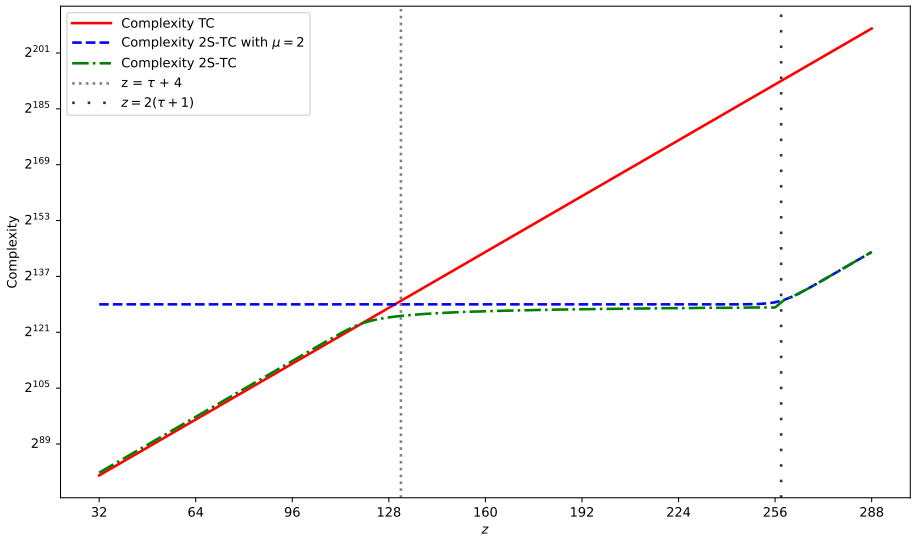

Fig. 5. Complexity of the different attack strategies for tag length $\tau = 128$ and varying zero-padding length $z \in [32, 288]$. For further explanation on the curves, we refer to Fig. 4, which shows a zoomed-in version of the same plot. The gray (sparsely-dashed) line at $z = 2(\tau + 1)$ marks which term from $2^{\frac{z}{2}} + 2^{\tau+1}$, i.e., the 2S-TC attack with $\mu = 2$, is the dominant one: left of the line, the second term is dominant, while right of the line, the first one is.

Counterexample to the Previous Bound. The CMT-4 security of the Ascon-like structure with zero-padding has already been studied in [20], which claims that for any zero-padding scheme that expands the input message M by adding z bits of zeros in any position, the bit security against CMT-4 attacks is $\min(\frac{l+\tau}{2}, \frac{c}{2}, \frac{\tau+z}{2})$, where $c_F = c_G$ is assumed and its value is denoted by c. Note that this is a bound where the tightness is unknown because of the gap between l and $\frac{l}{2}$ in the first term for the IS attack. Nevertheless, our attack provides a counterexample to this claimed bound [20] for certain parameter choices. Let us recall Example 3. Here, the state size is 512 bits, $r = 128$, $c = 384$, $l = 256$, $\nu = 128$, $\kappa = 128$, $\tau = 128$, and $z = 128$, where the zero-padding is applied in the prefix encoding or anywhere as long as 128-bit message input is available after the last padding bits. The lower-bound by [20] tells that at least $\min\left(\frac{256+128}{2}, \frac{384}{2}, \frac{128+128}{2}\right) = 128$-bit security is ensured. On the other hand, our attack breaks the CMT-4 security with a complexity of $2^{125.26}$ as calculated in Table 6.

Ascon with truncated tag. Ascon is standardized by NIST [24]. Its AEAD mode adopts the duplex structure with the initial, final, and tag key-blindings. Ascon adopts the following parameters: the state size is 320 bits, $r = r_F = r_G = 128$, $c_F = c_G = 192$, $l = 64$, $\nu = 128$, $\kappa = 128$, and $\tau = 128$. The

NIST standard makes several changes from the NIST-LWC-submission version of Ascon-128a [12]. In particular, the NIST standard allows implementers to truncate the tag when a small tag is preferred. Suppose that the tag is truncated to 64 bits, which is a legitimate tag size in other AEAD modes by NIST, e.g., AES-CCM [14]. In fact, 64-bit tags are used in several protocols, such as TLS [18] and media encryption applications [7]. Suppose that the zero-padding with $z = 64$ via the prefix zero-padding is adopted.

With the previous evaluation, the generic attack cost is given by $\min\{l + \frac{\tau}{2}, \frac{c}{2}, \frac{\tau+z}{2}\}$, where $c = c_F = c_G$. Using the parameters of Ascon with 64-bit tags, the complexity is $\min\{64 + \frac{64}{2}, \frac{192}{2}, \frac{64+64}{2}\} = 64$ bits. Our 2S-TC attack with the complexity of $2^{\mathrm{mulcoll}(z,\mu)} + 2^{\tau+1-\log(\mu)}$ can improve this complexity. With $z = \tau = 64$, the value of μ that minimize the total attack complexity is $\mu = 9$, which gives $\mathrm{mulcoll}(z, \mu) = 2^{58.94}$, $2^{\tau+1-\log(\mu)} = 2^{61.83}$, and the total attack complexity is $2^{62.01}$.

Ascon-128. Although it was not selected as a standard by NIST, Ascon-128 is a primary member of the design of the Ascon-family during the NIST lightweight cryptographic competition [12]. Compared to the NIST standard, Ascon-128 adopts the smaller rate $r = 64$ and the larger capacity $c = 256$. The tag size is fixed to $\tau = 128$ and the tag truncation is not allowed. When $z = 128$, the previous generic attack cost is $\min(64 + \frac{128}{2}, \frac{256}{2}, \frac{128+128}{2}) = 128$ bits. As exemplified in Section 4.1, our 2S-TC attack can improve this complexity, which becomes minimal at $2^{125.26}$ by choosing $\mu = 15$.

5 Additional Committing Attacks

In this section, we give two more new, improved committing attacks. In Section 5.1 we describe an extended version of the IS attack while Section 5.2 gives a novel attack following a meet-in-the-middle style.

5.1 Two-Round Initial State Attack

The $\mathrm{CMT}^{\star}_{\mathrm{K}}$ initial state attack against the duplex sponge described in Section 3.4 has (logarithmic) complexity $\frac{\nu}{2} + l$ and is based on the requirement that $r_G \geq \frac{\nu}{2} + l$. In the following, we present a new variant of the attack using multi-collisions that also works for certain parameter sets with $r_G < \frac{\nu}{2} + l$. We start with a description of the new attack, then describe a concrete setting where our attack is applicable, and compare it to the existing IS attack.

Description of the Two-Round Attack. As for the standard IS attack, the core idea is to choose a ciphertext block C_1 and a state $\hat{G}_2$ (recall that $\hat{G}_2$ is the state *after* the absorption in the first round) such that the first r_G bits of $\hat{G}_2$ agree with C_1, and compute back to the initial state using different first message blocks, but the same associated data. For a $\mathrm{CMT}^{\star}_{\mathrm{K}}$ attack, we need to find a

collision for the nonce and hit the correct IV—hence we require at least $2^{\frac{\nu}{2}+l}$ states for the attack. Since we have 2^{r_G} options for each message block, we can generate 2^{r_G} states in one round of message absorption. The simple IS attack only covers the case where $r_G \geq \frac{\nu}{2} + l$ as then we can generate enough states for the birthday and preimage attack in one round. Otherwise, if $r_G < \frac{\nu}{2} + l$, we have to extend the attack to a second round—this is why we refer to this attack as two-round initial-state (2R-IS) attack. We choose C_2 and a state $\hat{G}_3$—again conditioned on C_2 agreeing with the first r_G bits of $\hat{G}_3$—and generate 2^{r_G} different states $\check{G}_2^{(1)}, \ldots, \check{G}_2^{(r_G)}$ by XORing 2^{r_G} different messages $M_2^{(i)}$ to the rate-part of $\hat{G}_3$. Then, we apply the inverse permutation π^{-1} to all of these states $\check{G}_2^{(1)}, \ldots, \check{G}_2^{(r_G)}$. Since we need the same ciphertext block C_1 for the committing attack, we are only interested in states $\pi^{-1}\left(\check{G}_2^{(i)}\right)$ for which the rate-parts agree. We can consider this as a balls-into-bins problem where 2^{r_G} balls (the 2^{r_G} states obtained from inversion) are uniformly at random distributed into 2^{r_G} bins (the 2^{r_G} options for the rate bits that the states can have). The maximum load B is the number of balls in the bin that has the most balls. We choose μ such that the probability $\Pr[B \geq \mu]$ is greater or equal to $\frac{1}{2}$ using [21, Theorem 1]. Then, we find a μ-multi-collision with probability at least $\frac{1}{2}$. For each of the corresponding μ states, we can XOR all possible 2^{r_G} messages to the rate part, resulting in $\mu 2^{r_G}$ states $\check{G}_1^{(1)}, \ldots, \check{G}_1^{(\mu 2^{r_G})}$. If $\mu 2^{r_G} \geq 2^{\frac{\nu}{2}+l}$, we have generated enough states to apply the attack. We illustrate this attack in Fig. 11.

