IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2024, No. 4, pp. 191–248. DOI:10.46586/tosc.v2024.i4.191-248

Committing AE from Sponges

Security Analysis of the NIST LWC Finalists

Juliane Krämer[1], Patrick Struck[2] and Maximiliane Weishäupl[1]

[1] Universität Regensburg, Regensburg, Germany
juliane.kraemer@ur.de maximiliane.weishaeupl@ur.de
[2] Universität Konstanz, Konstanz, Germany
patrick.struck@uni-konstanz.de

Abstract. Committing security has gained considerable attention in the field of authenticated encryption (AE). This can be traced back to a line of recent attacks, which entail that AE schemes used in practice should not only provide confidentiality and authenticity, but also committing security. Roughly speaking, a committing AE scheme guarantees that ciphertexts will decrypt only for one key. Despite the recent research effort in this area, the finalists of the NIST lightweight cryptography standardization process have not been put under consideration yet. We close this gap by providing an analysis of these schemes with respect to their committing security. Despite the structural similarities the finalists exhibit, our results are of a quite heterogeneous nature: We break four of the schemes with effectively no costs, while for two schemes our attacks are costlier, yet still efficient. For the remaining three schemes Isap, Ascon, and (a slightly modified version of) Schwaemm, we give formal security proofs. Our analysis reveals that sponges are well-suited for building committing AE schemes. Furthermore, we show several negative results when applying the zero-padding method to the NIST finalists.

Keywords: Authenticated Encryption · Committing Security · NIST LWC Finalists

1 Introduction

The most fundamental cryptographic concept is symmetric encryption, allowing two parties, Alice and Bob, which share some secret key, to securely exchange messages. The initial goal—and still a cornerstone—is confidentiality which prevents anyone but Alice and Bob from recovering the message from a ciphertext. In modern cryptography, security requirements have been enhanced to also incorporate authenticity [Vau02], which ensures that no third party can produce a ciphertext that Bob would accept as one generated by Alice. On that account, *authenticated encryption* (AE), which encompasses both confidentiality and authenticity was introduced and has, since then, become the gold standard [NIST15, Ber14]. While authenticated encryption has undergone some changes—from probabilistic over IV-based to nonce-based—nowadays, the research community agrees on *authenticated encryption with associated data* as the right approach. Such a scheme generates a ciphertext C by encrypting a message M under a *context* (K, N, A), consisting of a key K, a nonce N, and associated data A. Authenticity should hold for both the associated data and the message, while confidentiality is required only for the message.

The relevance of authenticated encryption is not only reflected by the conducted research, but also by the fact that AE schemes are deployed ubiquitously, e.g., in TLS 1.3 [Res18]. The CAESAR competition for authenticated encryption [Ber14] and the recent NIST lightweight cryptography (LWC) standardization process [NIST15], both called specifically for AE schemes which are deemed secure if they provide both confidentiality and authenticity.

Received: 2024-09-01 Accepted: 2024-11-01 Published: 2024-12-18

However, a series of recent attacks [LGR21, DGRW18, ADG+22] has shown that there still might be attack scenarios not covered by the standard notions. The *Facebook message franking attack* [DGRW18] enabled Alice, a malicious user, to send an offensive or even illegal content to Bob. If Bob tries to report this, it will fail as Facebook will see a harmless content—prepared by Alice as part of the attack—instead. Further examples are the *subscribe with Google attack* [ADG+22] and the *partitioning oracle attack* [LGR21]. The latter allows for more efficient key recovery: The adversary crafts a ciphertext that decrypts validly under multiple keys (for instance a list of leaked keys) and sends it to the recipient; if the recipient rejects the ciphertext, the adversary can rule out all keys which are valid for the sent ciphertext.

In fact, these attacks can all be traced back to the same problem: The existence of ciphertexts that decrypt validly under more than one key. This is neither prevented by confidentiality nor by authenticity, bearing the need for an additional security notion. To this end, committing security [BH22] was defined by requiring each ciphertext to be a commitment to the key ($\mathrm{CMT}_{\mathsf{K}}$) or even to the whole context (CMT). The latter notion is the strongest one and is formalized by the following security game: The adversary outputs two tuples $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$, each consisting of key, nonce, associated data, and message, and wins if their contexts differ, i.e., $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$, and $\textsc{Ae.Enc}(K, N, A, M) = \textsc{Ae.Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$ holds.

The aforementioned attacks demonstrate that the consequences of using non-committing authenticated encryption can be severe. Considering that there are most likely more attacks, which have yet to be discovered, it is important to deal with this problem. One possibility would be to design protocols in such a way that usage of non-committing authenticated encryption does not result in attacks. However, this approach is ill-advised as it requires a separate analysis for each protocol and puts the burden on the protocol designers. A better approach is to prove authenticated encryption schemes to be committing as those can then be used in different protocols without worrying about committing attacks.

To this end, AE schemes used in practice need to be analyzed with respect to committing security. This process has already begun and a number of commonly used AE schemes (GCM, SIV, CCM, EAX, OCB3) have been examined [BH22, MLGR23]. A majority of them were shown to not achieve committing security. Arguably among the most important AE schemes are the finalists of the NIST LWC standardization process. While these schemes have received significant analysis with respect to confidentiality and authenticity [TMC+23], there is barely any research with respect to their committing security [NSS23, DGL23].

While not all of the existing committing attacks are immediately critical in the lightweight setting, the partitioning oracle attack is applicable and can facilitate key recovery on embedded devices, as is classically done via side-channel attacks (described in [SW24]). This highlights the importance of committing security for lightweight AE schemes—in particular if one takes into consideration that more committing attacks that are relevant for such schemes might arise in the future.

1.1 Contribution

In this paper we analyze the committing security of the NIST LWC finalists that are based on (tweakable) block-ciphers or permutations.[1] More precisely, we focus on the authenticated encryption mode of the schemes, while the underlying primitives, i.e., (tweakable) block-ciphers or permutations, are assumed to be ideal. We follow the example of [MLGR23], to define a boundary between committing insecure and secure schemes. The line is drawn at 64-bit security, i.e., a scheme providing at least 64-bit committing security is called secure, while all others are called insecure.

We divide the NIST LWC finalists into two groups: Firstly, Elephant [BCDM21]

[1]This covers all finalists except Grain-128aead [HJM+21] which comes with a dedicated design.

and Isap [DEM+21] follow the *Encrypt-then-MAC* (EtM) paradigm, as they start by encrypting the message and then authenticate the resulting ciphertext alongside the context. Secondly, Romulus [IKM+21], Gift-Cofb [BCI+21], Photon-Beetle [BCD+21], Xoodyak [DHP+21], TinyJambu [WH21], Ascon [DEMS21], and Schwaemm [BBdS+21] share a common structure, in the sense that they first process the context and then the message. We refer to schemes of this type as *Context-pre-Processing* (CpP) schemes.

Surprisingly, even though the NIST finalists show strong structural similarities, our results regarding their committing security are of a very heterogeneous nature. As can be seen in Table 1, the results vary from attacks that require essentially no queries[2] to attacks that are costlier—still using significantly less than 2^{64} queries—and proofs showing about 64-bit committing security. In summary, there are four schemes we break completely, two schemes we break efficiently, and three schemes[3] for which we show committing security.

Several of our attacks share the same idea. This is the case for Romulus and Gift-Cofb, which are both block-cipher-based AE schemes and feature a state-update-function, which is invoked in an alternating manner with the block-cipher. The attacks boil down to the fact that for a fixed ciphertext, key, and nonce, one can find associated data such that the ciphertext decrypts validly under this context. For this, starting from the target ciphertext, the component that processes the message is inverted. Then the fact that associated data blocks are XORed onto the whole state is used to connect the initial state with the state obtained from the reverse computation. This attack strategy depends heavily on the invertibility of the state-update-function. For Romulus, we show that such an inversion is always possible, while for Gift-Cofb it works with a probability of $\frac{1}{2}$. This implies that the attack cost for Gift-Cofb depends on the length of the ciphertext, as we need to invert the state-update-function for each ciphertext block and the attack only works if *all* are invertible. However, by choosing a short ciphertext we obtain a very efficient attack. The XORing of an input onto the whole state is a vulnerability that is also exploited in our attack on Elephant, a permutation-based AE scheme. In contrast to Romulus and Gift-Cofb, Elephant is an Encrypt-then-MAC scheme. This structure simplifies the committing attack as we only need to find two different contexts that verify the ciphertext correctly—in this case the decryption of Elephant will never return $\perp$.[4] Due to this, it suffices to concentrate on the MAC and, more precisely, finding a tag collision. The latter is easily achieved, as the associated data is XORed to the full state during the tag generation of Elephant.

Except for these three schemes, none of the other NIST finalists carry out a full-state XOR. Xoodyak arguably comes very close, as it is a full-state sponge, which reserves only a few bits for padding, which are not directly accessible via the inputs. Therefore, we are able to control most of the state by a direct XOR, while for the remaining bits a birthday attack is applied. Similarly, the attack on TinyJambu, a block-cipher-based scheme, also boils down to a birthday attack. We exploit that TinyJambu uses a tag of just 64 bits (the shortest one among all considered schemes[5]), hence colliding tags can be found with reasonable cost.

Our attack on Photon-Beetle, a sponge-based AE scheme, exploits the choice of the initial state. For most of the finalists, this state contains some fixed initialization vector, whereas for Photon-Beetle it consists exclusively of key and nonce. However, this implies that the initial state can be controlled completely by a committing adversary, which will turn out to be the key ingredient of our attack. Simply speaking, the attack allows to choose an intermediate state (outcome of the context-pre-processing) that results in the same ciphertext. We can invert this intermediate state for different associated data

[2] More precisely, these attacks need only the minimal cost of computing the respective encryption algorithm twice (once for each of the output tuples).

[3] Note that we consider a slightly modified version of Schwaemm.

[4] Menda et al. [MLGR23] coin this property as *NoFailDecrypt*.

[5] Elephant uses 64-bit tags as well, but also gives a parameter set with 128-bit tags.

and take the outcome as the key-nonce pair.

None of these attacks are applicable to any of the sponge-based schemes Isap, Ascon, and Schwaemm. We give security proofs for these schemes, showing that they achieve about 64-bit committing security. The high-level idea of all proofs is similar: we show that the schemes can be viewed as plain sponge constructions and give bounds for finding colliding tags, which directly translate to bounds on the committing security. Extra care is necessary when dealing with the core features of the schemes—the re-keying mechanism deployed in Isap and the state-/output-blinding applied in both Ascon and Schwaemm.

We can make the following observations for the committing security of a scheme. A too short tag enables efficient birthday attacks that might extend to committing attacks, as—depending on the scheme—colliding ciphertexts might be easy to find. Further, our analysis reveals that the size of the state that cannot directly be influenced by the inputs (for sponge-based schemes, this is the capacity) plays a crucial role regarding the committing security of a scheme. If the size of this "input-independent" state is too small (even if that happens only once in the scheme), a CMT adversary can produce a full-state collision with only a few tries, which in almost all cases we considered, enables an efficient attack. In fact, a variation of this concept is used in all of our attacks except for the one on TinyJambu, which exploits its short tag length. This weakness can also be present in the initial state, i.e., if it contains only a small fixed part, while the rest is filled with inputs provided by the CMT adversary. Note that for the NIST finalists the aforementioned weakness is only present in some of the sponge-based schemes. From this, we can derive some explicit criteria. The schemes we prove committing secure fulfill all of these criteria, while the ones proven insecure lack at least one of them. This suggests that these properties might be sufficient to achieve CMT security—however, we want to emphasize that the criteria should only be seen as a heuristic, as they are observations from the analysis of the schemes, not proven statements. Taking into account our convention of 64-bit committing security, the three criteria are: (I) the tag length is at least 128 bits, (II) at least 128 bits of the state are unaffected by the inputs, and—solely for sponges—(III) the initial state contains at least 64 bits that are independent of the inputs. While we observe that the absence of these properties often enables committing attacks, this is not necessarily the case as the example of Schwaemm shows: The unmodified version does not fulfill (III), as the initial state is completely determined by key and nonce—however, the output-blinding deployed in Schwaemm prevents a CMT attack.

Given the many negative results, we continue with the question whether the NIST finalists can be patched easily to achieve committing security. A very simple method—especially to preserve the lightweight aspect—is the zero-padding approach [ADG+22]. However, we show several scheme-specific negative results regarding the NIST finalists when applying zero-padding. This implies that some of the finalists require the application of costlier transforms to achieve committing security.

In conclusion, our analysis shows that most NIST LWC finalists do not achieve committing security (some even after being zero-padded) and those that do, are all sponge-based.