Note that if $\mu 2^{r_G} \leq 2^{\frac{\nu}{2}+l}$, it is unlikely that we can extend the 2R-IS attack to another round—like we can for the simple IS attack described above—as μ is too small to guarantee a μ'-multi-collision with $\mu' > \mu$ for the next round.

Applying the Attack. For the 2R-IS attack to outperform the standard IS one, we need parameter sets where $r_G \leq \frac{\nu}{2} + l$, but r_G is close to $\frac{\nu}{2} + l$ such that $\mu 2^{r_G} \geq 2^{\frac{\nu}{2}+l}$. Ketje [6], a second round candidate to the CAESAR competition allows variable parameter sets, such that we can construct a fitting example: Ketje SR uses a permutation of $b = 400$ bits with rate $r = 36$ bits and the initial state consists of the key of variable length up to 382 bits, the nonce of length up to $382 - \kappa$ bits and pad$(\kappa + \nu)$. We choose a parameter set with $\kappa = 340$ and $\nu = 42$, as then $r = 36 < \frac{(382-340)}{2} + 18 = 39$ holds. For $r = 36$, we choose $\mu = 2^{3.32}$, i.e., $2^{3.32} \cdot 2^{36} > 2^{39}$.

Comparison to the Standard Attack. Instead of the 2R-IS attack described above, we could also use a standard birthday and preimage attack with less queries, which will result in a worse success probability for the attack. To compare our attack to this one, we compute how many queries we need in each attack to obtain the same success probability.

The cost of our 2R-IS attack is $(\mu + 1)2^{r_G}$ and the success probability is $\frac{1}{4}$ (since we find a μ multi-collision with probability $\frac{1}{2}$ and a nonce collision with probability $\frac{1}{2}$). For the standard birthday attack we use $q = 2^{r_G}$ states with $r_G < \frac{\nu}{2} + l$. Then, we find $2^{r_G - l}$ initial states with the correct IV. Among

these, we find a nonce collision with probability $p = \frac{(2^{r_G-l})^2}{2^{\nu+1}}$. To achieve success probability $\frac{1}{4}$, we have to repeat the attack $n = \ln(1-\frac{1}{4})/\ln(1-p)$ times, i.e., in total the costs are $n2^{r_G}$. Thus, to compare the attacks, one needs to compare $\mu+1$ and n.

Example 4. For the Ketje SR parameter set described above, we have $r = 36$, $\nu = 42$, and $l = 18$. Then $\mu + 1 = 2^{3.32} + 1 < 2^{5.20} \approx n$ holds, i.e., our 2R-IS attack is slightly less costly.

5.2 Meet-in-the-Middle (MitM) Attack

CMT-4 Attack against (Exclusive) Tag Key-Blinding. This attack illustrates an interesting observation: by adding the tag key-blinding (tag KB) to the duplex sponge, a new differential propagation emerges. This enables a new attack approach that was not covered by the previous ones, i.e., IS, CC, and TC attacks, which improves the previous attack complexity for tag KB for certain parameter choices. Moreover the attack can even outperform attacks against the plain duplex construction, which shows that the tag KB can have a negative effect on committing security. In Appendix G, we elaborate on the conditions for this.

By applying the existing analysis from Table 2, when the attack complexities for IS attack and CC attack are high enough, the security of the tag KB is evaluated as $\frac{\tau+z}{2}$. Here, we demonstrate an attack faster than $\frac{\tau+z}{2}$.

The core idea of our attack is that, in the tag KB, two tags collide when $\lfloor \check{G}_{\gamma+1} \rfloor_\tau \oplus K = \lfloor \check{G}_{\gamma+1} \rfloor'_\tau \oplus K'$, namely $\Delta \lfloor \check{G}_{\gamma+1} \rfloor_\tau = \Delta K$, where $\lfloor \check{G}_{\gamma+1} \rfloor_\tau$ denotes the τ bits of the state before the tag KB is applied. To meet this relation, we first fix an internal state, and then, starting there, independently compute forwards and backwards by choosing distinct associated data values. The forward computation needs to generate a collision on rate bits onto which the zero-padding is XORed, and to choose the message blocks in a way such that the ciphertexts agree. Differently from the case without tag KB, the values before the tag key-blinding do not need to collide because their difference can be canceled out with ΔK independently computed in backward. Hence, a MitM-style attack is applied and can decrease the attack complexity from $2^{\frac{\tau+z}{2}}$.

To be more specific, the attack first fixes the internal state value during the AD processing (some state within F). Without loss of generality, we suppose that the state after the i-th AD block is processed, namely $\check{F}_i$, is fixed. We then process the following forward and backward computations in parallel.

Forward: search for pairs of $(A_{i+1}, \ldots, A_\alpha, M)$ and $(A'_{i+1}, \ldots, A'_\alpha, M')$ so that their ciphertext values collide. For each pair, $\lfloor \check{G}_{\gamma+1} \rfloor_\tau$ and $\lfloor \check{G}_{\gamma+1} \rfloor'_\tau$ are obtained, so is $\Delta \lfloor \check{G}_{\gamma+1} \rfloor_\tau$.

Backward: search for many $(A_i, \ldots, A_1)$ such that the IV at the initial state is satisfied. Then make all possible pairs of such $(A_i, \ldots, A_1)$, and the corresponding ΔK is obtained for each pair.

In the forward computation, the attacker chooses 2^X tuples $(A_{i+1}, \ldots, A_\alpha)$ and computes M so that the ciphertext takes a fixed pre-determined value except for the z bits of the message and computes $\left\lfloor \check{G}_{\gamma+1} \right\rfloor_\tau$. Then, we have 2^{2X} possible pairs, and 2^{2X-z} pairs will remain after the check of the collision on the z bits.[16] Namely, we have 2^{2X-z} choices of $\Delta \left\lfloor \check{G}_{\gamma+1} \right\rfloor_\tau$. In the backward computation, the attacker chooses 2^Y choices of $(A_i, \ldots, A_1)$ and computes the corresponding $(K \parallel N \parallel IV)$ by inverting π, and picks ones confirming the l-bit IV at the beginning, which reduces the the number of choices of $(A_i, \ldots, A_1)$ to 2^{Y-l}. Then, we make pairs among them, and we will have $2^{2(Y-l)}$ choices of ΔK. Finally, we match $\Delta \left\lfloor \check{G}_{\gamma+1} \right\rfloor_\tau$ and ΔK. For the case with $\kappa \geq \tau$, the equality $2^{2X-z} \times 2^{2(Y-l)} = 2^\tau$ must be satisfied and we want to have $X = Y$ to balance the complexities of two directions. Those give $X = Y = \frac{\tau+z}{4} + \frac{l}{2}$, so the total attack complexity is $2^X + 2^Y = 2^{\frac{\tau+z}{4}+\frac{l}{2}+1}$. A sketch of the attack is presented in Fig. 12 and the pseudocode is given in Appendix H.

Example 5. Suppose that the state size is 512 bits, $r = r_F = r_G = 128$, $c = c_F = c_G = 384$, $l = 64$, $\nu = 192$, $\kappa = 256$, $\tau = 256$, $z = 128$. Then, the complexities of the IS attack, CC attack, and TC attack are $2^{l+\frac{\min\{\kappa,\tau\}}{2}} = 2^{192}$, $2^{\frac{c}{2}} = 2^{192}$, and $2^{\frac{\tau+z}{2}} = 2^{192}$, respectively. Our MitM attack requires $2^{\frac{\tau+z}{4}+\frac{l}{2}+1} = 2^{96+32+1} = 2^{129}$, which outperforms the existing analysis.

Note that the attack requires that both the initial key-blinding and final key-blinding are not used, because the initial key-blinding prevents the attacker from performing the backward computation to the initial state, and the final key-blinding makes the computations in two directions dependent on each other.

6 Conclusion

In this paper, we studied CMT-4 and $\text{CMT}^\star_\text{K}$ attacks against AEAD schemes based on duplex sponges, taking into account both existing attack classes and three novel attacks proposed in this work. In addition to the plain duplex construction, we also considered schemes deploying state-update-functions—and in both cases, we also examined the commonly used design features of key-blinding and zero-padding. Based on our results, we can draw the following conclusions regarding the effect of these modifications on committing security.

– On the one hand, key-blinding can influence committing security in a positive manner: CMT-4 IS attacks become more costly when key-blinding is deployed, and in the case of $\text{CMT}^\star_\text{K}$, tag key-blinding even prevents IS and CC attacks altogether. On the other hand, our new CMT-4 meet-in-the-middle attack shows that, for certain parameter choices, tag key-blinding can also weaken the committing security in some settings.

[16] Note that the attack works regardless of the actual position of the zero-padding bits.

– For zero-padding, the results heavily depend on whether the prefix or suffix variant is deployed. To shield against CC attacks, prefix zero-padding is preferable. In contrast, our new 2S-TC attack works without limitations for prefix zero-padding, while suffix zero-padding—if sufficiently long—can thwart the attack entirely. These results refine the claims of [20] by showing that zero-padding is not always equivalent to having a longer tag and that the position of the zero-padding matters.
– For state-update-functions, our results reveal positive effects on $\mathrm{CMT}^{\star}_{\mathrm{K}}$ security if instances that are invertible and 2-to-1 are used.

In summary, we observe that the way in which these design features are deployed can have a substantial impact on the committing security of a scheme. Hereby, our new attacks reveal crucial vulnerabilities depending on these modifications and the parameters of the scheme.

References

1. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to abuse and fix authenticated encryption without key commitment. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 3291–3308. USENIX Association (Aug 2022), https://www.usenix.org/conference/usenixsecurity22/presentation/albertini
2. Bao, Z., Chakraborti, A., Datta, N., Guo, J., Nandi, M., Peyrin, T., Yasuda, K.: PHOTON-Beetle. Tech. rep., National Institute of Standards and Technology (2021), available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists
3. Beierle, C., Biryukov, A., dos Santos, L.C., Großschädl, J., Perrin, L., Udovenko, A., Velichkov, V., Wang, Q., Moradi, A., Shahmirzadi, A.R.: Schwaemm and Esch. Tech. rep., National Institute of Standards and Technology (2021), available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists
4. Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 845–875. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_29
5. Bellare, M., Hoang, V.T.: Succinctly-committing authenticated encryption. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part IV. LNCS, vol. 14923, pp. 305–339. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68385-5_10
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Ketje v2. Tech. rep., Submission to the CAESAR Competition (2016), available at https://keccak.team/files/Ketjev2-doc2.0.pdf
7. Campagna, M., Maximov, A., Mattsson, J.P.: Galois Counter Mode with Secure Short Tags (GCM-SST). https://datatracker.ietf.org/doc/draft-mattsson-cfrg-aes-gcm-sst/03/ (2023)
8. Chan, J., Rogaway, P.: On committing authenticated-encryption. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part II. LNCS, vol. 13555, pp. 275–294. Springer, Cham (Sep 2022). https://doi.org/10.1007/978-3-031-17146-8_14

9. Chen, S., Karadžić, V.: Committing authenticated encryption: Generic transforms with hash functions. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part I. LNCS, vol. 15601, pp. 93–123. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91107-1_4
10. Chen, Y.L., Davidson, M., Dworkin, M., Kelsey, J., Sasaki, Y., Turan, M.S., Chang, D., Mouha, N., Thompson, A.: Requirements for cryptographic accordions. Tech. Rep. NIST IR 8552, National Institute of Standards and Technology (2025). https://doi.org/10.6028/NIST.IR.8552, https://doi.org/10.6028/NIST.IR.8552
11. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP. Tech. rep., National Institute of Standards and Technology (2021), available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists
12. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Tech. rep., National Institute of Standards and Technology (2021), available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists
13. Dodis, Y., Grubbs, P., Ristenpart, T., Woodage, J.: Fast message franking: From invisible salamanders to encryptment. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 155–186. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_6
14. Dworkin, M.: Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. NIST Special Publication (SP) 800-38C SP 800-38C, National Institute of Standards and Technology, Gaithersburg, MD, USA (2007). https://doi.org/10.6028/NIST.SP.800-38C, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
15. Farshim, P., Orlandi, C., Roşie, R.: Security of symmetric primitives under incorrect usage of keys. IACR Trans. Symm. Cryptol. **2017**(1), 449–473 (2017). https://doi.org/10.13154/tosc.v2017.i1.449-473
16. Krämer, J., Struck, P., Weishäupl, M.: Committing AE from Sponges security analysis of the NIST LWC finalists. IACR Trans. Symm. Cryptol. **2024**(4), 191–248 (2024). https://doi.org/10.46586/tosc.v2024.i4.191-248
17. Len, J., Grubbs, P., Ristenpart, T.: Partitioning oracle attacks. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 195–212. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/len
18. McGrew, D., Bailey, D.: AES-CCM Cipher Suites for Transport Layer Security (TLS). RFC 6655 (Jul 2012). https://doi.org/10.17487/RFC6655, https://www.rfc-editor.org/info/rfc6655
19. Menda, S., Len, J., Grubbs, P., Ristenpart, T.: Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part IV. LNCS, vol. 14007, pp. 379–407. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30634-1_13
20. Naito, Y., Sasaki, Y., Sugawara, T.: Committing security of Ascon: Cryptanalysis on primitive and proof on mode. IACR Trans. Symm. Cryptol. **2023**(4), 420–451 (2023). https://doi.org/10.46586/tosc.v2023.i4.420-451
21. Schulte-Geers, E., Waggoner, B.: Balls and bins – simple concentration bounds. https://arxiv.org/abs/2205.14494 (2022), https://arxiv.org/abs/2205.14494
22. Struck, P., Weishäupl, M.: Constructing committing and leakage-resilient authenticated encryption. IACR Trans. Symm. Cryptol. **2024**(1), 497–528 (2024). https://doi.org/10.46586/tosc.v2024.i1.497-528

23. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multi-collisions. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **91-A**(1), 39–45 (2008). https://doi.org/10.1093/IETFEC/E91-A.1.39, https://doi.org/10.1093/ietfec/e91-a.1.39
24. Turan, M.S., McKay, K., Kang, J., Kelsey, J., Chang, D.: Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions. NIST Special Publication (SP) 800-232 SP 800-232, National Institute of Standards and Technology, Gaithersburg, MD, USA (2025). https://doi.org/10.6028/NIST.SP.800-232, https://doi.org/10.6028/NIST.SP.800-232

A Notation Overview

Table 7 provides an overview of the notation used throughout this work.

Table 7. Frequently used notation.

Symbol	Meaning
τ	Tag length (in bits)
κ	Key length (in bits)
ν	Nonce length (in bits)
l	IV length (in bits)
z	Number of zero-padding bits
$\perp$	Error symbol
π, π^{-1}	Permutation and its inverse in a sponge, respectively
F	Context-processing component
r_F, c_F	Rate & capacity for processing AD, respectively
G	message-processing component
r_G, c_G	Rate & capacity for processing message, respectively
S	Intermediate state (output of F, input to G)
b	Total state size: $b = r_F + c_F = r_G + c_G$
$\hat{F}$, $\check{F}$	Intermediate states during F processing
$\hat{G}$, $\check{G}$	Intermediate states during G processing
γ	Number of message/ciphertext blocks
μ	Size of a multi-collision
ξ	State-update-function
Y, O	Output state and output data from ξ
mulcoll(z, μ)	Cost to find a μ-multicollision on z bits
Pr$[\cdot]$	Probability of an event
$\ln(\cdot)$	Natural logarithm
$\log(\cdot)$	Base-2 logarithm
$\oplus$	Bitwise XOR
$\|$	Concatenation
$\lceil X \rceil_k$	First k bits of X
$\lfloor X \rfloor_k$	Last k bits of X
$\langle X \rangle_n$	n bits binary representation of X
$\ggg$	Bitwise rotation
DoF	Degree of freedom (adversary's controllable input bits)

B Theorem Statements

In this section, we give some of the formal theorem statements for the attacks from Section 3.

B.1 Initial State Attack Results

Theorem 1 (CMT-4 **IS Attack against Duplex).** *Let* Duplex *be the generic duplex sponge AEAD scheme depicted in Fig. 1 and Fig. 3. Let π be modeled as a random permutation. There exists IS-adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ making q queries to π such that*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{Duplex}[0,0,0]}(\mathcal{A}_1) \geq \frac{q}{2^l}\,, \quad \mathbf{Adv}^{\text{CMT-4}}_{\mathsf{Duplex}[\star,\star,\star]}(\mathcal{A}_2) \geq \frac{q^2}{2^{\kappa+2l}}.$$

In addition, the constructions of the form $\mathsf{Duplex}[1,\star,\star]$ *(i.e., the key-blinding includes initial key-blinding), the above result holds only if the rate of the associated data processing to satisfy $r_F \geq \frac{\kappa}{2} + l$.*

Theorem 2 (CMT-4 **IS Attack against Zero-Padded Duplex).** *Let* ZP-Duplex *be the generic duplex sponge AEAD scheme with zero-padding. Let π be modeled as a random permutation. There exists IS-adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ making q queries to π such that*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{ZP\text{-}Duplex}[0,0,0]}(\mathcal{A}_1) \geq \frac{q}{2^l}\,, \quad \mathbf{Adv}^{\text{CMT-4}}_{\mathsf{ZP\text{-}Duplex}[\star,\star,\star]}(\mathcal{A}_2) \geq \frac{q^2}{2^{\kappa+2l}}.$$

In addition, the constructions of the form $\mathsf{Duplex}[1,\star,\star]$ *(i.e., if the key-blinding includes initial key-blinding), the above result holds only if the rate of the associated data processing to satisfy $r_F \geq \frac{\kappa}{2} + l$.*

Theorem 3 (CMT$^\star_{\text{K}}$ **IS Attack against Duplex).** *Let* Duplex *be the generic duplex sponge AEAD scheme depicted in Fig. 1 and Fig. 3. Let π be modeled as a random permutation. Then,*

- *There exists an IS-adversaries $\mathcal{A}_1$ making q queries to π such that*

$$\mathbf{Adv}^{\text{CMT}^\star_{\text{K}}}_{\mathsf{Duplex}[0,0,0]}(\mathcal{A}_1) \geq \frac{q^2}{2^{\nu+2l}}\,,$$

  given that $r_G \geq \nu/2 + l$.
- *There exists an IS-adversaries $\mathcal{A}_2$ making q queries to π such that*

$$\mathbf{Adv}^{\text{CMT}^\star_{\text{K}}}_{\mathsf{Duplex}[1,0,0]}(\mathcal{A}_2) \geq \frac{q^2}{2^{\nu+2\kappa+2l}}\,,$$

  given that $r_G \geq \nu/2 + l + \kappa$.
- *There exists an IS-adversaries $\mathcal{A}_3$ making q queries to π such that*

$$\mathbf{Adv}^{\text{CMT}^\star_{\text{K}}}_{\mathsf{Duplex}[\star,1,0]}(\mathcal{A}_3) \geq \frac{q^2}{2^{\nu+2\kappa+2l}}\,,$$

  given that $r_G \geq \nu/2 + l$.