1.2 Related Work

Committing security can be traced back to [ABN10, FLPQ13] where the focus was on public-key encryption. In [FOR17]—using the name *key-robustness*—Farshim et al. gave first definitions of committing security for symmetric encryption. Recently, Bellare and Hoang [BH22] introduced different variants of committing security for authenticated encryption, covering the prior variants where a ciphertext is a commitment to the key, but also stronger forms where a ciphertext is a commitment to all inputs. Ultimately, Menda et al. [MLGR23] developed a framework for fine-grained committing security notions. Instead of just having a ciphertext being a commitment to either the key or all inputs, it allows for variants where it is a commitment to, say, the key and the nonce. Along with

Table 1: Overview of results: a ✗ indicates a committing (CMT) attack with essentially no queries; a ✦ indicates a CMT attack with significantly less than 2^{64} queries; and a ✓ indicates about 64-bit CMT security. Further, the table depicts three properties (I), (II), and (III), that are relevant for the committing analysis: (I) the tag length is at least 128 bits, (II) at least 128 bits of the state are unaffected by the inputs, and (III) at least 64 bits of the initial state are independent of the inputs. Note that we consider property (III) only for sponge-based schemes. The letters **y** and **n** indicate that the property is and is not, respectively, present in the scheme. If the letter is red, we exploit the respective property in our committing attack.

		Properties				
Scheme	CMT	(I)	(II)	(III)	Sponge	Section
Romulus	✗	**y**	**n**	-	**n**	Section 3.1
Elephant	✗	**n**[a]	**n**	-	**n**	Appendix B.1
GIFT-COFB	✗	**y**	**n**	-	**n**	Appendix B.2
Photon-Beetle	✗	**y**	**y**	**n**	**y**	Appendix B.3
TinyJambu	✦	**n**	**n**	**y**	**y**[b]	Section 3.2
Xoodyak	✦	**y**	**n**	**y**	**y**	Appendix B.4
Ascon	✓	**y**	**y**	**y**	**y**	Section 3.3
Isap	✓	**y**	**y**	**y**	**y**	Appendix B.5
Schwaemm	✓	**y**	**y**	**y**[c]	**y**	Appendix B.6

[a] This is only the case for two out of three parameter sets, including the main one.
[b] TinyJambu can be viewed as a sponge construction based on a block-cipher.
[c] This holds for our slightly modified version $\text{Schwaemm}_{\text{IV}}$, but not the original one.

these committing notions, they also coin the term *context discovery attacks*. In contrast to committing attacks, which require the adversary to find two contexts that decrypt the same ciphertext, context discovery attacks require finding a context that decrypts a given ciphertext. Concurrently to [MLGR23], Chan and Rogaway [CR22] also developed a fine-grained definitional framework for committing security, for instance, allowing for variants where the adversary has to use honest keys, i.e., randomly sampled ones.

In concurrent and independent work, Naito et al. [NSS23] study the committing security of Ascon. While we aim at analyzing all NIST finalists, the focal point of their work is on giving an exhaustive analysis of Ascon. They study both the mode and the underlying permutation of Ascon, whereas we focus our analysis on the mode. Further, they analyze how committing security can be increased by zero-padding the message. The results on the committing security of the unmodified Ascon mode, treated in both works, agree.

2 Authenticated Encryption and the NIST LWC Finalists

In this section, we introduce the notation and recall important definitions. We then provide a general classification of the NIST LWC finalists and high-level approaches for the committing attacks.

2.1 Notation

Throughout this work, we write $\{0,1\}^*$ for the set of bit strings with arbitrary length. By $\{0,1\}^{\leq r}$ ($\{0,1\}^{\geq r}$) we denote the set of bit string with length at most r (at least r). For a bit string S of length n, we write $\lceil S \rceil_r$, $\lfloor S \rfloor_c$, and $[S]_i^j$ for the first r bits, the last c bits, and the i-th to j-th bits of S, respectively. For bit strings X, Y, and Z, $|X|$ describes the

Game CMT (CMT-3 in [BH22])

$$
\begin{array}{ll}
1: & (K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}() \\
2: & \textbf{if } (K, N, A) = (\overline{K}, \overline{N}, \overline{A}) \\
3: & \quad \textbf{return } 0 \\
4: & (C, T) \leftarrow \textsc{Enc}(K, N, A, M) \\
5: & (\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) \\
6: & \textbf{return } ((C, T) = (\overline{C}, \overline{T}))
\end{array}
$$

Figure 1: Security game CMT.

length of X and $Y \parallel Z$ denotes the concatenation of Y and Z. For an integer k, the set $\{1, \ldots, k\}$ is written as $[k]$. We write $X_1, \ldots, X_l \xleftarrow{r} X$ to denote that X is split into bit strings X_1 to X_l s.t. $|X_i| = r$, for $i \in [l-1]$ and $|X_l| \leq r$. Bit rotation resp. bit shift of x by b bits to the left is written as $x \lll b$ resp. $x \ll b$ ($\ggg$ resp. $\gg$ denote the same in the other direction). The encoding of x into one byte is described by $\mathtt{enc}_8(x)$. For sake of simplicity, we use ι as a generic value for domain separation in several schemes as our results are independent of it. Standard cryptographic background on sponges, block-ciphers (BC), and tweakable block-ciphers (TBC) as well as some results needed for our proofs are given in Appendix A.

2.2 Definitions

We recall the definitions of authenticated encryption and committing security.

Definition 1. An authenticated encryption (AE) scheme with associated data is a pair of two algorithms $(\textsc{Enc}, \textsc{Dec})$ such that

- $\textsc{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$ takes a key K, a nonce N, associated data A, and a message M as input and outputs a ciphertext (C, T).
- $\textsc{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\bot\}$ takes a key K, a nonce N, associated data A, and a ciphertext (C, T) as input and outputs a message M or $\bot$.

The sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{M}$, and $\mathcal{C}$ denote the key space, nonce space, associated data space, message space, and ciphertext space, respectively. Throughout this work, we consider these sets to be bit strings of certain length, more precisely, $\mathcal{K} = \{0,1\}^\kappa$, $\mathcal{N} = \{0,1\}^\nu$, $\mathcal{A} = \{0,1\}^*$, $\mathcal{M} = \{0,1\}^*$, and $\mathcal{C} = \{0,1\}^* \times \{0,1\}^\tau$. An AE scheme is called *correct*, if $\textsc{Dec}(K, N, A, \textsc{Enc}(K, N, A, M)) = M$, for any (K, N, A, M). We note further that all considered schemes are *tidy* [NRS14], i.e., $M = \textsc{Dec}(K, N, A, C)$ implies that $C = \textsc{Enc}(K, N, A, M)$. Following [MLGR23], we call the triple (K, N, A) a *context*.

Simply speaking, committing security requires the adversary to find two context-message pairs that encrypt to the same ciphertext. We recall some weaker forms in Appendix A.

Definition 2. Let $\textsc{Ae} = (\textsc{Enc}, \textsc{Dec})$ be an authenticated encryption scheme and the game CMT be defined as in Fig. 1. For any adversary $\mathcal{A}$, its CMT advantage is defined as

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Ae}}(\mathcal{A}) \coloneqq \Pr[\mathsf{CMT}(\mathcal{A}) \to 1]\,.$$

2.3 NIST LWC Finalists

The NIST LWC standardization process [NIST15] required the submitted AE schemes to achieve the well-established notions of confidentiality and authenticity. For the former, the

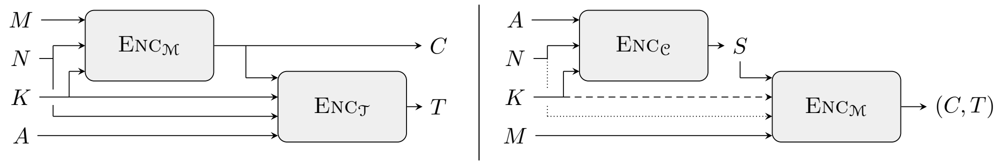


Figure 2: Illustration of Encrypt-then-MAC schemes (left) and Context-pre-Processing schemes (right). Both ELEPHANT and ISAP are Encrypt-then-MAC schemes. The other schemes are Context-pre-Processing schemes. The dotted/dashed arrows indicate that only some of the analyzed schemes exhibit these dependencies: PHOTON-BEETLE and XOODYAK have neither of the two; GIFT-COFB, TINYJAMBU, ASCON, and SCHWAEMM have only the dashed arrow; and ROMULUS has both arrows.

requirement was to maintain security as long as nonces are unique—security in case of repeating nonces can be mentioned as a special feature. Committing security is neither mentioned as a requirement nor a feature to be advertised. However, it is important to note that the call for algorithms was published the same year as the first attack [DGRW18] that exploited the absence of committing security. Due to the more recent research in this area, it can be expected that committing security will either become a requirement or at least a feature considered relevant for cryptographic standards.[6]

The AE schemes that we study in this work are the NIST LWC finalists that are based on (tweakable) block-ciphers or permutations.[7] Details on the classification and parameters of the schemes can be found in Appendix A.1—note that for each candidate we focus on the main parameter set.

2.3.1 Classes of AE Schemes.

The considered schemes can be divided into two classes. The first class encompasses AE schemes that follow the *Encrypt-then-MAC* (EtM) paradigm [BN00]. These schemes first encrypt the message and subsequently authenticate the resulting ciphertext alongside the nonce and the associated data. The second class comprises AE schemes that follow what we call *Context-pre-Processing* (CpP). These schemes first process the context (K, N, A) via a function $\textsc{Enc}_{\mathcal{C}}$. The result is then processed together with M, and optionally K and N, yielding the ciphertext (C, T) via a function $\textsc{Enc}_{\mathcal{M}}$. Both classes are illustrated in Fig. 2. Out of the schemes that we analyze in this work, ELEPHANT and ISAP follow the EtM paradigm, whereas the others—ROMULUS, PHOTON-BEETLE, GIFT-COFB, XOODYAK, TINYJAMBU, ASCON, and SCHWAEMM—follow the CpP-approach.

Attacking Encrypt-then-MAC Schemes. For the EtM schemes, we can focus on the underlying MAC. Once we have two contexts $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$ that verify the same ciphertext (C, T), we can immediately derive a committing attack. This is the case because for the described contexts, the decryption algorithm will return some messages $M, \overline{M} \neq \perp$.[8] Using the tidyness property, we get $\textsc{Enc}(K, N, A, M) = (C, T) = \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$, hence winning the game CMT.

Attacking Context-pre-Processing Schemes. For the CpP schemes, we focus on the state S that is outputted by $\textsc{Enc}_{\mathcal{C}}$ and then fed into $\textsc{Enc}_{\mathcal{M}}$. The general idea is to generate the first context (K, N, A) and a message M at random, and compute the corresponding

[6] A recent NIST workshop [NIST23] mentions committing security as a desirable property.

[7] This covers all finalists except GRAIN-128AEAD [HJM+21].

[8] For both ELEPHANT and ISAP, the underlying decryption algorithm never returns $\perp$, thus the AE scheme returns $\perp$ iff the verification of the tag fails.

ciphertext (C, T). Then, we invert $\textsc{Enc}_{\mathcal{M}}$ for the same ciphertext (C, T) and—depending on the scheme—a different key $\overline{K}$ and nonce $\overline{N}$, which yields the state $\overline{S}$ (along with the message $\overline{M}$). In the last step, we find associated data $\overline{A}$ such that $\textsc{Enc}_{\mathcal{C}}$ with input $(\overline{K}, \overline{N}, \overline{A})$ results in $\overline{S}$, which ultimately yields a committing attack as $\textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) = \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, \textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A}), \overline{M}) = \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, \overline{S}, \overline{M}) = (C, T)$. The step of finding $\overline{A}$ is essentially what was recently coined a *context discovery attack* [MLGR23]. This is a stronger attack that easily translates to a committing attack as shown in [MLGR23]. Indeed, our attacks against Romulus, Gift-Cofb, Elephant, and Photon-Beetle can easily be translated into context discovery attacks; for the other committing attacks this is not the case.

2.3.2 State-Update-Function.

Out of the nine finalists, Romulus, Gift-Cofb, Photon-Beetle, and Schwaemm deploy a so-called *state-update-function* (this name is adopted from [IKM+21]). This function—which we will generally denote by ξ—takes as input a state S and some additional input data I, and outputs a new state Y and additional output data O. The state-update-functions works very similar for all four schemes: one of the outputs is the XOR of the inputs whereas the other is the XOR of the input data I and some underlying function—which depends on the respective scheme—applied to the input states. Details are given in Appendix A.1.

Typically, the state-update-function is used to process the associated data and the message: The current state is used as the input state S while the associated data or the message—more precisely, a block of it—is used as the input data I. The output state Y is used as the new state while the output data O yields the ciphertext or is simply discarded when the associated data is processed. For decryption, the schemes use the inverse of ξ. Here it is important to note that, inverse is to be understood *only* in relation to the output data, i.e., for any (S, I), $\xi(S, I) = (Y, O) \Rightarrow \xi^{-1}(S, O) = (Y, I)$.

For our attacks against Romulus and Gift-Cofb, we need to invert the state-update-function with respect to *both* outputs, which is not obviously possible from the specifications. Our attack against Photon-Beetle is independent of the used state-update-function and for Schwaemm the state-update-function is incorporated into our security proof.