Theorem 4 ($\mathrm{CMT}^{\star}_{\mathrm{K}}$ **IS Attack against Zero-Padded Duplex).** *Let* ZP-Duplex *be the zero-padded Duplex Sponge AEAD scheme. Let* π *be modeled as a random permutation. Then,*

- *There exists an IS-adversaries* $\mathcal{A}_1$ *making* q *queries to* π *such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}[0,0,0]}(\mathcal{A}_1) \geq \frac{q^2}{2^{\nu+2l}}\,,$$

  given that $r_G - z \geq \nu/2 + l$ *(for prefix) and* $r_G \geq \nu/2 + l$ *(for suffix).*
- *There exists an IS-adversaries* $\mathcal{A}_2$ *making* q *queries to* π *such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}[1,0,0]}(\mathcal{A}_2) \geq \frac{q^2}{2^{\nu+2\kappa+2l}}\,,$$

  given that $r_G - z \geq \nu/2 + l + \kappa$ *(for prefix) and* $r_G \geq \nu/2 + l + \kappa$ *(for suffix).*
- *There exists an IS-adversaries* $\mathcal{A}_3$ *making* q *queries to* π *such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}[\star,1,0]}(\mathcal{A}_3) \geq \frac{q^2}{2^{\nu+2\kappa+2l}}\,,$$

  given that $r_G - z \geq \nu/2 + l$.

B.2 Capacity Collision Attack Results

Theorem 5 (CMT-4 **CC Attack against Duplex).** *Consider the generic duplex sponge AEAD scheme* Duplex *with any key-blinding. Let* π *be modeled as a random permutation. There exists a capacity-collision adversary* $\mathcal{A}$ *making* q *queries such that,*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{Duplex}}(\mathcal{A}) \geq \frac{q^2}{2^{\min(c_F, c_G)}}\,.$$

Theorem 6 (CMT-4 **CC Attack against Zero-Padded Duplex).** *Consider the generic duplex sponge AEAD scheme with zero-padding* ZP-Duplex *with any key-blinding. Let* π *be modeled as a random permutation.*

- *If the construction uses prefix zero-padding, then there exists a capacity-collision adversary* $\mathcal{A}$ *making* q *queries such that*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{ZP\text{-}Duplex}}(\mathcal{A}_1) \geq \frac{q^2}{2^{\min(c_F, c_G + z)}}\,.$$

- *If the construction uses suffix zero-padding, then there exists a capacity-collision adversary* $\mathcal{A}$ *making* q *queries such that*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{ZP\text{-}Duplex}}(\mathcal{A}_2) \geq \frac{q^2}{2^{\min(c_F, c_G)}}\,.$$

Theorem 7 ($\mathrm{CMT}^{\star}_{\mathrm{K}}$ CC Attack against Duplex). *Consider the generic duplex sponge AEAD scheme* $\mathsf{Duplex}[b_1, b_2, b_3]$ *with* (b_1, b_2, b_3) *key-blinding. Let* π *be modeled as a random permutation. There exists a capacity-collision adversary* $\mathcal{A}$ *making* q *queries such that,*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{Duplex}[\star,\star,0]}(\mathcal{A}) \geq \frac{q^2}{2^{c_G}}\,,$$

provided $\kappa \geq c_G/2$.

Theorem 8 ($\mathrm{CMT}^{\star}_{\mathrm{K}}$ CC Attack against Zero-Padded Duplex). *Consider the generic duplex sponge AEAD scheme with zero-padding* $\mathsf{ZP\text{-}Duplex}[b_1, b_2, b_3]$ *with* (b_1, b_2, b_3) *key-blinding. Let* π *be modeled as a random permutation.*

- *If the construction uses prefix zero-padding, then there exists a capacity-collision adversary* $\mathcal{A}_1$ *making* q *queries such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}[\star,\star,0]}(\mathcal{A}_1) \geq \frac{q^2}{2^{c_G+z}}\,,$$

  provided $2\kappa \geq c_G + z$.
- *If the construction uses suffix zero-padding, then*
  - *there exists a capacity-collision adversary* $\mathcal{A}_2$ *making* q *queries such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{Duplex}[\star,0,0]}(\mathcal{A}_2) \geq \frac{q^2}{2^{c_G}}\,,$$

    provided $2\kappa \geq c_G$.
  - *there exists a capacity-collision adversary* $\mathcal{A}_3$ *making* q *queries such that*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}[\star,1,0]}(\mathcal{A}_3) \geq \frac{q^2}{2^{c_G+z}}\,,$$

    provided $2\kappa \geq c_G + z$.

B.3 Tag Collision Attack Results

Theorem 9 (CMT-4 TC Attack against Duplex). *Consider the generic duplex sponge AEAD scheme* Duplex *with any key-blinding. Let* π *be modeled as a random permutation. There exists a tag-collision adversary* $\mathcal{A}$ *making* q *queries such that,*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{Duplex}}(\mathcal{A}) \geq \frac{q^2}{2^{\tau}}.$$

Theorem 10 (CMT-4 TC Attack against Zero-Padded Duplex). *Consider the Generic zero-padded Duplex Sponge AEAD scheme* $\mathsf{ZP\text{-}Duplex}$ *with any key-blinding. Let* π *be modeled as a random permutation. There exists a tag-collision adversary* $\mathcal{A}$ *making* q *queries such that,*

$$\mathbf{Adv}^{\text{CMT-4}}_{\mathsf{ZP\text{-}Duplex}}(\mathcal{A}) \geq \frac{q^2}{2^{\tau+z}}\,.$$

Theorem 11 ($\mathrm{CMT}^{\star}_{\mathrm{K}}$ TC Attack against Duplex). *Consider the generic duplex sponge AEAD scheme* Duplex *with any key-blinding. Let π be modeled as a random permutation. There exists a tag-collision adversary $\mathcal{A}$ making q queries such that,*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{Duplex}}(\mathcal{A}) \geq \frac{q^2}{2^{\tau}}\,,$$

provided $\kappa \geq \tau/2$.

Theorem 12 ($\mathrm{CMT}^{\star}_{\mathrm{K}}$ TC Attack against Zero-Padded Duplex). *Consider the Generic zero-padded Duplex Sponge AEAD scheme* ZP-Duplex *with any key-blinding. Let π be modeled as a random permutation. There exists a tag-collision adversary $\mathcal{A}$ making q queries such that,*

$$\mathbf{Adv}^{\mathrm{CMT}^{\star}_{\mathrm{K}}}_{\mathsf{ZP\text{-}Duplex}}(\mathcal{A}) \geq \frac{q^2}{2^{\tau+z}}\,,$$

provided $2\kappa \geq \tau + z$.

C Additional Plots

In this section we provide more plots illustrating the complexity of our 2S-TC attack described in Section 4. The complexity of the TC attack and the two parts of the 2S-TC attack (2S-TC$_1$ and 2S-TC$_2$) for varying μ are shown in Fig. 6, Fig. 7, and Fig. 8 for $\tau = 32$, $\tau = 64$, and $\tau = 128$, respectively.

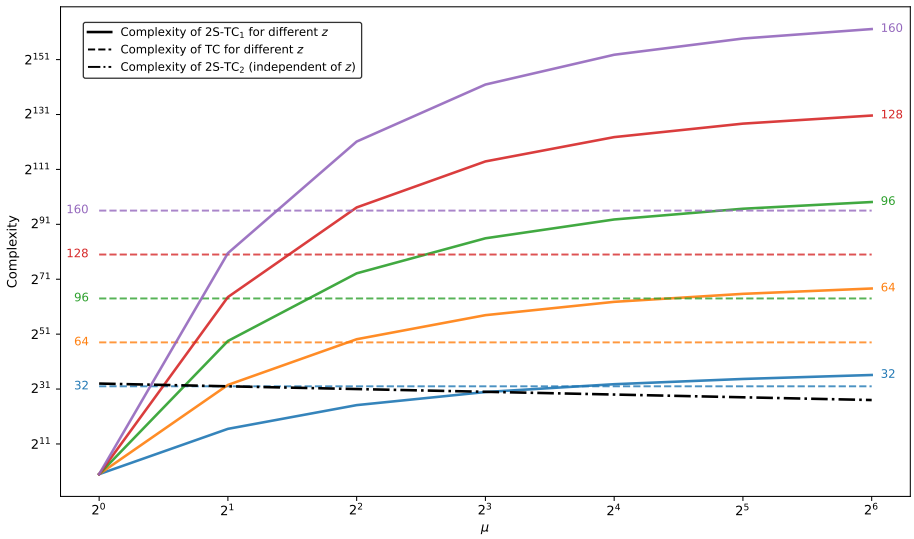

Fig. 6. Complexities of the standard and the two-stage tag-collision attacks for tag length $\tau = 32$ and varying size of the multi-collision μ. Note that the picture contains the plots for various values of z, which are written to the very left or very right of the curves. The two stages of the 2S-TC attack are plotted separately: the first phase (2S-TC$_1$) is depicted as a solid line, while the second phase (2S-TC$_2$) is depicted as a dash-dotted line. Note that the first phase of the attack depends on the value of z, while the second one does not. The dashed lines depict the complexity of the standard attack, which are constant as they do not depend on μ. To minimize the total attack complexity of 2S-TC, the value of μ should be chosen according to the intersection point of the solid and the dash-dotted lines (but at least as 2), which balances the cost of 2S-TC$_1$ and 2S-TC$_2$. If this intersection point lies below the corresponding dashed line, the 2S-TC attack outperforms the TC one.