2.3.3 Achieving Committing Security via Transformations.

There are several transformations that turn an arbitrary AE scheme into one that is committing. Clearly, such transformations can be applied to the NIST LWC finalists to make them committing. However, there are several reasons against this: Firstly, these transformations often do not achieve CMT security as we target here but weaker notions [MLGR23]. Secondly, these transformations impose some overhead which—especially considering the lightweight aspect of these schemes—might render them impractical. Thirdly, consider, say, Isap, which comes with a formal security proof incorporating side-channel leakage. Since none of the transformations are analyzed w.r.t. side-channel leakage, applying them to Isap can render the leakage security guarantees obsolete.[9]

3 Committing Security Analysis

Here, we analyze the CMT security of the NIST LWC finalists. For Romulus (cf. Section 3.1) we give an attack that breaks committing security with essentially no cost—requiring the bare minimum of two encryptions. For TinyJambu (cf. Section 3.2), we

[9]Very recently, Struck and Weishäupl [SW24] developed a generic transformation that turns an AE scheme into one that is both committing and leakage-resilient.

provide a committing attack requiring about 2^{33} queries. For Ascon (cf. Section 3.3) we give a formal proof showing that the scheme achieves committing security of about 64-bit. These three schemes represent the different kind of results we have for the NIST LWC finalists and illustrate the core ideas. At the end of each section, we briefly discuss the remaining schemes with similar results and a formal analysis is given in Appendix B.

Our analysis covers all of the NIST LWC finalists except Grain-128aead, which we exclude due to our focus on schemes based on (tweakable) block-ciphers and permutations. While it is easy to see that the committing security of Grain-128aead is upper-bound at 32-bit (as there is only one parameter set featuring a 64-bit tag), developing more efficient attacks requires a dedicated analysis. In particular, the so-called accumulator used in Grain-128aead for tag generation has to be analyzed in detail: Since Grain-128aead is a stream cipher, achieving the same ciphertexts can be easily done, while the more challenging part seems to be forcing same tags. The accumulator gathers message, associated data, and a pseudorandom bit string into the final tag; it is however no ideal primitive, i.e., a concrete analysis is necessary to find better committing attacks (or exclude the existence of such).

3.1 Romulus

Romulus [IKM+21, IKMP20] is an authenticated encryption scheme based on tweakable block-ciphers. For the concrete instantiation of the TBC, they use Skinny [BJK+16] and the authenticated encryption mode bears similarities with Cofb [CIMN17]. Romulus comes in three different variants Romulus-N, Romulus-M, and Romulus-T. The former is the main candidate while the other two are designed with additional security guarantees in mind: Romulus-M achieves security against nonce-misuse while Romulus-T is designed to maintain security even in the presence of side-channel leakage. Throughout this work we only consider the main variant Romulus-N, which we simply refer to as Romulus.

3.1.1 Description of Romulus

The pseudocode of Romulus is given in Fig. 4. The scheme is further illustrated in Fig. 3. It follows the CpP-approach, i.e., it first computes $S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$ and afterwards $(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, S, M)$. Both $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$ apply the tweakable block-cipher and the state-update-function ξ in an alternating manner. The latter, i.e.,

$$\xi : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n, \ \xi(S, I) = (S \oplus I, G(S) \oplus I)$$

is an important component of Romulus and the matrix G it utilizes, is

$$G = \begin{pmatrix} G_s & 0 & 0 & \cdots & 0 \\ 0 & G_s & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & G_s & 0 \\ 0 & \cdots & 0 & 0 & G_s \end{pmatrix}, \text{ where } G_s = \begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&1 \end{pmatrix}.$$

3.1.2 Committing Attack Against Romulus

We show that Romulus does not achieve CMT security. The attack is stated in the following theorem.

Theorem 1. *Consider* Romulus *which is illustrated and described in Fig. 3 and Fig. 4, respectively. Let* TBC *be modeled as an ideal tweakable cipher* $\widetilde{\mathsf{E}}$. *Then there exists an*

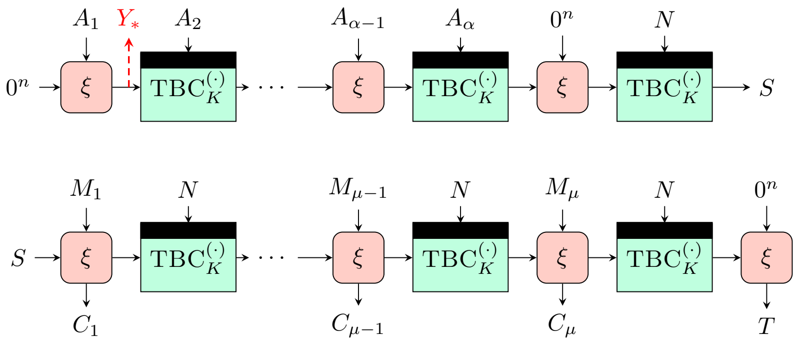


Figure 3: Illustration of Romulus (for α an even number) in terms of $\textsc{Enc}_{\mathcal{C}}$ (top) and $\textsc{Enc}_{\mathcal{M}}$ (bottom). The values that are input from the top into $\mathrm{TBC}_K^{(\cdot)}$ are used as tweaks (for simplicity, we drop the counters making the tweaks unique). The state Y_*, marked in red, is used in our CMT attack.

adversary $\mathcal{A}$, making q queries to $\widetilde{\mathsf{E}}$, such that

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Romulus}}(\mathcal{A}) = 1\,,$$

where $q = 2\mu + \left\lfloor \frac{\alpha}{2} \right\rfloor + \left\lfloor \frac{\overline{\alpha}}{2} \right\rfloor + 2$. Here, μ is the number of message blocks, α is the number of associated data blocks of the first tuple, and $\overline{\alpha}$ is the number of blocks for the second tuple that $\mathcal{A}$ outputs.

For the proof of Theorem 1, we formulate and prove three lemmas. Firstly, we show that the state-update-function ξ is invertible (Lemma 1). Secondly, we prove that we can invert both $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$ (Lemma 2 and Lemma 3), where, for the latter, we make use of the invertibility of ξ.

Recall that the state-update-function ξ of Romulus maps a state S and an input I to a new state Y and an output O. In Romulus.Dec, the inverse of ξ is considered, however, inverse is understood only in relation to the output data. This means that the inverse function will *not* invert the state. When looking at Romulus, one can see that the output of ξ is discarded in $\textsc{Enc}_{\mathcal{C}}$—a fact that will be exploited later. For our attack against $\textsc{Enc}_{\mathcal{M}}$, this no longer works, as we have to invert the output of ξ while maintaining equal ciphertexts. The following lemma shows that we can invert ξ with respect to *both* its output and state. We write M for the input and C for the output of ξ (instead of I and O), which is the case for our scenario.

Lemma 1. *The state-update-function of* Romulus *is invertible.*

Proof. Note that ξ can be expressed as the block-matrix $\left(\begin{array}{c|c} \mathsf{id} & \mathsf{id} \\ \hline G & \mathsf{id} \end{array}\right)$ and hence in order to show the claim, we need to find a right-sided inverse to this matrix. For $+$ denoting component-wise addition mod 2, consider $\left(\begin{array}{c|c} F & F \\ \hline F + \mathsf{id} & F \end{array}\right)$,

$$\text{with } F = \begin{pmatrix} F_s & 0 & 0 & \cdots & 0 \\ 0 & F_s & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & F_s & 0 \\ 0 & \cdots & 0 & 0 & F_s \end{pmatrix} \text{ and } F_s = \begin{pmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&1 \\ 1&1&0&0&0&0&0&1 \\ 1&1&1&0&0&0&0&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&1 \\ 1&1&1&1&1&1&0&1 \\ 1&1&1&1&1&1&1&1 \end{pmatrix}.$$

$\textsc{Romulus}.\textsc{Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, S, M)$
$\textbf{return } (C, T)$

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$A_1, \dots, A_\alpha \xleftarrow{n} \mathtt{pad}_{\mathrm{L}}(A, n)$
$S \leftarrow 0^n$
$\textbf{for } i = 1, \dots, \left\lfloor \frac{\alpha}{2} \right\rfloor$
$\quad (Y, \cdot) \leftarrow \xi(S, A_{2i-1})$
$\quad S \leftarrow \mathrm{TBC}^{A_{2i}}(K, Y)$
$V \leftarrow 0^n$
$\textbf{if } \alpha \bmod 2 \neq 0$
$\quad V \leftarrow A_\alpha$
$(Y, \cdot) \leftarrow \xi(S, V)$
$S \leftarrow \mathrm{TBC}^{N}(K, Y)$
$\textbf{return } S$

$\textsc{Enc}_{\mathcal{M}}(K, N, S, M)$

$M_1, \dots, M_\mu \xleftarrow{n} \mathtt{pad}_{\mathrm{L}}(M, n)$
$\textbf{for } i = 1, \dots, \mu - 1$
$\quad (Y, C_i) \leftarrow \xi(S, M_i)$
$\quad S \leftarrow \mathrm{TBC}^{N}(K, Y)$
$(Y, C_\mu) \leftarrow \xi(S, M_\mu)$
$S \leftarrow \mathrm{TBC}^{N}(K, Y)$
$(\cdot, O) \leftarrow \xi(S, 0^n)$
$T \leftarrow \lceil O \rceil_\tau$
$C \leftarrow \lceil C_1 \parallel \dots \parallel C_\mu \rceil_{|M|}$
$\textbf{return } (C, T)$

$\xi(S, I)$

$Y \leftarrow S \oplus I$
$O \leftarrow G(S) \oplus I$
$\textbf{return } (Y, O)$

Figure 4: Pseudocode of Romulus [IKM+21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$. For sake of simplicity, we drop the counter that is part of the tweak.

Observe that

$$\left(\begin{array}{c|c} \mathsf{id} & \mathsf{id} \\ G & \mathsf{id} \end{array}\right) \cdot \left(\begin{array}{c|c} F & F \\ F + \mathsf{id} & F \end{array}\right) = \left(\begin{array}{c|c} F + F + \mathsf{id} & F + F \\ GF + F + \mathsf{id} & GF + F \end{array}\right) = \left(\begin{array}{c|c} \mathsf{id} & 0 \\ 0 & \mathsf{id} \end{array}\right),$$

where the last equality follows from a direct computation which shows that $G_s F_s + F_s = F_s(G_s + \mathsf{id}) = \mathsf{id}$ and thus $GF + F = F_s(G + \mathsf{id}) = \mathsf{id}$. □

Next, we give an adversary that inverts $\textsc{Enc}_{\mathcal{C}}$, i.e., for a given output S of $\textsc{Enc}_{\mathcal{C}}$ and a partial context (K, N), it finds matching associated data. We exploit the fact that the associated data blocks are XORed to the full state in $\textsc{Enc}_{\mathcal{C}}$. The detailed proof can be found in Appendix C.1.

Lemma 2. *Consider* Romulus *which is illustrated and described in Fig. 3 and Fig. 4, respectively. There exists an adversary $\mathcal{A}_{\mathcal{C}}$, making q queries to $\widetilde{\mathsf{E}}$ such that for any $(K, N, S) \in \mathcal{K} \times \mathcal{N} \times \{0,1\}^n$, it holds that*

$$\Pr[\textsc{Enc}_{\mathcal{C}}(K, N, A) = S \mid A \leftarrow \mathcal{A}_{\mathcal{C}}(K, N, S)] = 1\,.$$

The number of ideal tweakable cipher queries by $\mathcal{A}_{\mathcal{C}}$ is $q = \left\lfloor \frac{\alpha}{2} \right\rfloor + 1$ for α being the number of associated data blocks that $\mathcal{A}_{\mathcal{C}}$ outputs.

We now give an adversary that inverts $\textsc{Enc}_{\mathcal{M}}$, i.e., for a given ciphertext (C, T) and a partial input (K, N), it finds a matching pair of state S and message M. The attack relies heavily on the invertibility of ξ as shown in Lemma 1 and is proven in Appendix C.2.