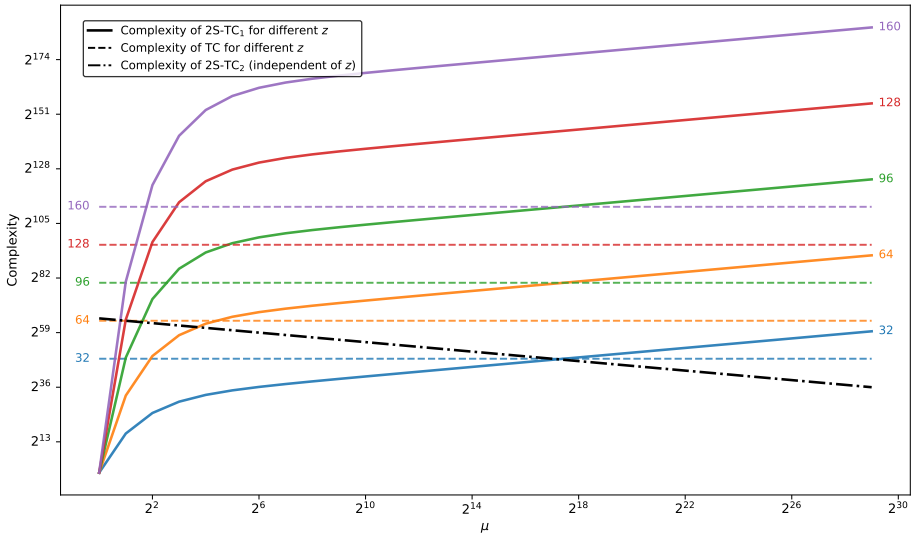

Fig. 7. Complexities of the standard and the two-stage tag-collision attacks for tag length $\tau = 64$ and varying size of the multi-collision μ. For further details, we refer to the caption of Fig. 6.

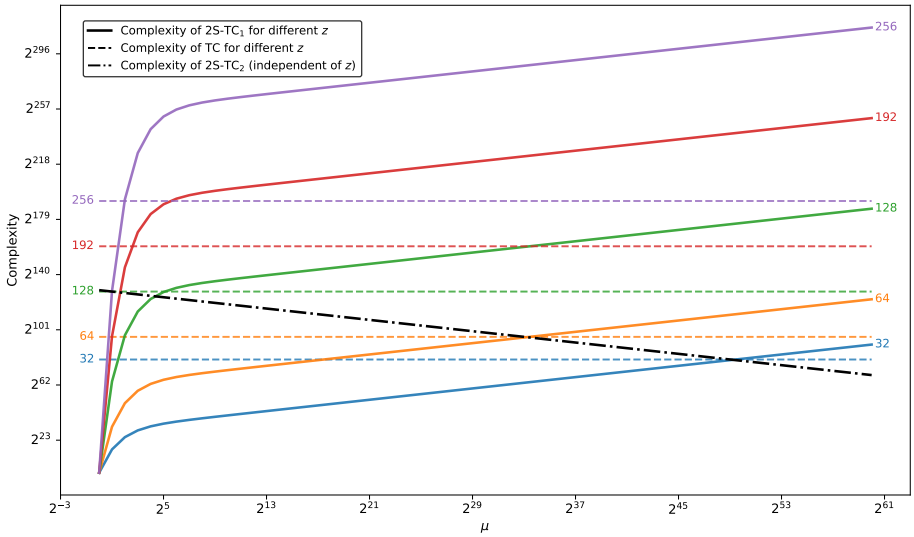

Fig. 8. Complexities of the standard and the two-stage tag-collision attacks for tag length $\tau = 128$ and varying size of the multi-collision μ. For further details, we refer to the caption of Fig. 6.

D Existing Attack Classes for Duplex Sponges with State-Update-Functions

In this section, we extend the committing analysis from plain duplex sponges to ones employing so-called state-update-functions. We provide a description of state-update-functions in Section D.1, and present our CMT-4 results in Section D.2, and the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ ones in Section D.3.

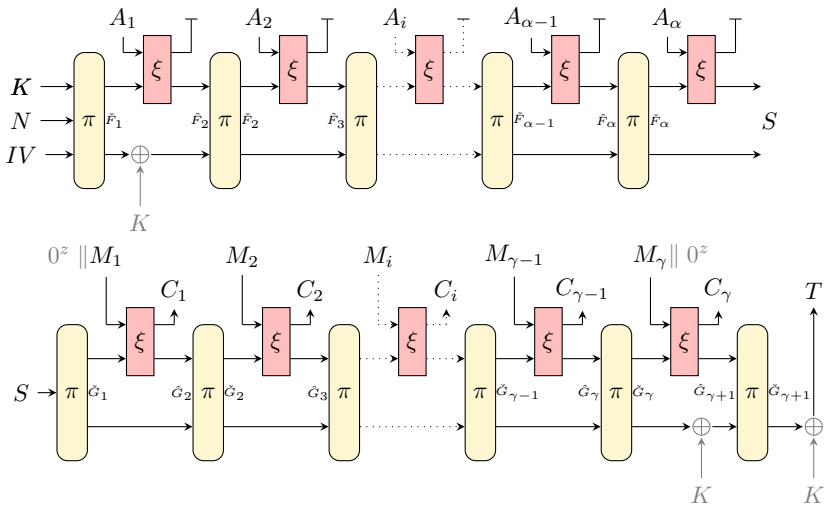

Fig. 9. Illustration of the AEAD scheme based on a state-update-function. The gray parts are optional.

D.1 State-Update-Functions

State-update-functions are a design-feature deployed by multiple of the NIST LWC Finalists. They are applied in alternating manner with the block-cipher or permutation used by the schemes. An illustration for a sponge-based AE scheme using a state-update-function is given in Fig. 9, while the corresponding pseudocode can be found in Fig. 10.

We denote a state-update-functions as

$$\xi(S, I) = (Y, O)\,,$$

where S and Y are the state and the updated state, respectively, and I and O are the input and output, respectively. Note that decryption requires a form of inversion of ξ, however, only with respect to the input and output: given S and O, compute Y and I. In particular, the specifications do not generally describe the full inverse of ξ in the sense $\xi^{-1}(Y, O) = (S, I)$.

$F(K, N, A)$	$G(S, M)$
$A_1, \ldots, A_\alpha \xleftarrow{r_F} A$	$M_1, \ldots, M_\gamma \xleftarrow{r_G} M$
$\check{F}_1 \leftarrow \pi(K \parallel N \parallel IV)$	$\check{G}_1 \leftarrow \pi(S)$
for $i = 1$ **to** $\alpha - 1$	**for** $i = 1$ **to** $\gamma - 1$
$\quad(\hat{F}_{i+1}, \cdot) \leftarrow \xi(\check{F}_i, A_i)$	$\quad(\hat{G}_{i+1}, C_{i+1}) \leftarrow \xi(\check{G}_i, M_i)$
$\quad\check{F}_{i+1} \leftarrow \pi(\hat{F}_{i+1})$	$\quad\check{G}_{i+1} \leftarrow \pi(\hat{G}_{i+1})$
$(S, \cdot) \leftarrow \xi(\check{F}_\alpha, A_\alpha)$	$(\hat{G}_{\gamma+1}, C_\gamma) \leftarrow \xi(\check{G}_\gamma, M_\gamma)$
return S	$\check{G}_{\gamma+1} \leftarrow \pi(\hat{G}_{\gamma+1})$
	$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\gamma \rceil_{\lvert M \rvert}$
	$T \leftarrow \lfloor \check{G}_{\gamma+1} \rfloor_\tau$
	return (C, T)

Fig. 10. Generic construction using a state-update-function ξ.

The NIST LWC finalists follow a common structure, where one of the outputs of ξ is computed as the XOR of the inputs and the other one is of the form $F(S) \oplus I$ for some function F. More precisely, the state-update-functions are defined as

$$\xi(S, I) = (Y, O) = \begin{cases} (S \oplus I, G \cdot S \oplus I) & \textsc{Romulus} \\ ((S_2, S_1 \lll 1) \oplus I, S \oplus I) & \textsc{Gift-Cofb} \\ (S \oplus I, (S_2, S_1 \ggg 1) \oplus I) & \textsc{Photon-Beetle} \\ ((S_2, S_2 \oplus S_1) \oplus I, S \oplus I) & \textsc{Schwaemm} \end{cases}$$

where G is a binary matrix, and S_1, S_2 denote the two halves of S. These state-update-functions can be grouped into two categories.

1. Romulus and Schwaemm have invertible state-update-functions.[17] For Romulus, this was explicitly shown in [16], and for Schwaemm, one can check that $\xi^{-1}(Y, O) := ((O_2 \oplus Y_2, Y_1 \oplus Y_2 \oplus O_1 \oplus O_2), (Y_2 \oplus O_1 \oplus O_2, Y_1 \oplus Y_2 \oplus O_1))$ is the inverse.
2. Gift-Cofb and Photon-Beetle have state-update-functions that are two-to-one, i.e., each element in the image of ξ has exactly two preimages.[18] Since the functions are so similar for the two schemes, we give the argument only for the Photon-Beetle case. Note that $\xi(S, I) = (S \oplus I, (S_2, S_1 \ggg 1) \oplus I)$ is a linear function from the vectorspace $\mathbb{F}_2^r \times \mathbb{F}_2^r$ into itself. An element in the kernel must fulfill $S = I = (S_2, S_1 \ggg 1)$, i.e., $S_1 = S_2 = (S_1 \ggg 1)$.

[17] Note that the approach used by the NIST LWC finalist TinyJambu is equivalent to using the invertible state-update-function $\xi(S, I) = ((S_1 \oplus I, S_2), (S_2 \oplus I))$. An important difference is that the lengths of state and input/output are not the same for TinyJambu (input/output is half as long as the state).

[18] The duplex sponge can be viewed as a 2^r-to-1 function.

Since, next to the all 0 string, the only element invariant under rotation is the all 1 bit string, the kernel has dimension 1. Thus, the dimension of the image of ξ is $2r-1$ and each element in the image has 2 preimages.

In the following analysis, we consider sponges using state-update-functions that are either invertible or 2-to-1.

D.2 CMT-4 Results for Sponges using State-Update-Functions

In Table 8, we provide an overview for the CMT-4 results for sponges using state-update-functions.

Table 8. CMT-4 results for duplex sponge construction with state-update-function. The boxed terms are applicable only for the prefix zero-padding cases. The symbol † indicates that if the construction uses Init KB, the attack is possible only when $r_F \geq \frac{\kappa}{2} + l$.

	Cost			
	IS	CC		TC
		ξ: invertible	ξ: 2-to-1	
Sponge-$\xi[0,0,0]$	l	$\frac{c_F}{2}$	$\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G}{2}\right)$	$\frac{\tau}{2}$
Sponge-$\xi[\star,\star,\star]$	$\frac{\kappa}{2} + l^{\dagger}$	$\frac{c_F}{2}$	$\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G}{2}\right)$	$\frac{\tau}{2}$
ZP-Sponge-$\xi[0,0,0]$	l	$\frac{c_F}{2}$	$\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G + \boxed{z}}{2}\right)$	$\frac{\tau+z}{2}$
ZP-Sponge-$\xi[\star,\star,\star]$	$\frac{\kappa}{2} + l^{\dagger}$	$\frac{c_F}{2}$	$\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G + \boxed{z}}{2}\right)$	$\frac{\tau+z}{2}$

CMT-4 Initial State Attack. The CMT-4 initial state attack works analogously to the duplex case, as we can target a state in the F component, where the outputs O_i of the state-update-function are discarded. More precisely, we can choose the state S and invert F, by choosing the O_i in a way such that the state-update-function can be inverted on the tuple consisting of the rate part of the state and O_i. This is possible, if every state Y appears in the image of the state-update-function—which is the case for the ones used by the NIST LWC finalists. Then, since inversion of the F component is possible, all attacks from the duplex setting apply.

CMT-4 Capacity Collision Attack. The general attack strategy works as in the duplex case; however, based on the type of state-update-function, there are some key differences.

Firstly, if the state-update-function is invertible, we can only target the F component as a capacity collision in the G component cannot be transformed into a full state collision. To do this, one would, given two random states S and S' (namely the rate parts of $\check{G}$ and $\check{G}'$), need to find two message blocks M, M' such that $\xi(S, M) = \xi(S', M')$, which is impossible for an invertible ξ. Hence in this case, the complexity is $\frac{c_F}{2}$.

Secondly, if the state-update-function is 2-to-1, we can again target both the F and G components. However, for G, we have to account for the probability that the problem above can be solved for the concrete state-update-function under consideration. Denote by ς the complexity to find, given two random states S and S' (namely the rate parts of $\check{G}$ and $\check{G}'$), two message blocks M, M' such that $\xi(S, M) = \xi(S', M')$. Then, in this case, the total complexity is $\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G}{2}\right)$.

Example 6. In the case of PHOTON-BEETLE, the state-update-function is defined as $\xi(S, I) = (S \oplus I, (S_2, S_1 \ggg 1) \oplus I)$, where S_1 and S_2 are the two halves of S. We compute the complexity ς for this state-update-function, i.e., given two random states S and S', what is the probability that we can find inputs I, I' such that $\xi(S, I) = \xi(S', I')$? Note that for this equation to hold, also $S \oplus S' = I \oplus I' = (S_2, S_1 \ggg 1) \oplus (S'_2, S'_1 \ggg 1)$ has to hold. In particular, this implies $S_1 \oplus S'_1 = (S_1 \ggg 1) \oplus (S'_1 \ggg 1) = (S_1 \oplus S'_1) \ggg 1$, which holds only if $S_1 \oplus S'_1$ is the all 0 or the all 1 string. Thus, since S and S' are random states, the complexity is roughly $\varsigma = 2^{\frac{r}{2}-1}$.

In the following, we discuss the differences of the attack when the scheme uses key-blinding or zero-padding.

Zero-Padding. As in the duplex case, only prefix zero-padding has an effect on the attack, as one can always target a state *before* the suffix zero-padding—since the states are identical from the target state onwards, suffix zero-padding has no effect. Regarding prefix zero-padding, there is no effect on the attack in the case that ξ is invertible, as we can only target the F component. In the case that a 2-to-1 state-update-function is used, the complexity increases to $\min\left(\frac{c_F}{2}, \varsigma + \frac{c_G+z}{2}\right)$, as the zero-padding requires a z collision of the state. Note that when attacking the G component, we target any state *after* the zero-padding.

Key-Blinding. The attack is not affected by key-blinding. Here, the same reasoning as in the duplex case applies.

CMT-4 Tag Collision Attack. The tag collision attack works as in the duplex setting with the single exception that we need the following property of state-update-functions for the attack to be successful: Given the 2^r possible states $S_i^{(1)}, \ldots, S_i^{(2^r)}$ on r bits, one can find 2^r messages $M_i^{(1)}, \ldots, M_i^{(2^r)}$ s.t. the ciphertexts $C_i^{(j)}$ obtained by $(Y_i^{(j)}, C_i^{(j)}) = \xi(S_i^{(j)}, M_i^{(j)})$ agree for all $j \in \{1, \ldots, 2^r\}$. This property is given for all of the state-update-functions used by the NIST LWC finalists.

Zero-Padding and Key-Blinding. As in the duplex case, both cases of zero-padding increase the complexity to $\frac{\tau+z}{2}$, while key-blinding does not have an effect on the attack.

D.3 $\mathrm{CMT}^{\star}_{\mathrm{K}}$ Results for Sponges using State-Update-Functions

In Table 9, we describe the results for $\mathrm{CMT}^{\star}_{\mathrm{K}}$.

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Initial State Attack. In contrast to the CMT-4 case, the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ initial state attack does not transfer as easily from the duplex setting. As we consider a $\mathrm{CMT}^{\star}_{\mathrm{K}}$ attack, we cannot target the F component anymore, but need to target a state in G instead. While in the duplex setting, one can produce 2^r different states that will agree after XORing the first message block (and thus also yield the same ciphertext block), the same is not true when a state-update-function is used. Since we can only obtain 1 or 2 preimages for a given output, we cannot gather enough states in a single round of message absorption to execute the birthday/preimage attacks against the initial state.

However, we can try to extend the attack to multiple rounds, gathering more states each round. Whether this strategy works depends on the type of state-update-function. If it is invertible, we cannot increase the number of states over multiple rounds, as there is always exactly one preimage under ρ. If the state-update-function is 2-to-1, we can accumulate preimages over multiple rounds: starting with one state $\hat{G}_{i+1}$ and a ciphertext block C_i, we can find two preimages $\check{G}_i, \check{G}'_i$ under ρ;[19] then, applying the inverse permutation, we obtain two different states $\hat{G}_i, \hat{G}'_i$ in the previous round; choosing another ciphertext block C_{i-1}, we invert ρ on $\hat{G}_i$ and $\hat{G}'_i$, each time obtaining two preimages, resulting in a total of four states. Thus, to gather x states, this process has to be repeated over $\log(x)$ rounds. However, note that we can only invert ρ on some state and ciphertext block if they are in the image of ρ. Since the state-update-function is 2-to-1, the size of the image is 2^{2r-1}. In the attack, we obtain random states (by applying the inverse permutation), while the ciphertext blocks can be chosen—however it has to be the same block for all states to obtain a successful committing attack. Thus, we need to account for the probability p_j that, given j states $Y_1, \ldots, Y_j$, one can find a *single* ciphertext block C such that $(Y_1, C), \ldots, (Y_j, C)$ lies in the image of ρ. Then the total success probability of the attack is $\prod_{j=1}^{\log(x)-1} p_j$, which is highly unlikely for most cases of p_j.