Lemma 3. *Consider* Romulus *which is illustrated and described in Fig. 3 and Fig. 4, respectively. There exists an adversary $\mathcal{A}_{\mathcal{M}}$, making q queries to $\widetilde{\mathsf{E}}$ such that for any $(K, N, (C, T)) \in \mathcal{K} \times \mathcal{N} \times \mathcal{C}$, it holds that*

$$\Pr[\textsc{Enc}_{\mathcal{M}}(K, N, S, M) = (C, T) \mid (S, M) \leftarrow \mathcal{A}_{\mathcal{M}}(K, N, (C, T))] = 1\,.$$

Romulus adversary $\mathcal{A}()$

$(K, N, A, M) \leftarrow\!\!\$\ \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{K}, \overline{N}) \leftarrow\!\!\$\ \mathcal{K} \times \mathcal{N}$
$(\overline{S}, \overline{M}) \leftarrow \mathcal{A}_{\mathcal{M}}(\overline{K}, \overline{N}, (C, T))$
$\overline{A} \leftarrow \mathcal{A}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{S})$
return $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$

Enc$_{\mathcal{M}}$ adversary $\mathcal{A}_{\mathcal{M}}(K, N, (C, T))$

$C_1, \ldots, C_\gamma \xleftarrow{n} C$
$S \leftarrow G^{-1}(T)$
for $i = \mu, \ldots, 1$ **do**
  $Y \leftarrow \widetilde{\mathsf{E}}^{-1}(K, N, S)$
  $(S, M_i) \leftarrow \xi^{-1}(Y, C_i)$
$M \leftarrow M_1 \parallel \ldots \parallel M_\mu$
return (S, M)

Enc$_{\mathcal{C}}$ adversary $\mathcal{A}_{\mathcal{C}}(K, N, S)$

$\alpha \leftarrow\!\!\$\ 2\mathbb{N}$
$A_2, \ldots, A_\alpha \leftarrow\!\!\$\ \{0,1\}^n$
$A_{\alpha+1} \leftarrow 0^n$
$Y \leftarrow \widetilde{\mathsf{E}}^{-1}(K, N, S)$
for $i = \frac{\alpha}{2}, \ldots, 1$ **do**
  $S \leftarrow Y \oplus A_{2i+1}$
  $Y \leftarrow \widetilde{\mathsf{E}}^{-1}(K, A_{2i}, S)$
$A_1 \leftarrow Y$
$A \leftarrow A_1 \parallel \ldots \parallel A_\alpha$
return A

Figure 5: Romulus adversary $\mathcal{A}$ from Theorem 1 which uses the inverse state-update-function ξ^{-1} from Lemma 1.

The number of ideal tweakable cipher queries by $\mathcal{A}_{\mathcal{M}}$ is $q = \mu$ for μ being the number of ciphertext blocks that $\mathcal{A}_{\mathcal{M}}$ receives as input.

Proof (of Theorem 1). We construct the following adversary $\mathcal{A}$ against Romulus as shown in Fig. 5. It samples a context (K, N, A) together with a message M at random and computes the ciphertext $(C, T) \leftarrow$ Romulus.Enc(K, N, A, M). It then samples $(\overline{K}, \overline{N})$ at random, computes $(\overline{S}, \overline{M}) \leftarrow \mathcal{A}_{\mathcal{M}}(\overline{K}, \overline{N}, (C, T))$, and $\overline{A} \leftarrow \mathcal{A}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{S})$. Finally, $\mathcal{A}$ outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. By using Lemma 2 and Lemma 3, we obtain

$$\textsc{Romulus.Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) = \textsc{Romulus.Enc}(K, N, A, M)\,.$$

As for the number of queries to the ideal tweakable cipher $\widetilde{\mathsf{E}}$, $\mathcal{A}$ makes $\mu + \lfloor \frac{\alpha}{2} \rfloor + 1$ while computing the ciphertext (C, T) for the first tuple and additionally μ and $\lfloor \frac{\alpha}{2} \rfloor + 1$ queries while running $\mathcal{A}_{\mathcal{M}}$ and $\mathcal{A}_{\mathcal{C}}$, respectively. This accumulates to $q = 2\mu + \lfloor \frac{\alpha}{2} \rfloor + \lfloor \frac{\alpha}{2} \rfloor + 2$ queries in total and concludes the proof. □

The gist of the attack is finding a different $\overline{A}$ which yields the target ciphertext. The attack easily extends to a context discovery attack (CDY$^\star_A$) [MLGR23]. Hence, we can conclude that Romulus is also vulnerable with respect to the weaker security notions CMT$_\mathsf{K}$ and CMT$_\mathsf{N}$ by using [MLGR23, Corollary 3]. Furthermore, the attack can be translated to one against CMT$_\mathsf{A}$ by observing that the adversary can choose $\overline{A}$ to differ from A at some point (note that $\mathcal{A}$ can freely choose all but one block). Finally, the attack is also extendable to the more restricted notion CMT$^\star_\mathsf{A}$ by choosing the second key-nonce pair $(\overline{K}, \overline{N})$ not at random but equal to the first pair (K, N).

Similar Results. Just as Romulus, we can attack Elephant (cf. Appendix B.1), Gift-Cofb (cf. Appendix B.2), and Photon-Beetle (cf. Appendix B.3) with a minimum number of queries. The attack against Gift-Cofb is very similar to the one given here, the core difference is the state-update-function. The attack against Elephant is even simpler as the scheme does not use a state-update-function. For Photon-Beetle, our attack exploits the choice of its initial state.

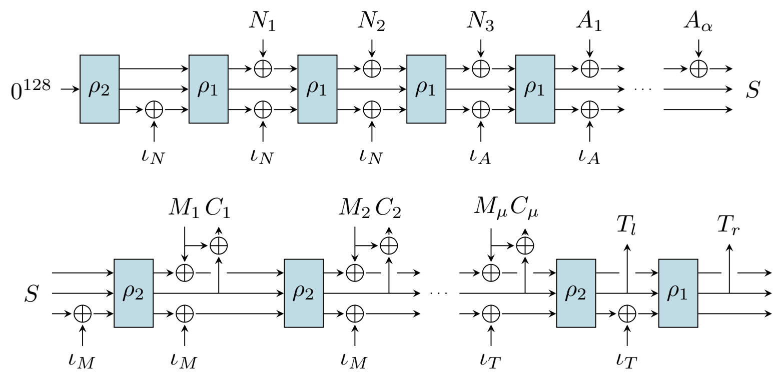


Figure 6: Illustration of TinyJambu in terms of $\text{Enc}_{\mathcal{C}}$ (top) and $\text{Enc}_{\mathcal{M}}$ (bottom), where $\rho_1 = \text{BC}_1(K, \cdot)$ and $\rho_2 = \text{BC}_2(K, \cdot)$.

3.2 TinyJambu

TinyJambu [WH21] is a block-cipher-based authenticated encryption scheme. The specification introduces the TinyJambu mode, which is a lightweight variant of the Jambu mode [WH16]. The latter was part of the CAESAR competition [Ber14]. For the permutation underlying TinyJambu, a keyed permutation based on non-linear feedback shift registers is defined.

3.2.1 Description of TinyJambu

The pseudocode of TinyJambu is given in Fig. 7 and an illustration of the scheme can be found in Fig. 6. TinyJambu follows the CpP-approach, i.e., it first processes the context (K, N, A) via the function $\text{Enc}_{\mathcal{C}}$ and then passes the output on to $\text{Enc}_{\mathcal{M}}$, where it is processed together with the message. TinyJambu uses two keyed permutations BC_1 and BC_2, both based on the same keyed permutation that is applied 640 and 1024 times for BC_1 and BC_2, respectively.

3.2.2 Committing Attack Against TinyJambu

In this section, we show that TinyJambu does not achieve CMT security.[10] The attack exploits the short tag length of 64 bits in TinyJambu, which enables an efficient deployment of the birthday bound. In the security proof of TinyJambu (see [WH21, Section 6]), this setting is modeled with only one permutation ρ. We adopt the same for the TinyJambu attack given in the following.

Theorem 2. *Consider* TinyJambu *which is illustrated and described in Fig. 6 and Fig. 7, respectively. Let* BC_1 *and* BC_2 *be modeled as an ideal cipher* E*. Then there exists an adversary* $\mathcal{A}$ *that makes* q *queries to* E *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\text{TinyJambu}}(\mathcal{A}) \geq \frac{3}{8}\,.$$

Here, $q = 2(2^{32}+1)(6+\alpha+\mu)$ *for* α *and* μ *the number of associated data and message blocks, respectively, that* $\mathcal{A}$ *outputs.*

[10] Dunkelmann et al. [DGL23] show that TinyJambu is not CMT_{K}-secure. Their committing attack is based on a related-key forgery attack on the underlying block-cipher and only applies to the larger parameter sets but not the main one. In contrast, our attack is based on the mode and works for all parameter sets.

$\textsc{TinyJambu}.\textsc{Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, S, M)$
return (C, T)

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$N_1, N_2, N_3 \xleftarrow{32} N$
$A_1, \dots, A_\alpha \xleftarrow{32} \mathtt{pad}_{0^*}(A, 32)$
$S \leftarrow \mathrm{BC}_2(K, 0^{128})$
for $i = 1, \dots, 3$
    $S \leftarrow S \oplus (0^{64} \parallel \iota_N)$
    $S \leftarrow \mathrm{BC}_1(K, S)$
    $S \leftarrow (\lceil S \rceil_{32} \oplus N_i) \parallel \lfloor S \rfloor_{96}$
for $i = 1, \dots, \alpha$
    $S \leftarrow S \oplus (0^{64} \parallel \iota_A)$
    $S \leftarrow \mathrm{BC}_1(K, S)$
    $S \leftarrow (\lceil S \rceil_{32} \oplus A_i) \parallel \lfloor S \rfloor_{96}$
return S

$\textsc{Enc}_{\mathcal{M}}(K, S, M)$

$M_1, \dots, M_\mu \xleftarrow{32} \mathtt{pad}_{0^*}(M, 32)$
for $i = 1, \dots, \mu$
    $S \leftarrow S \oplus (0^{64} \parallel \iota_M)$
    $S \leftarrow \mathrm{BC}_2(K, S)$
    $S \leftarrow (\lceil S \rceil_{32} \oplus M_i) \parallel \lfloor S \rfloor_{96}$
    $C_i \leftarrow [S]_{33}^{64}$
$S \leftarrow S \oplus (0^{64} \parallel \iota_T)$
$S \leftarrow \mathrm{BC}_2(K, S)$
$T_l \leftarrow [S]_{33}^{64}$
$S \leftarrow S \oplus (0^{64} \parallel \iota_T)$
$S \leftarrow \mathrm{BC}_1(K, S)$
$T_r \leftarrow [S]_{33}^{64}$
$C \leftarrow \lceil C_1 \parallel \dots \parallel C_\mu \rceil_{|M|}$
$T \leftarrow T_l \parallel T_r$
return (C, T)

Figure 7: Pseudocode of TinyJambu [WH21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$. If the last block of associated data or message is not of full length, TinyJambu XORs the respective lengths into the last bits (as part of ι_A and ι_M).

Proof. We construct a CMT adversary $\mathcal{A}$ against TinyJambu as follows: First, we randomly choose two different keys $K \neq \overline{K}$ and a target ciphertext C. Note that, due to the structure of TinyJambu, the context produces a key stream which is XORed with the message to obtain the ciphertext. Hence, for a random context it is always possible to find a M such that the TinyJambu encryption results in the initially chosen target ciphertext C. In the following, we will implicitly consider this "matching" message for each context that occurs. Hence it suffices to find two different contexts that—together with their matching message—yield colliding tags.

We start by building two lists of tags, where for one we use K as key and in the other $\overline{K}$. For this, sample distinct (N_i, A_i) for $i \in \{1, \dots, 2^{32}+1\}$ and compute the corresponding tags T_i using the key K. We then set $L = (T_i)_{i \in [2^{32}+1]}$. Analogously, we sample distinct $(\overline{N}_i, \overline{A}_i)$[11] for $i \in [2^{32}+1]$ and write the corresponding tags $\overline{T}_i$ (computed using $\overline{K}$) into the list $\overline{L} = (\overline{T}_i)_{i \in [2^{32}+1]}$. Building the two lists, takes a total of $q = 2(2^{32}+1)(6+\alpha+\mu)$ queries to ρ.

For the context (K, N_i, A_i), denote the states before the second to last permutation application (see Fig. 6) by S_i, and analogously for $(\overline{K}, \overline{N}_i, \overline{A}_i)$ by $\overline{S}_i$. Note that for a fixed key, TinyJambu can be considered a sponge-based function with rate $r = 32$ and capacity $c = 96$. Therefore, the event $S_i = S_j$ for $i \neq j$ (and analogously $\overline{S}_i = \overline{S}_j$ for $i \neq j$), constitutes an inner collision, which is—for a sponge with capacity 96—highly unlikely[12]. As we model both BC_1 and BC_2 by an ideal cipher E and the states S_i (and respectively $\overline{S}_i$) collide with negligible probability, we can assume the list elements to be distributed uniformly and independently.

This puts us in the situation of Lemma 5 (for $l_1 = l_2 = 2^{32}+1$ and $\tau = 64$), hence we

[11] For sake of simplicity, we assume that $\mathcal{A}$ chooses the $\overline{A}_i$ with same block length.
[12] More precisely, the probability is $\frac{q(q+1)}{2^{97}} - \frac{q(q-1)}{2^{129}}$ [BDPVA07].

obtain the following lower bound for finding a collision $T_i = \overline{T}_j$:

$$\left(1 - \exp\left(\frac{-(2^{33}+2)(2^{33}+1)}{2^{65}}\right)\right) \cdot \frac{2 \cdot (2^{32}+1)^2}{(2^{33}+2)^2 - (2^{33}+2)} . \tag{1}$$

Since $\frac{-(2^{33}+2)(2^{33}+1)}{2^{65}} \leq \frac{-2^{33} \cdot 2^{33}}{2^{65}} = -2$ the first factor in Eq. (1) can be bounded below by $1 - \exp\left(\frac{-(2^{33}+2)(2^{33}+1)}{2^{65}}\right) \geq 1 - e^{-2} \geq \frac{3}{4}$. The second factor in Eq. (1) simplifies to $\frac{2^{32}+1}{2^{33}+1}$ which is lower bound by $\frac{1}{2}$. In total, the probability for finding a tag collision (and hence winning the game CMT) is at least $\frac{3}{8}$. □

The attack exploits the fact that TinyJambu uses a very short tag (64 bits) compared to the other schemes—the only other scheme considered in this work with a 64-bit tag is Elephant, though they also provide a parameter set with a larger tag. Increasing the tag length of TinyJambu would render our attack impractical. Note, however, that increasing the tag length to 128 does not make TinyJambu committing secure. For such a variant of TinyJambu, we can similarly apply a birthday attack to find a collision on the capacity part, while the associated data is processed. Such a 96-bit collision can be found with about 2^{48} queries and, by properly choosing the associated data, results in a full collision. One can modify the parameters such that a 127-bit collision has to be found—though this variant is impractical as the inputs would be absorbed bit-wise.

By construction, the above attack is a CMT_{K} attack, as K and $\overline{K}$ were chosen to be different. Moreover, by requiring not only the tuples (N_i, A_i) to differ for all i, but the individual nonces and associated data, we also obtain a CMT_{N} and a CMT_{A} attack.

Similar Results. The sponge-based AE scheme Xoodyak (cf. Appendix B.4) can also be efficiently attacked using a birthday attack. We target the full-state sponge part of Xoodyak when the associated data is processed. Due to the padding of Xoodyak, the adversary cannot control the entire state—32 bits are reserved for the padding. Hence, a birthday attack on these 32 bits is required which then extends to a CMT attack.

3.3 Ascon

Ascon [DEMS21] is a sponge-based AE scheme. The scheme was chosen as the primary candidate for lightweight applications in the CAESAR competition. Furthermore, Ascon was selected to be standardized as part of the NIST LWC standardization process. As part of the CAESAR competition and the NIST LWC standardization process, Ascon enjoys a long line of research, in particular, with respect to the underlying permutation Ascon-P. For the authenticated encryption mode, no formal security analysis existed until recently, when Lefevre and Mennink [LM24] gave the first security proof for Ascon.[13]

3.3.1 Description of Ascon

The pseudocode of Ascon is given in Fig. 9 and further illustration is provided in Fig. 8. Similar to the other schemes, Ascon can be viewed as a CpP scheme which first processes the context using $\text{Enc}_{\mathcal{C}}$ before the message is processed using $\text{Enc}_{\mathcal{M}}$. A core feature of Ascon is that at the very start (first permutation of $\text{Enc}_{\mathcal{C}}$) and the very end (last permutation of $\text{Enc}_{\mathcal{M}}$), it uses more rounds of the underlying permutation for security (ρ^a and ρ^b for $a = 12$ and $b = 6$). Note that Ascon XORs the key three additional times: After the first permutation as well as before and after the last permutation. We call the former two instances state-blinding and the latter output-blinding.

[13] An earlier work [JLM14] showed security for a simplified version of Ascon.

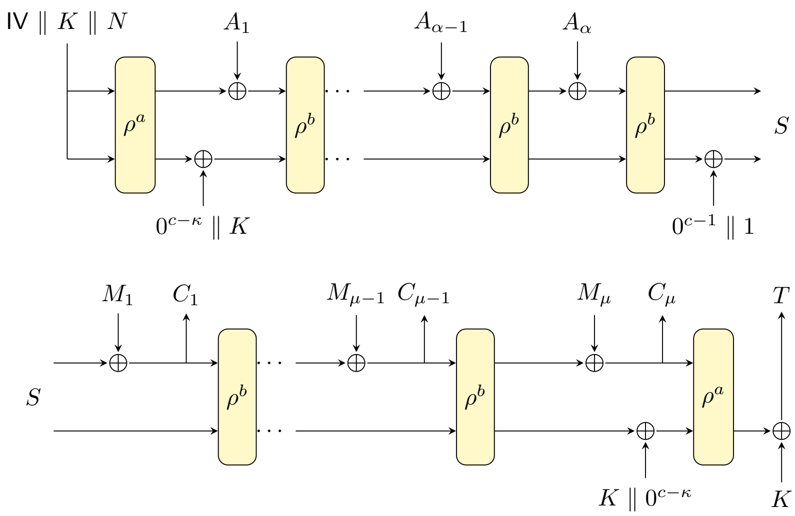


Figure 8: Illustration of Ascon in terms of $\textsc{Enc}_{\mathcal{C}}$ (top) and $\textsc{Enc}_{\mathcal{M}}$ (bottom).

3.3.2 Committing Security of Ascon

We show that Ascon achieves CMT security. We model the two permutations ρ^a and ρ^b by one ideal permutation ρ, essentialy adopting the approach of Isap [DEM+20], where the different permutations for the re-keying function are also modeled by one random permutation. Further, we consider a slightly different order of inputs at two points in the Ascon encryption. Firstly, the initial state is changed by moving the initialization vector from the beginning of the state to the end. Secondly, the state-blinding is changed so that it affects the first bits of the inner state rather than the last bits.[14] Note that these changes are purely cosmetic. They do not influence the overall security of Ascon but simplify our proof, as we can capture the state-blinding by considering a larger rate. Lastly, for sake of simplicity, we drop the domain separation of Ascon in the proof. It can, however, be easily incorporated by XORing $1 \parallel 0^{c-1}$ instead of $0^{c-1} \parallel 1$, i.e., moving the domain separation to the first bit of the capacity as opposed to the last bit. This neither influences the security of Ascon nor interferes with the purpose of the domain separation but it allows us to incorporate the domain separation into the rate.[15]

We show committing security of Ascon by arguing about the collision resistance of plain sponges—an alternative approach is to make use of the indifferentiability, as was done in concurrent and independent work [NSS23]. Theorem 9 provides a bound for the collision resistance of plain sponges, however, applying it to (a plain sponge version of) Ascon does not yield a good bound. The reason for this is that the capacity of Ascon's initial state (64 bits) is small compared to the one of the remaining states (≥ 128 bits). As part of our argument, we prove that Theorem 9 also holds if the initial state has a smaller capacity than the rest of the sponge construction.[16]

This generalized version of [BS23, Theorem 8.6] uses the same graph modelling as the original proof. We consider a sponge-based hash function $\mathcal{H}$, obtained from a permutation $\rho\colon \{0,1\}^n \to \{0,1\}^n$, with capacity c, rate r, initial capacity $c^\star$, and output length $w \leq r$. We model a CR adversary against $\mathcal{H}$ as follows: We build a directed graph G from the ideal permutation queries the adversary makes. The nodes in G are the 2^n bit strings of

[14] This only affects the first state-blinding; the second one is already of that form.

[15] The same argument appears in [DJS19] (AE scheme Slae) and [DHP+21] (Xoodyak).

[16] Naito and Ohta [NO14] showed that a smaller capacity in the initial state can also be tolerated when considering indifferentiability, which the initial result [BDPV08] did not.

$\textsc{Ascon}.\textsc{Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, S, M)$
return (C, T)

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$A_1, \ldots, A_\alpha \xleftarrow{r} \texttt{pad}_{10^*}(A, r)$
$S \leftarrow \rho^a(\mathsf{IV} \parallel K \parallel N)$
$S \leftarrow S \oplus (0^{n-\kappa} \parallel K)$
for $i = 1, \ldots, \alpha$
  $Y \leftarrow S \oplus (A_i \parallel 0^c)$
  $S \leftarrow \rho^b(Y)$
$S \leftarrow S \oplus (0^{n-1} \parallel 1)$
return S

$\textsc{Enc}_{\mathcal{M}}(K, S, M)$

$M_1, \ldots, M_\mu \xleftarrow{r} \texttt{pad}_{10^*}(M, r)$
$Y \leftarrow S \oplus (M_1 \parallel 0^c)$
$C_1 \leftarrow \lceil Y \rceil_r$
for $i = 2, \ldots, \mu$
  $S \leftarrow \rho^b(Y)$
  $Y \leftarrow S \oplus (M_i \parallel 0^c)$
  $C_i \leftarrow \lceil Y \rceil_r$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
$Y \leftarrow Y \oplus (0^r \parallel K \parallel 0^{c-\kappa})$
$S \leftarrow \rho^a(Y)$
$T \leftarrow \lfloor S \rfloor_\tau \oplus K$
return (C, T)

Figure 9: Pseudocode of Ascon [DEMS21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$.

length n and an edge from Y to S is added if $\mathcal{A}$ makes a query of the form $\rho(Y) = S$ or $\rho^{-1}(S) = Y$ (the graph starts with no edges). The edges resulting from ρ queries are called *forward edges* and the ones resulting from ρ^{-1} queries are referred to as *backward edges*. For $s \geq 1$, a *PS-path* of length s is a sequence of $2s$ nodes

$$Y_0, S_1, Y_1, S_2, \ldots, S_{s-1}, Y_{s-1}, S_s$$

with

1. $\lfloor Y_0 \rfloor_{c^\star} = \mathsf{IV}$,
2. $\lfloor Y_i \rfloor_c = \lfloor S_i \rfloor_c$ for all $i \in \{1, \ldots, s-1\}$, and
3. G contains edges from Y_{i-1} to S_i for all $i \in \{1, \ldots, s\}$.

We define the *input* of a PS-path as $I :- (Z_0, \ldots, Z_{s-1}) \in \{0,1\}^{n-c^\star} \times \{0,1\}^r \times \cdots \times \{0,1\}^r$ for $Z_0 = \lceil Y_0 \rceil_{n-c^\star}$ and $Z_i = \lceil S_i \rceil_r \oplus \lceil Y_i \rceil_r$ for all $i \in \{1, \ldots, s-1\}$. Further, we define the *result* of a PS-path as $\lceil S_s \rceil_w$ which corresponds to the output of the hash function. As a notation for PS-paths that incorporates the input, we write

$$Z_0 | Y_0 \to \cdots \to S_{s-2} | Z_{s-2} | Y_{s-2} \to S_{s-1} | Z_{s-1} | Y_{s-1} \to S_s\,.$$

Next, we define two properties a pair $(P, \overline{P})$ of PS-paths can have. For this, denote the nodes in $\overline{P}$ by $\overline{Y}_0, \overline{S}_1, \overline{Y}_1, \ldots, \overline{Y}_{\bar{l}-1}, \overline{S}_{\bar{l}}$. Firstly, the paths P and $\overline{P}$ are *colliding* if their inputs differ but their results agree. Secondly, the paths P and $\overline{P}$ are *problematic* if their inputs differ and

1. $Y_{l-1} = \overline{Y}_{\bar{l}-1}$ or
2. at least one of the edges in P or $\overline{P}$ is a backward edge.

We can show the following lemma regarding the probability of finding a pair of problematic paths. The proof can be found in Appendix C.3.

Lemma 4. *Consider the graph modelling as described above. Let* PP *be the event that* $\mathcal{A}$ *finds a pair of problematic paths and* q *the number of queries that* $\mathcal{A}$ *makes to* ρ*, then*

$$\Pr[\mathsf{PP}] \leq \frac{q}{2^{c^\star - 1}} + \frac{q(q-1)}{2^c}\,.$$

Using this result yields a generalization of [BS23, Theorem 8.6] to the case that the initial state has a smaller capacity. This is proven in Appendix C.4.

Theorem 3 (Generalized version from [BS23, Theorem 8.6])**.** *For $\mathcal{H}$ a hash function obtained from a permutation $\rho\colon \{0,1\}^n \to \{0,1\}^n$, with capacity c, rate r (so $n = r + c$), initial capacity $c^\star$, and output length $w \leq r$, it holds that*

$$\mathbf{Adv}^{\mathsf{CR}}_{H}(\mathcal{A}) \leq \frac{q(q-1)}{2^w} + \frac{q}{2^{c^\star-1}} + \frac{q(q-1)}{2^c}\,.$$

The theorem below establishes the committing security of Ascon.

Theorem 4. *Consider* Ascon *which is illustrated and described in Fig. 8 and Fig. 9, respectively. Let ρ^a and ρ^b be modeled as a random permutation ρ. Then for any adversary $\mathcal{A}$ making $q \leq 2^{127}$ queries to ρ, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Ascon}}(\mathcal{A}) \leq 1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right) + \frac{q}{2^{63}} + \frac{q(q-1)}{2^{128}}\,.$$

Proof. Let $\mathcal{A}$ be a CMT adversary against Ascon with output denoted by (K, N, A, M), $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$. Further note that IV denotes the initialization vector used in Ascon. As a first step, we observe that finding different inputs to the Ascon encryption that give the same ciphertext is easy due to the duplex construction used in the sponge. The difficulty in breaking CMT security for Ascon lies in finding a tag collision, which is why we focus our attention on this task. An adversary that wins the game CMT against Ascon, in particular finds a tag collision, i.e., it wins the game $\mathsf{TagColl}$ (see Fig. 15). Hence we obtain

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Ascon}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{TagColl}}_{\textsc{Ascon}}(\mathcal{A})\,.$$

We consider the directed graph described above for $n = 320$. We assume $\mathcal{A}$ to make queries to ρ that correspond to its output, i.e., querying all states that occur during the evaluation of Ascon for the output tuples of $\mathcal{A}$. This assumption is without loss of generality, as we can easily transform any adversary $\mathcal{A}$ into one that runs $\mathcal{A}$ to obtain $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ and—before outputting the same—makes all queries to ρ corresponding to (K, N, A, M) and $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$. Additionally, we assume $\mathcal{A}$ to make no redundant queries, i.e., once two values (Y, S) are known to be connected via an edge, no further ρ queries are made on Y and no further ρ^{-1} queries are made on S.