The initial state attack is also not possible when considering additional zero-padding or key-blinding.

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Capacity Collision Attack. The argumentation from the CMT-4 attack applies with the difference that we cannot target the F component due to the associated data having to stay the same, which implies that a capacity collision cannot be extended to a full-state collision. Hence, by omitting all terms

[19] We take it as understood that ρ^{-1} is applied only to the rate-part of $\check{G}_i$ and $\check{G}'_i$.

Table 9. $\mathrm{CMT}^{\star}_{\mathrm{K}}$ table for sponge construction with state-update-function. The bounds for CC are only applicable for 2-to-1 state-update-functions; for invertible state-update-functions, the attacks are infeasible. The boxed term is applicable only for the prefix zero-padding. For each attack, the degree of freedom (DoF) is κ, and it must exceed the cost for the attack to be effective.

	Cost		
	IS	CC	TC
Sponge-$\xi[\star,\star,0]$	—	$\varsigma + \frac{c_G}{2}$	$\frac{\tau}{2}$
Sponge-$\xi[\star,\star,1]$	—	—	$\frac{\tau}{2}$
ZP-Sponge-$\xi[\star,\star,0]$	—	$\varsigma + \frac{c_G + \boxed{z}}{2}$	$\frac{\tau+z}{2}$
ZP-Sponge-$\xi[\star,\star,1]$	—	—	$\frac{\tau+z}{2}$

concerning c_F from the CMT-4 case, we obtain the complexities for the $\mathrm{CMT}^{\star}_{\mathrm{K}}$ setting. Further, since the only degree of freedom is the key, the attacks are only possible if the attack complexity is smaller than or equal to κ.

$\mathrm{CMT}^{\star}_{\mathrm{K}}$ Tag Collision Attack. The attack works as in the CMT-4 setting, with the exception that only the key can differ, while the same nonce and associated data have to be used. Thus, the attack is only possible if the complexity is smaller than or equal to κ. Further, we require the same assumption of ξ as in the tag collision attack in the CMT-4 setting.

E Formal Attack Algorithms

The pseudocode for the CMT-4 attack exploiting the IS collision without and with the key-blinding is presented in Algorithm 4 and Algorithm 5, respectively.

Algorithm 4 A CMT-4 Attack exploiting Initial State Collision.

function CMT-4_`IS_Adversary`( )
  Choose an arbitrary context (K, N, A) and a message M
  $S \leftarrow F(K, N, A)$
  for $i = 1$ to q **do**
    Choose a fresh associated data $A^{(i)}$
    $(K^{(i)}, N^{(i)}, IV^{(i)}) \leftarrow F^{-1}(S, A^{(i)})$
    if $IV^{(i)} = IV$ **then** ▷ l-bit
      return $(K, N, A, M), (K^{(i)}, N^{(i)}, A^{(i)}, M)$
  return $\perp$

Algorithm 5 A CMT-4 Attack exploiting IS Collision for KB.

function CMT-4_`IS_Adversary_KB`( )
  Choose an arbitrary state S and an arbitrary message M
  Initialize a list $\mathcal{L} \leftarrow \emptyset$
  for $i = 1$ to q **do**
    Choose a fresh associated data $A^{(i)}$
    $(K^{(i)}, N^{(i)}, IV^{(i)}) \leftarrow F^{-1}(S, A^{(i)})$
    if $IV^{(i)} = IV$ **then** ▷ l-bit
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(K^{(i)}, N^{(i)}, A^{(i)})\}$
  Find distinct $(K^{(i)}, N^{(i)}, A^{(i)}), (K^{(j)}, N^{(j)}, A^{(j)}) \in \mathcal{L}$ with $K^{(i)} = K^{(j)}$. ▷ κ-bit
  return $(K^{(i)}, N^{(i)}, A^{(i)}, M),\ (K^{(j)}, N^{(j)}, A^{(j)}, M)$

The pseudocode for the CMT-4 attack exploiting capacity collision and tag collision is presented in Algorithm 6 and Algorithm 7, respectively.

Algorithm 6 A CMT-4 Attack exploiting Capacity Collision.

function CMT-4_CC_Adversary( )
  if $c_F < c_G$ **then**
    for $i = 1$ to q **do**
      $S^{(i)} \leftarrow F(K^{(i)}, N^{(i)}, A^{(i)})$
    if $\exists i, j \in [q]\colon \lfloor \pi(S^{(i)}) \rfloor_{c_F} = \lfloor \pi(S^{(j)}) \rfloor_{c_F}$ **then**
      return $(K^{(i)}, N^{(i)}, A^{(i)} \parallel 0^{r_F}, M)$, $(K^{(j)}, N^{(j)}, A^{(j)} \parallel \lceil \pi(S^{(i)}) \oplus \pi(S^{(j)}) \rceil_{r_F}, M)$
  else
    for $i = 1$ to q **do**
      $S^{(i)} \leftarrow F(K^{(i)}, N^{(i)}, A^{(i)})$
    if $\exists i, j \in [q]\colon \lfloor S^{(i)} \rfloor_{c_G} = \lfloor S^{(j)} \rfloor_{c_G}$ **then**
      return $(K^{(i)}, N^{(i)}, A^{(i)}, 0^{r_G} \parallel M)$, $(K^{(j)}, N^{(j)}, A^{(j)}, \lceil S^{(i)} \oplus S^{(j)} \rceil_{r_G} \parallel M)$
  return $\perp$

Algorithm 7 A CMT-4 Attack exploiting Tag Collision.

function CMT-4_TC_Adversary( )
  for $i = 1$ to q **do**
    $S^{(i)} \leftarrow F(K^{(i)}, N^{(i)}, A^{(i)})$
    $(C^{(i)}, T^{(i)}) \leftarrow G(S^{(i)}, \lceil S^{(i)} \oplus S^{(1)} \rceil_{r_G})$
  if $\exists i, j \in [q]\colon T^{(i)} = T^{(j)}$ **then**
    return $(K^{(i)}, N^{(i)}, A^{(i)}, \lceil S^{(i)} \oplus S^{(1)} \rceil_{r_G})$, $(K^{(j)}, N^{(j)}, A^{(j)}, \lceil S^{(j)} \oplus S^{(1)} \rceil_{r_G})$
  return $\perp$

F Illustration of Attacks from Section 5

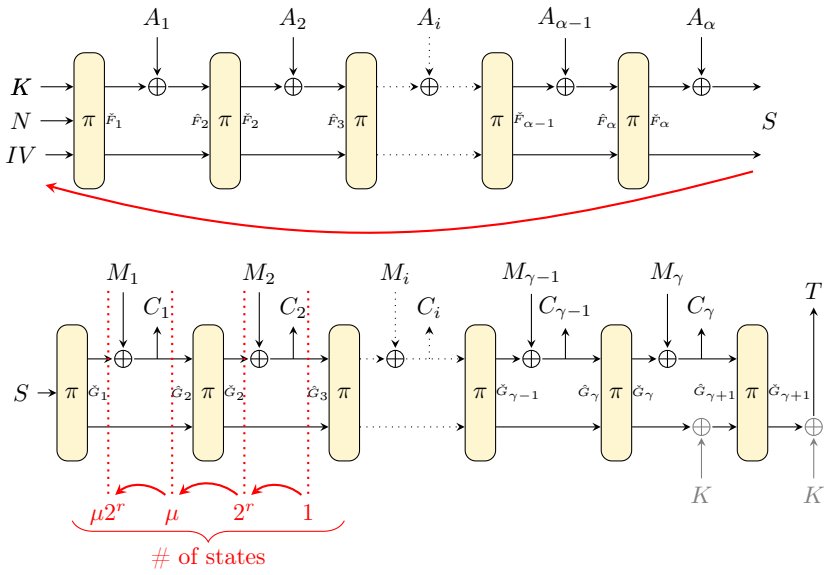

Fig. 11. Illustration of the 2R-IS attack.

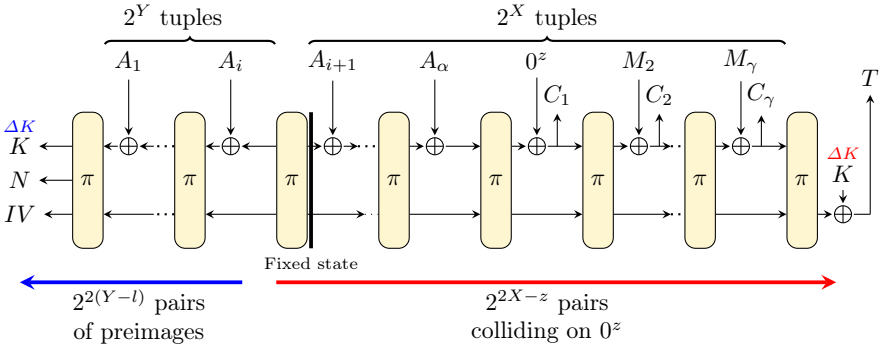

Fig. 12. Illustration of Meet-in-the-Middle Attack.