We define a special kind of path, the *A-path*, which models Ascon's tag generation—recall that we passed over to the game $\mathsf{TagColl}$ at the beginning of the proof. An A-path P_{A} of length l[17] is a sequence of $2l$ nodes

$$Y_0, S_1, Y_1, S_2, \ldots, S_{l-1}, Y_{l-1}, S_l$$

with the following four properties:

1. $Y_0 = K \parallel N \parallel \mathsf{IV}$,
2. $\lfloor Y_1 \rfloor_{256} = \lfloor S_1 \rfloor_{256} \oplus (K \parallel 0^{128})$ and $\lfloor Y_{l-1} \rfloor_{256} = \lfloor S_{l-1} \rfloor_{256} \oplus (K \parallel 0^{128})$,
3. $\lfloor Y_i \rfloor_{256} = \lfloor S_i \rfloor_{256}$ for all $i \in \{2, \ldots, l-2\}$, and
4. G contains edges from Y_{i-1} to S_i for all $i \in \{1, \ldots, l\}$,

for some $K, N \in \{0,1\}^{128}$. We define the *input* of an A-path P_{A} as

$$I :- (K, N, X_1, \ldots, X_{l-1}) \in \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{64} \cdots \times \{0,1\}^{64}$$

[17] Note that $l \geq 3$, as Ascon involves at least three applications of ρ (for $\mu = \alpha = 1$).

for $K \parallel N := \lceil Y_0 \rceil_{256}$ and $X_i = \lceil S_i \rceil_{64} \oplus \lceil Y_i \rceil_{64}$ for all $i \in \{1, \dots l-1\}$. The *result* of P_{A} is defined as $R = \lfloor S_l \rfloor_{128} \oplus K$. By construction, this models the tag generation of Ascon: The sponge processes the input consisting of key K, nonce N, and the tuple of associated data and message $(A, M) = (X_1, \dots, X_{l-1})$ and the result is the tag. As a notation for A-paths that incorporates the input, we write

$$(K \parallel N)|Y_0 \to \cdots \to S_{l-2}|X_{l-2}|Y_{l-2} \to S_{l-1}|X_{l-1}|Y_{l-1} \to S_l\,.^{18}$$

Next, we define two properties a pair $(P_{\mathrm{A}}, \overline{P}_{\mathrm{A}})$ of A-paths can have. For this, denote the nodes in $\overline{P}_{\mathrm{A}}$ by $\overline{Y}_0, \overline{S}_1, \overline{Y}_1, \dots, \overline{Y}_{\bar{l}-1}, \overline{S}_{\bar{l}}$. Firstly, the paths P_{A} and $\overline{P}_{\mathrm{A}}$ are *colliding* if their inputs differ but their results agree. Secondly, the paths P_{A} and $\overline{P}_{\mathrm{A}}$ are *problematic* if their inputs differ and (1) $Y_{l-1} = \overline{Y}_{\bar{l}-1}$ or (2) at least one of the edges in P or $\overline{P}$ is a backward edge. We are interested in the event CP_{A} that $\mathcal{A}$ finds a pair of colliding A-paths. Note that finding such paths means that $\mathcal{A}$ wins the game $\mathsf{TagColl}$. In order to compute the probability of CP_{A}, we define the auxiliary event PP_{A} that $\mathcal{A}$ finds a pair of problematic A-paths. Using this, we obtain

$$\mathbf{Adv}^{\mathsf{TagColl}}_{\textsc{Ascon}}(\mathcal{A}) = \Pr[\mathsf{CP}_{\mathrm{A}}] \leq \Pr[\mathsf{CP}_{\mathrm{A}} \wedge \neg\mathsf{PP}_{\mathrm{A}}] + \Pr[\mathsf{PP}_{\mathrm{A}}]\,,$$

and proceed by deriving upper bounds for both of the above summands.

We start with the easier case, which is giving an upper bound for the probability that $\mathsf{CP}_{\mathrm{A}} \wedge \neg\mathsf{PP}_{\mathrm{A}}$ holds, i.e., that $\mathcal{A}$ finds a pair of colliding A-paths that is not problematic. Hence, $\mathcal{A}$ finds two different inputs $I = (K, N, X_1, \dots, X_{l-1})$ and $\overline{I} = (\overline{K}, \overline{N}, \overline{X}_1, \dots, \overline{X}_{l-1})$ such that the corresponding A-paths

$$\begin{gathered} Y_0, S_1, Y_1, S_2, \dots, S_{l-1}, Y_{l-1}, S_l \\ \overline{Y}_0, \overline{S}_1, \overline{Y}_1, \overline{S}_2, \dots, \overline{S}_{\bar{l}-1}, \overline{Y}_{\bar{l}-1}, \overline{S}_{\bar{l}} \end{gathered}$$

fulfill $Y_{l-1} \neq \overline{Y}_{\bar{l}-1}$ and have equal results, i.e., $\lfloor \rho(Y_{l-1}) \rfloor_{128} \oplus K = R = \overline{R} = \left\lfloor \rho(\overline{Y}_{\bar{l}-1}) \right\rfloor_{128} \oplus \overline{K}$. By definition of an A-path, this implies

$$\left\lfloor \rho(S_{l-1} \oplus (X_{l-1} \| K \| 0^{128})) \right\rfloor_{128} \oplus K = \left\lfloor \rho(\overline{S}_{\bar{l}-1} \oplus (\overline{X}_{\bar{l}-1} \| \overline{K} \| 0^{128})) \right\rfloor_{128} \oplus \overline{K}\,. \tag{2}$$

Since ρ is a random permutation and $\mathcal{A}$ only used forward queries (as $\neg\mathsf{PP}_{\mathrm{A}}$ holds), finding such a collision is unlikely. We assume—to the benefit of the adversary $\mathcal{A}$—that it can choose $S_{l-1}, \overline{S}_{\bar{l}-1} \in \{0,1\}^{128}$ freely, i.e., it must not be part of an A-path for some input. The probability of $\mathcal{A}$ finding (S_{l-1}, X_{l-1}, K), $(\overline{S}_{\bar{l}-1}, \overline{X}_{\bar{l}-1}, \overline{K})$[19] such that $Y_{l-1} \neq \overline{Y}_{\bar{l}-1}$ and Eq. (2) holds with q queries, equals the probability of finding a collision in a list of q uniformly distributed elements. Using Theorem 11 for $q \leq 2^{127}$, the latter can be bounded from above by

$$1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right)\,.$$

Next, we turn our attention to deriving an upper bound for $\Pr[\mathsf{PP}_{\mathrm{A}}]$. Lemma 4 shows that finding problematic paths is hard, even for a plain sponge without Ascon's blinding mechanisms, which is why we reduce to this setting. We consider a sponge-based hash function $\mathcal{H}$ obtained from the permutation ρ with rate 256 for the first round of absorption and rate 196 for all remaining ones. Further, its initial state is given by $0^{256} \parallel \mathsf{IV}$ and the output produced by $\mathcal{H}$ has length 128. We next observe that a pair of problematic

[18] While not visible in this representation, by definition of A-paths, Y_1 and S_1 (respectively Y_{l-1} and S_{l-1}) differ not only in their first 64 bits but also from bit 65 to 192, where the key is XORed.

[19] Note that $\mathcal{A}$ must choose $(S_{l-1}, X_{l-1}, K) \neq (\overline{S}_{\bar{l}-1}, \overline{X}_{\bar{l}-1}, \overline{K})$ to ensure $Y_{l-1} \neq \overline{Y}_{\bar{l}-1}$.

A-paths, can also be considered as a pair of problematic PS-paths, i.e., in particular the event PP_{A} implies the event PP. Let $(P_{\text{A}}, \overline{P}_{\text{A}})$ be a pair of problematic A-paths, i.e.,

$$\begin{aligned} P_{\text{A}} =&(K \parallel N)|Y_0 \to \cdots \to S_{l-2}|X_{l-2}|Y_{l-2} \to S_{l-1}|X_{l-1}|Y_{l-1} \to S_l \\ \overline{P}_{\text{A}} =&(\overline{K} \parallel \overline{N})|\overline{Y}_0 \to \cdots \to \overline{S}_{\bar{l}-2}|\overline{X}_{\bar{l}-2}|\overline{Y}_{\bar{l}-2} \to \overline{S}_{\bar{l}-1}|\overline{X}_{\bar{l}-1}|\overline{Y}_{\bar{l}-1} \to \overline{S}_{\bar{l}}\,. \end{aligned}$$

By defining

$$\begin{aligned} Z_0 &= K \parallel N \in \{0,1\}^{256} & \overline{Z}_0 &= \overline{K} \parallel \overline{N} \in \{0,1\}^{256} \\ Z_1 &= X_1 \parallel K \in \{0,1\}^{192} & \overline{Z}_1 &= \overline{X}_1 \parallel \overline{K} \in \{0,1\}^{192} \\ Z_i &= X_i \parallel 0^{128} \in \{0,1\}^{192} & \overline{Z}_i &= \overline{X}_i \parallel 0^{128} \in \{0,1\}^{192} \\ Z_{l-1} &= X_{l-1} \parallel K \in \{0,1\}^{192} & \overline{Z}_{\bar{l}-1} &= \overline{X}_{\bar{l}-1} \parallel \overline{K} \in \{0,1\}^{192} \end{aligned}$$

we obtain the following presentation of $(P_{\text{A}}, \overline{P}_{\text{A}})$ as PS-paths:

$$\begin{aligned} P =&Z_0|Y_0 \to \cdots \to S_{l-2}|Z_{l-2}|Y_{l-2} \to S_{l-1}|Z_{l-1}|Y_{l-1} \to S_l \\ \overline{P} =&\overline{Z}_0|\overline{Y}_0 \to \cdots \to \overline{S}_{\bar{l}-2}|\overline{Z}_{\bar{l}-2}|\overline{Y}_{\bar{l}-2} \to \overline{S}_{\bar{l}-1}|\overline{Z}_{\bar{l}-1}|\overline{Y}_{\bar{l}-1} \to \overline{S}_{\bar{l}}\,. \end{aligned}$$

Visualization for this is provided in Fig. 10. As we neither change $(Y_{l-1}, \overline{Y}_{\bar{l}-1})$ nor any of the edges, the paths $(P, \overline{P})$ form a pair of problematic PS-paths. Thus, we have shown that PP_{A} implies PP, hence $\Pr[\mathsf{PP}_{\text{A}}] \leq \Pr[\mathsf{PP}]$. This allows us to focus on the plain sponge setting for the rest of the proof. Applying Lemma 4 for $n = 320$, $c^\star = 64$, and $c = 128$, then yields the following bound for finding a pair of problematic plain sponge paths

$$\Pr[\mathsf{PP}] \leq \frac{q}{2^{63}} + \frac{q(q-1)}{2^{128}}\,.$$

In total, we have shown

$$\begin{aligned} \mathbf{Adv}^{\mathsf{CMT}}_{\text{Ascon}}(\mathcal{A}) &\leq \mathbf{Adv}^{\mathsf{TagColl}}_{\text{Ascon}}(\mathcal{A}) \\ &\leq \Pr[\mathsf{CP}_{\text{A}} \wedge \neg\mathsf{PP}_{\text{A}}] + \Pr[\mathsf{PP}_{\text{A}}] \\ &\leq 1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right) + \frac{q}{2^{63}} + \frac{q(q-1)}{2^{128}}\,, \end{aligned}$$

which finishes the proof. □

Similar Results. Similar to Ascon, we can show committing security for Isap (cf. Appendix B.5) and Schwaemm (cf. Appendix B.6). For Isap, the overall idea is similar to the one used in the Ascon proof. While Isap's large IV allows a direct application of Theorem 9, extra care is necessary to handle its re-keying mechanism. For Schwaemm, we consider a slightly modified version $\text{Schwaemm}_{\mathsf{IV}}$, where we introduce a fixed IV in the initial state as Ascon and Isap have. The proof then makes use of the indifferentiability of (the plain sponge version) of $\text{Schwaemm}_{\mathsf{IV}}$ from a random function.

4 Zero-Padding the NIST Finalists

In this section we consider the zero-padding approach for the NIST finalists, which is described in Section 4.1. In Section 4.2, we refine the class of CpP schemes and show for a subclass—containing Photon-Beetle and Xoodyak—that CMT_{K} attacks reduce to finding key collisions in the underlying function $\text{Enc}_{\mathcal{C}}$. In Section 4.3, we show that zp-Elephant does not achieve CMT_{K} if the number of padded zeros is less or equal to the block-size of Elephant. In Section 4.4, we show that zp-Isap can be attacked with about $2^{\frac{\tau}{2}}$ queries regardless of the number of padded zeros.

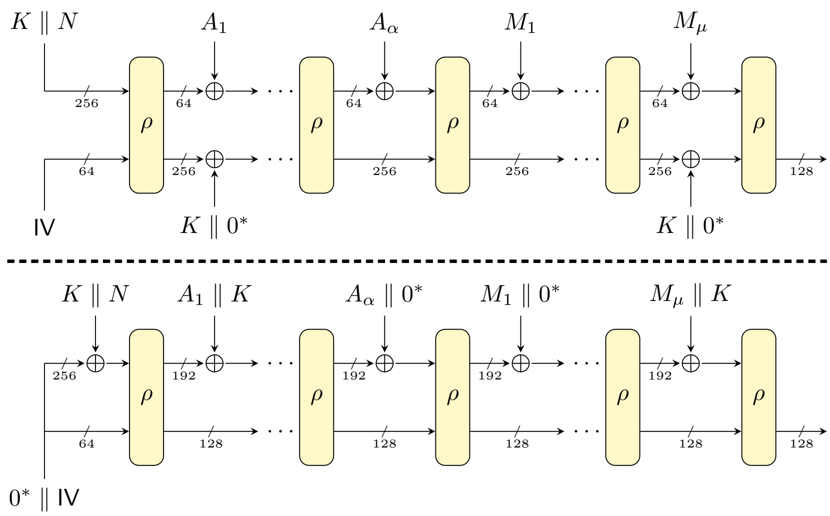


Figure 10: Illustration of a proof step for Ascon (Theorem 4). Ascon is represented as a plain sponge with a larger rate.

4.1 Motivation and Description of the Zero-Padding Approach

As discussed in Section 2, there are several transformations that turn arbitrary AE schemes into committing AE schemes but these might not work in the context of the NIST LWC finalists. A much simpler and less invasive method is the so-called zero-padding suggested by Albertini et al. [ADG+22]. Here, a number of zeros is prepended to the message before encrypting it, i.e.,

$$\textsc{zp-Ae.Enc}(K, N, A, M) \coloneqq \textsc{Ae.Enc}(K, N, A, 0^z \parallel M)\,.$$

Decryption uses the decryption algorithm of the AE scheme and additionally checks whether the resulting message has z leading zeros. If it does, the zeros are discarded and the message is returned, otherwise, the ciphertext is rejected and $\perp$ is outputted. For any scheme Ae, we write its corresponding zero-padding scheme with a prefix zp-, i.e., zp-Ae; the number of zeros that are padded is denoted by z. Zero-padding is particularly interesting as it does not require any changes to the scheme: prepending zeros to the message and verifying that a decrypted message has leading zeros can be done outside of the AE scheme.

Albertini et al. [ADG+22] suggested zero-padding as a method to obtain $\mathrm{CMT}_{\mathsf{K}}$ security and Menda et al. [MLGR23] showed that in general it does not achieve CMT security as $\mathrm{CMT}^{\star}_{\mathsf{A}}$ attacks are still possible. Since all our attacks are also $\mathrm{CMT}^{\star}_{\mathsf{A}}$ attacks—or can be easily modified to be—we cannot achieve CMT security via the zero-padding. Nonetheless, zero-padding as a method to obtain $\mathrm{CMT}_{\mathsf{K}}$ security is still interesting, as the latter seems to be the more relevant notion for practical attacks [ADG+22, DGRW18, LGR21].

This raises the question whether the broken NIST LWC finalists can be patched by zero-padding to achieve $\mathrm{CMT}_{\mathsf{K}}$ security. We provide several negative results for this: The schemes zp-Photon-Beetle and zp-Xoodyak remain insecure, irrespectively of the value of z, while zp-Elephant is vulnerable if $z \leq n$.

Besides achieving $\mathrm{CMT}_{\mathsf{K}}$ security, Naito et al. [NSS23] showed that zero-padding can increase CMT security (up to certain level that is) for Ascon. More precisely, they show that padding zeros can compensate for smaller tags. This raises the question whether it also improves the security of other schemes. We provide yet another negative result, showing that Isap does not benefit from zero-padding regarding its CMT security.

Game $\mathsf{KeyColl}$

$(K, N, A), (\overline{K}, \overline{N}, \overline{A}) \leftarrow \mathcal{A}()$
if $K = \overline{K}$
  return 0
$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$\overline{S} \leftarrow \textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A})$
return $(S = \overline{S})$

Figure 11: Security game KeyColl for CpP schemes.

4.2 Zero-Padding and Context-Pre-Processing Schemes

In Section 2, we introduced CpP schemes and argued that most of the NIST finalists fall into this category. Recall that CpP schemes first process the context (K, N, A) via $\textsc{Enc}_{\mathcal{C}}$ and afterwards process the result together with the message (and optionally the key K and nonce N) via $\textsc{Enc}_{\mathcal{M}}$ yielding the ciphertext. The classification of CpP schemes can be refined further, depending on whether $\textsc{Enc}_{\mathcal{M}}$ takes the key K and nonce N as input. We call schemes where $\textsc{Enc}_{\mathcal{M}}$ solely takes the output of $\textsc{Enc}_{\mathcal{C}}$ and the message M as input *full-CpP schemes*. Out of the NIST LWC finalists, Photon-Beetle and Xoodyak are full-CpP schemes, whereas for the others the key K is also input to $\textsc{Enc}_{\mathcal{M}}$ (and for Romulus additionally also the nonce N).[20] For CpP schemes, we define a game KeyColl, which asks the adversary to find a key collision for $\textsc{Enc}_{\mathcal{C}}$, i.e., two contexts that differ (at least) in the keys but result in the same output of $\textsc{Enc}_{\mathcal{C}}$.

Definition 3. Let Ae be a CpP scheme, with encryption composed of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$, and let the game KeyColl be defined as in Fig. 11. For any adversary $\mathcal{A}$, its KeyColl advantage is defined as

$$\mathbf{Adv}^{\mathsf{KeyColl}}_{\textsc{Enc}_{\mathcal{C}}}(\mathcal{A}) := \Pr[\mathsf{KeyColl}(\mathcal{A}) \to 1] \,.$$

For full-CpP schemes, key committing security can be lower bounded by finding a key collision for the pre-processing component $\textsc{Enc}_{\mathcal{C}}$. Any key collision attack against $\textsc{Enc}_{\mathcal{C}}$ translates to a key committing attack against the AE scheme—even when applying the zero-padding, regardless of how many zeroes are prepended to the message. This is formally stated in the theorem below, which is proven in Appendix C.5.

Theorem 5. *Let* Ae *be a full-CpP scheme, where encryption is decomposed into* $\textsc{Enc}_{\mathcal{C}}$ *and* $\textsc{Enc}_{\mathcal{M}}$*. Then, for any adversary $\mathcal{A}$ against* $\textsc{Enc}_{\mathcal{C}}$*, there exists an adversary $\mathcal{B}$ against* zp-Ae *such that*

$$\mathbf{Adv}^{\mathsf{CMT_K}}_{\text{zp-}\textsc{Ae}}(\mathcal{B}) \geq \mathbf{Adv}^{\mathsf{KeyColl}}_{\textsc{Enc}_{\mathcal{C}}}(\mathcal{A}) \,.$$

Having established Theorem 5 we show that Photon-Beetle and Xoodyak both allow for key collision attacks against the function $\textsc{Enc}_{\mathcal{C}}$ as stated in the following theorem.

Theorem 6. *Let* $\textsc{Ae} \in \{\textsc{Photon-Beetle}, \textsc{Xoodyak}\}$ *with underlying components* $\textsc{Enc}_{\mathcal{C}}$ *and* $\textsc{Enc}_{\mathcal{M}}$*. Let further ρ be modeled as an ideal permutation. Then there exists an adversary $\mathcal{A}$, making q queries to ρ, such that*

$$\mathbf{Adv}^{\mathsf{KeyColl}}_{\textsc{Enc}_{\mathcal{C}}}(\mathcal{A}) \geq \frac{1}{2} \,.$$

For Photon-Beetle*, we have $q = \alpha + \overline{\alpha}$. For* Xoodyak*, we have $q = 2^{17} + 1$.*

[20] There is no scheme for which the nonce is input to $\textsc{Enc}_{\mathcal{M}}$ while the key is not.

Proof. The proof follows from the proofs of Theorem 14 (Photon-Beetle) and Theorem 15 (Xoodyak). □

In combination, the above two theorems show that neither Photon-Beetle nor Xoodyak are CMT_{K} secure even when using the zero-padding approach.

4.3 Zero-Padding and Elephant

The following theorem shows that Elephant with zero-padding does not achieve CMT_{K} security provided that the number of padded zeros is smaller than the block size of Elephant. The proof can be found in Appendix C.6.

Theorem 7. *Consider* Elephant *which is illustrated and described in Fig. 18 and Fig. 19, respectively. Let the tweakable block-cipher* TEM *be modeled as an ideal tweakable cipher* $\widetilde{\mathsf{E}}$ *and let $z \leq n$. There exists an adversary $\mathcal{A}$, making q queries to $\widetilde{\mathsf{E}}$, such that*

$$\mathbf{Adv}^{\mathsf{CMT_K}}_{\text{ZP-Elephant}}(\mathcal{A}) = 1\,,$$

where $q = 2\mu + 2\gamma + \alpha + \overline{\alpha}$. Here, μ is the number of message blocks while computing $\text{Enc}_{\mathcal{M}}$ *(note that the block length is based on $0^z \parallel M$ and not just M) and γ is the number of ciphertext blocks while computing* $\text{Enc}_{\mathcal{T}}$*. Furthermore, α and $\overline{\alpha}$ are the number of associated data blocks for the two tuples that $\mathcal{A}$ outputs.*

Theorem 7 shows that zero-padding is pointless for Elephant if the number of prepended zeros is less than or equal to the block size (160/176/200 for the three parameter sets Jumbo/Dumbo/Delirium). To ensure a valid zero-padding for both tuples, the adversary needs to find (K, N) and $(\overline{K}, \overline{N})$ such that $\widetilde{\mathsf{E}}(K,(0,1),N) = \widetilde{\mathsf{E}}(\overline{K},(0,1),\overline{N})$ holds, which is trivial as $\widetilde{\mathsf{E}}$ is an ideal block-cipher for a fixed tweak—in this case $(0,1)$. The attack, however, does not apply anymore if the zero-padding affects more than one block. Assume that $z = 2n$, then $\mathcal{A}$ needs to find (K, N) and $(\overline{K}, \overline{N})$ such that both $\widetilde{\mathsf{E}}(K,(0,1),N) = \widetilde{\mathsf{E}}(\overline{K},(0,1),\overline{N})$ and $\widetilde{\mathsf{E}}(K,(0,2),N) = \widetilde{\mathsf{E}}(\overline{K},(0,2),\overline{N})$. By the same means as above, one can find (K, N) and $(\overline{K}, \overline{N})$ such that any one of the two equations holds but not necessarily both of them simultaneously. More generally, for $z = n + x$, the probability is $\frac{1}{2^x}$, i.e., the security increases with the number of padded zeros but only if the zero-padding exceeds the first block.

4.4 Zero-Padding and Isap

The following theorem (proven in detail in Appendix C.7) shows that Isap does not benefit from the zero-padding approach to increase its committing security. This is in sharp contrast to Ascon, for which Naito et al. [NSS23] showed that zero-padding indeed affects the committing security positively.

Theorem 8. *Consider* Isap *which is illustrated and described in Fig. 30 and Fig. 31, respectively. Let ρ_B and ρ_K be modeled by an ideal permutation ρ_1. Let further ρ_H and ρ_E be modeled by ideal permutations ρ_2 and ρ_3, respectively. Then there exists an adversary $\mathcal{A}$, making q_1, q_2, and q_3 queries to ρ_1, ρ_2, and ρ_3, respectively, such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\text{ZP-Isap}}(\mathcal{A}) \geq \frac{1}{2}\,,$$

where $q_1 = (2^{\frac{\tau}{2}+1} + 1)(\kappa + 1) + \nu$, $q_2 = (2^{\frac{\tau}{2}+1} + 1)(\alpha + \gamma + 2)$, and $q_3 = \mu$.

5 Conclusion

Out of the nine considered NIST finalists, we have shown that six do not achieve committing security while the remaining three do. For the former, we gave concrete attacks, while the others are backed up by formal security proofs. From the analysis, we identified three criteria that are related to committing security: (I) requires a tag length of at least 128 bits, (II) requires 128 bits of the state to be unaffected by the inputs, and (III) requires the initial state (for sponge-based AE schemes) to have 64 bits not affected by the inputs. We showed that the absence of these criteria often yields CMT attacks.