G Condition that Tag KB Weakens Committing Security

This section explains that under the following two conditions, CMT-4 security of the plain duplex construction *without* Tag KB is higher than that of the construction *with* Tag KB. In other words, by adding tag KB—and no others—to the duplex construction, the attack complexity can be lowered for certain parameter choices.

Condition 1. Prefix zero-padding is adopted, and a full block is occupied by the padded zero bits, i.e., z is a multiple of r_G.

Condition 2. A generated pair satisfies $M \neq M'$, in other words, $\Delta M \neq 0$.

Generic Attack Cost for Plain Duplex. We first show that under the two conditions above, the generic attack cost of the plain duplex construction is $\min(l + \frac{z}{2}, \min\left(\frac{c_F+z}{2}, \frac{c_G+z}{2}\right), \frac{\tau+z}{2})$. In particular, when $c_F = c_G = c$, it becomes

$$\min\Big(l + \frac{z}{2}, \frac{c+z}{2}, \frac{\tau+z}{2}\Big),$$

where the first, second, and third term corresponds to the complexity of the IV attack, CC attack, and TC attack, respectively.

In the plain duplex construction, from Condition 1, at least the first message block M_1 is fully occupied by the padded zero-bits. For simplicity, we assume that the the next message block M_2 has non-zero difference. Under the condition that $\Delta M \neq 0$, at least one of the elements (K, N, A) must have a non-zero difference. Otherwise, ΔM_2 generates a difference in the second ciphertext block C_2, which violates the definition of the committing attack. We then evaluate the complexity of each attack approach.

IV attack. We fix the whole state by fixing C_2 and the corresponding capacity bits. We then choose many values of M_2 and compute backward to the initial state to hit the l-bit IV with a cost of 2^l. In addition, a collision must be generated on the z bits of C_1, hence the above IV-preimage generation is iterated $2^{\frac{z}{2}}$ times, which results in the complexity of $2^{l+\frac{z}{2}}$.

CC attack. A difference from (K, N, A) propagates forward and is canceled with ΔM_2. Besides the capacity collision, the z bits of the state for encrypting the padded zeros must collide, which makes the attack complexity $2^{\frac{c_G+z}{2}}$. We can also start by choosing M_2 and compute backward as performed in the IV attack, and cancel the state difference by a difference of some associated data block. This approach has a complexity of $2^{\frac{c_F+z}{2}}$.

TC attack. A difference from (K, N, A) propagates forward and the collision must occur on the z bits of the state for encrypting the padded zeros and τ bits of the tag. This is the same as the case with $\Delta M = 0$, and the complexity remains unchanged from $2^{\frac{\tau+z}{2}}$.

Generic Attack Cost with Tag KB. The two conditions also impacts to the complexity for the duplex with tag KB. As discussed in Section 5.2, without the two conditions, the CMT-4 security of the tag KB with $\kappa \geq \tau$ is given by

$$\min\Big(l + \frac{\tau}{2}, \frac{c}{2}, \frac{\tau + z}{2}, \frac{\tau + z}{4} + \frac{l}{2}\Big)$$

by ignoring the constant factor and assuming $c = c_F = c_G$, where the first, second, third, and fourth term corresponds to the complexity of the IS, CC, TC, and MitM attack, respectively. As explained for the plain duplex construction, the complexities of IS and CC attacks increase by a factor of $\frac{z}{2}$ bits, and the complexity of TC attack remains unchanged. The complexity of the MitM attack also remains unchanged because it works with any zero-padding position and it satisfies $\Delta M \neq 0$. Hence, the security of tag KB in this setting is given by

$$\min\Big(l + \frac{\tau + z}{2}, \frac{c + z}{2}, \frac{\tau + z}{2}, \frac{\tau + z}{4} + \frac{l}{2}\Big).$$

Compared to the attack complexity for the plain duplex construction, as long as one of the IV, CC, and TC attacks achieves the lowest attack complexity, the security of tag KB is always higher than or equal to that of the plain duplex. However, for parameters where the MitM attack becomes the lowest attack complexity, the security of tag KB can be lower than that of the plain duplex.

Example 7. Suppose that the state size is 512 bits, $r = r_F = r_G = 256$, $c = c_F = c_G = 256$, $l = 128$, $\nu = 128$, $\kappa = 256$, $\tau = 256$, $z = 256$. Then, the complexity for the plain duplex construction is

$$\min(128 + \frac{256}{2}, \frac{256 + 256}{2}, \frac{256 + 256}{2}) = 256\,.$$

Meanwhile, the complexity for the duplex with the tag KB is

$$\min(128 + \frac{256 + 256}{2}, \frac{256 + 256}{2}, \frac{256 + 256}{2}, \frac{256 + 256}{4} + \frac{128}{2}) = 192\,,$$

which is smaller than the complexity for the plain duplex construction.

H Pseudocode of the Meet-in-the-Middle Attack

The pseudocode of our attack is depicted in Fig. 12, where the prefix zero-padding is adopted and the first message block is fully occupied by the encoded zero bits, i.e., $z = r_G$. The attack works for any number of associated data blocks α and message blocks γ as long as sufficient amount of values can be chosen for the associated data blocks.

Algorithm 8 A CMT-4 MitM Attack for Tag KB.

function `MitM_Adversary_KB`( )
  Choose an arbitrary state $\check{F}_{i+1}$ and a ciphertext $(C_2, \ldots, C_\gamma)$
  Initialize lists $\vec{\mathcal{L}} \leftarrow \emptyset$, $\overleftarrow{\mathcal{L}} \leftarrow \emptyset$, $\vec{\mathcal{L}}^{\mathsf{tmp}} \leftarrow \emptyset$, and $\overleftarrow{\mathcal{L}}^{\mathsf{tmp}} \leftarrow \emptyset$

  for $u = 1$ to $2^{\frac{\tau+z}{4}+\frac{l}{2}}$ **do**
    Choose a fresh AD $\vec{A} \coloneqq (A_{i+1}, \ldots, A_\alpha)$
    for $v = i+1$ to $\alpha - 1$ **do**
      $\check{F}_{v+1} \leftarrow \pi(\check{F}_v \oplus (A_v \parallel 0^{c_F}))$
    $\check{G}_1 \leftarrow \pi(\check{F}_\alpha \oplus (A_\alpha \parallel 0^{c_F}))$
    $C_1 \leftarrow \lceil \check{G}_1 \rceil_{r_G}$
    $\check{G}_2 \leftarrow \pi(\check{G}_1)$
    for $v = 2$ to γ **do**
      $M_v \leftarrow \lceil \check{G}_v \rceil_{r_G} \oplus C_v$
      $\check{G}_{v+1} \leftarrow \pi(C_v \parallel \lfloor \check{G}_v \rfloor_{c_G})$
    $M \leftarrow M_2 \parallel \cdots \parallel M_\gamma$
    $\vec{\mathcal{L}}^{\mathsf{tmp}} \leftarrow \vec{\mathcal{L}}^{\mathsf{tmp}} \cup \{(C_1, \lfloor \check{G}_{\gamma+1} \rfloor_\tau, \vec{A}, M)\}$
  for all pairs in $\vec{\mathcal{L}}^{\mathsf{tmp}}$ colliding on C_1 **do**
    $\vec{\mathcal{L}} \leftarrow \vec{\mathcal{L}} \cup \{\vec{A}, \vec{A}', M, M', \Delta \lfloor \check{G}_{\gamma+1} \rfloor_\tau\}$

  for $u = 1$ to $2^{\frac{\tau+z}{4}+\frac{l}{2}}$ **do**
    Choose a fresh AD $\overleftarrow{A} \coloneqq (A_1, \ldots, A_i)$
    $\hat{F}_{i+1} \coloneqq \pi^{-1}(\check{F}_{i+1})$
    for $v = i$ to 1 **do**
      $\hat{F}_v \leftarrow \pi^{-1}(\hat{F}_{v+1} \oplus (A_v \parallel 0^{c_F}))$
      if $\left\lceil \hat{F}_1 \right\rceil_l = IV$ **then**
        $\overleftarrow{\mathcal{L}}^{\mathsf{tmp}} \leftarrow \overleftarrow{\mathcal{L}}^{\mathsf{tmp}} \cup \{(K, N, \overleftarrow{A})\}$
  for all pairs that can be constructed from the elements in $\overleftarrow{\mathcal{L}}^{\mathsf{tmp}}$ **do**
    $\overleftarrow{\mathcal{L}} \leftarrow \overleftarrow{\mathcal{L}} \cup \{K, K', N, N', \overleftarrow{A}, \overleftarrow{A}', \Delta K\}$

  Find a match between $\Delta \lfloor \check{G}_{\gamma+1} \rfloor_\tau$ in $\vec{\mathcal{L}}$ and ΔK in $\overleftarrow{\mathcal{L}}$
  return $(K, N, \overleftarrow{A} \parallel \vec{A}, M), (K', N', \overleftarrow{A}' \parallel \vec{A}', M')$