Overall, our analysis reveals that sponge constructions built on top of large permutations are favorable for committing security, as having 128 bits unaffected by the inputs (II) is then more easily achievable. Note that for sponges this should also hold for the initial state (III). Constructions built from block-ciphers often fail to achieve (II), as the entire state typically consists only of 128 bits—though the NIST workshop on accordion cipher modes [NIST24] also considers block-ciphers with 256-bit states which could solve the issue. On the other hand, a tag of at least 128 bits (I) is typically not a problem and can also be achieved by block-ciphers and permutations with smaller state sizes.

Finally, we show that for several of the NIST finalists, the problems that lead to committing attacks persist even when deploying the zero-padding method. Furthermore, we show that the interesting observation from [NSS23], i.e., zero-padding improves the committing security of Ascon, does not apply to Isap. Overall, this shows that the benefit of the zero-padding is rather restricted.

Acknowledgments

Work of Juliane Krämer was supported by the German Research Foundation (DFG) – SFB 1119 – 236615297. Patrick Struck acknowledges funding by the Bavarian State Ministry of Science and the Arts in the framework of the bidt Graduate Center for Postdocs (while working at Universität Regensburg) and the Hector Foundation II. Work of Maximiliane Weishäupl was funded by the German Federal Ministry of Education and Research (BMBF) under the project Quant-ID (16KISQ111).

References

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Heidelberg, February 2010.

[ADG+22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022*, pages 3291–3308. USENIX Association, August 2022.

[BBC+20] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Lightweight AEAD and hashing using the Sparkle permutation family. *IACR Trans. Symm. Cryptol.*, 2020(S1):208–261, 2020.

[BBdS+21] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, Amir Moradi, and Aein Rezaei Shahmirzadi. Schwaemm and Esch. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[BCD+21] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. PHOTON-Beetle. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[BCDM20] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Dumbo, Jumbo, and Delirium: Parallel authenticated encryption for the lightweight circus. *IACR Trans. Symm. Cryptol.*, 2020(S1):5–30, 2020.

[BCDM21] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[BCI+21] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[BDP+16] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Keyak v2. Technical report, Submission to the CAESAR Competition, 2016. Available at https://keccak.team/files/Keyakv2-doc2.2.pdf.

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

[BDPVA07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop*, 2007.

[Ber14] Daniel J. Bernstein. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. https://competitions.cr.yp.to/caesar.html, 2014.

[BH22] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 845–875. Springer, Heidelberg, May / June 2022.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, Heidelberg, August 2016.

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

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, Heidelberg, September 2017.

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023. Draft 0.6, http://toc.cryptobook.us/.

[CDNY18] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *IACR TCHES*, 2018(2):218–241, 2018. https://tches.iacr.org/index.php/TCHES/article/view/881.

[CIMN17] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 277–298. Springer, Heidelberg, September 2017.

[CR22] John Chan and Phillip Rogaway. On committing authenticated-encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part II*, volume 13555 of *LNCS*, pages 275–294. Springer, Heidelberg, September 2022.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. *IACR Trans. Symm. Cryptol.*, 2017(1):80–105, 2017.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symm. Cryptol.*, 2020(S1):390–416, 2020.

[DEM+21] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[DGL23] Orr Dunkelmann, Shibam Ghosh, and Eran Lambooij. Practical related-key forgery attacks on full-round TinyJAMBU-192/256. In *ToSC 2023 (2)*, 2023.

[DGRW18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Heidelberg, August 2018.

[DHAK18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of Xoodoo and Xoofff. *IACR Trans. Symm. Cryptol.*, 2018(4):1–38, 2018.

[DHP+21] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Silvia Mella. Xoodyak. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 209–240. Springer, Heidelberg, December 2019.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 225–255. Springer, Heidelberg, December 2019.

[DPU+16] Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov. Design strategies for ARX with provable bounds: Sparx and LAX. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 484–513. Springer, Heidelberg, December 2016.

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, Heidelberg, February / March 2013.

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

[GJMN16] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 263–293. Springer, Heidelberg, May 2016.

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 222–239. Springer, Heidelberg, August 2011.

[HJM+21] Martin Hell, Thomas Johansson, Willi Meier, Jonathan Sönnerup, Hirotaka Yoshida, and Alexander Maximov. Grain-128AEAD. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[IKM+21] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, Thomas Peyrin, and Chun Guo. Romulus. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. *IACR Trans. Symm. Cryptol.*, 2020(1):43–120, 2020.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 85–104. Springer, Heidelberg, December 2014.

[LGR21] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 195–212. USENIX Association, August 2021.

[LM24] Charlotte Lefevre and Bart Mennink. Generic security of the Ascon mode: On the power of key blinding. In *SAC 2024*, 2024.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002.

[MLGR23] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *LNCS*, pages 379–407. Springer, Heidelberg, April 2023.

[NIST15] National Institute of Standards and Technology. Lightweight cryptography standardization process. https://csrc.nist.gov/projects/lightweight-cryptography, 2015.

[NIST23] National Institute of Standards and Technology. The third NIST workshop on block cipher modes of operation 2023. https://csrc.nist.gov/projects/lightweight-cryptography, 2023. Accessed: 2024-02-23.

[NIST24] National Institute of Standards and Technology. NIST workshop on the requirements for an accordion cipher mode 2024. https://csrc.nist.gov/Events/2024/accordion-cipher-mode-workshop-2024, 2024. Accessed: 2024-04-04.

[NO14] Yusuke Naito and Kazuo Ohta. Improved indifferentiable security analysis of PHOTON. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 340–357. Springer, Heidelberg, September 2014.

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

[NSS23] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Commiting security of Ascon: Cryptanalysis on primitive and proof on mode. In *ToSC 2023 (4)*, 2023.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[SW24] Patrick Struck and Maximiliane Weishäupl. Constructing committing and leakage-resilient authenticated encryption. In *ToSC 2024 (1)*, 2024.

[TMC+23] Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, Lawrence E. Bassham, Jinkeon Kang, Noah D. Waller, John M. Kelsey, and Deukjo Hong. Status report on the final round of the NIST lightweight cryptography standardization process. https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8454.pdf, 2023.

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

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

[WH16] Hongjun Wu and Tao Huang. The JAMBU lightweight authentication encryption mode (v2.1). Technical report, Submission to the CAESAR Competition, 2016. Available at https://competitions.cr.yp.to/round3/jambuv21.pdf.

[WH21] Hongjun Wu and Tao Huang. TinyJambu. Technical report, National Institute of Standards and Technology, 2021. Available at https://csrc.nist.gov/projects/lightweight-cryptography/finalists.

A Additional Preliminaries

In Appendix A.1, we give additional details about the NIST LWC finalists and in Appendix A.2 we describe the used paddings and define more security notions. We give some background on (tweakable) block-ciphers and sponges in Appendix A.3 and Appendix A.4, respectively. Finally, in Appendix A.5, we provide some results that are relevant for our attacks and proofs.

A.1 The NIST LWC Finalists

We provide two tables and a figure with additional information about the NIST LWC finalists regarding their similarities and differences. Table 2 gives an overview and Table 3 describes the parameter sets. Fig. 12 shows the state-update-function for the different schemes.

Table 2: Overview of the NIST LWC finalists regarding similarities and differences in their design. Here, CpP stands for Context-pre-Processing and EtM for Encrypt-then-MAC.

Scheme	Class of scheme	Underlying Primitive	Sponge	State-update-function
Romulus	CpP	Block-cipher[a]	No	Yes
Elephant	EtM	Permutation	No	No
Gift-Cofb	CpP	Block-cipher	No	Yes
Photon-Beetle	CpP	Permutation	Yes	Yes
TinyJambu	CpP	Block-cipher	Yes	No
Xoodyak	CpP	Permutation	Yes	No
Ascon	CpP	Permutation	Yes	No
Isap	EtM	Permutation	Yes	No
Schwaemm	CpP	Permutation	Yes	Yes

[a] Note that Romulus actually uses a tweakable block-cipher for its underlying primitive.

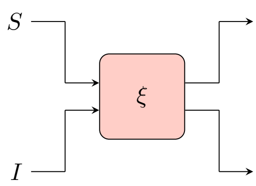
A.2 Paddings and Security Notions

The authenticated encryption schemes considered in this work, use common paddings which we recall below. The one-zero padding $\mathtt{pad}_{10^*}(\cdot, r)$, appends a 1, followed by 0s until the length is a multiple of r. Simply padding with 0s to a length of a multiple of r is denoted by $\mathtt{pad}_{0^*}(\cdot, r)$. By $\mathtt{pad}_{\mathtt{L}}(\cdot, r)$, we denote the padding which appends 0, followed by appending the length of the input such that the overall length is a multiple of r.

Below we define collision resistance of a hash function.

Table 3: Parameters of the NIST LWC finalists. Values for rate and capacity are only given for the sponge-based schemes. For Isap, the parameters are for the version using Keccak-P, the version using Ascon-P has $n = 320$ and $r = 64$. Note that for Xoodyak and Isap, components of the schemes use rates deviating from the ones given above: The $\text{Enc}_{\mathfrak{C}}$ component in Xoodyak is a full-state sponge and in Isap's re-keying mechanism a minimal rate of 1 is used.

Scheme	Key κ	Nonce ν	Tag τ	State n	Rate r	Capacity c
Romulus	128	128	128	128	-	-
Elephant	128	96	64	160	-	-
Gift-Cofb	128	128	128	128	-	-
Photon-Beetle	128	128	128	256	128	128
TinyJambu	128	96	64	128	32	96
Xoodyak	128	128	128	384	192	192
Ascon	128	128	128	320	64	256
Isap	128	128	128	400	144	256
Schwaemm	128	256	128	384	256	128



$$Y = \begin{cases} S \oplus I & \text{(Romulus)} \\ \widetilde{G}(S) \oplus I & \text{(Gift-Cofb)} \\ S \oplus I & \text{(Photon-Beetle)} \\ \mathsf{FeistelSwap}(S) \oplus I & \text{(Schwaemm)} \end{cases}$$

$$O = \begin{cases} G(S) \oplus I & \text{(Romulus)} \\ S \oplus I & \text{(Gift-Cofb)} \\ \mathsf{Shuffle}(S) \oplus I & \text{(Photon-Beetle)} \\ S \oplus I & \text{(Schwaemm)} \end{cases}$$

Figure 12: Illustration of the state-update-function ξ for the different schemes. The components G, $\widetilde{G}$, $\mathsf{Shuffle}$, and $\mathsf{FeistelSwap}$—if relevant for our results—are described along with the schemes in the respective sections.

Definition 4. Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^w$ be a hash function outputting bit strings of length w. For any adversary $\mathcal{A}$, its CR advantage is defined as

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}) \coloneqq \Pr[\mathcal{H}(X_1) = \mathcal{H}(X_2) \wedge X_1 \neq X_2 \mid (X_1, X_2) \leftarrow \mathcal{A}()] .$$

Menda et al. [MLGR23] defined several variants of committing security. These variants require different parts of the contexts to disagree (and sometimes also others to agree). Below we recall their security notions.[21]

Definition 5. Let $\text{Ae} = (\text{Enc}, \text{Dec})$ be an authenticated encryption scheme and the games $\mathsf{CMT}_{\mathsf{X}}$ and $\mathsf{CMT}^{\star}_{\mathsf{X}}$ for $\mathsf{X} \in \{\mathsf{K}, \mathsf{N}, \mathsf{A}\}$ be defined as in Fig. 1. For any adversary $\mathcal{A}$, its CMT_{X} and $\text{CMT}^{\star}_{\mathsf{X}}$ advantages are defined as

$$\mathbf{Adv}^{\mathsf{CMT}_{\mathsf{X}}}_{\text{Ae}}(\mathcal{A}) \coloneqq \Pr[\mathsf{CMT}_{\mathsf{X}}(\mathcal{A}) \to 1] \quad \text{and} \quad \mathbf{Adv}^{\mathsf{CMT}^{\star}_{\mathsf{X}}}_{\text{Ae}}(\mathcal{A}) \coloneqq \Pr[\mathsf{CMT}^{\star}_{\mathsf{X}}(\mathcal{A}) \to 1] ,$$

respectively.

Furthermore, Menda et al. [MLGR23] introduce so-called *context-discovery* notions. In this work, we are only interested in the notion $\text{CDY}^{\star}_{A}$. Roughly speaking, this notion provides an adversary with a ciphertext C and partial context (consisting of the key K

[21] Note that CMT_{K} originates from [BH22].

Game $\mathsf{CMT_K}$

$(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}()$
if $K = \overline{K}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game $\mathsf{CMT}^{\star}_{\mathsf{K}}$

$((K, \overline{K}), N, A, (M, \overline{M})) \leftarrow \mathcal{A}()$
if $K = \overline{K}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, N, A, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game $\mathsf{CMT_N}$

$(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}()$
if $N = \overline{N}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game $\mathsf{CMT}^{\star}_{\mathsf{N}}$

$(K, (N, \overline{N}), A, (M, \overline{M})) \leftarrow \mathcal{A}()$
if $N = \overline{N}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(K, \overline{N}, A, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game $\mathsf{CMT_A}$

$(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}()$
if $A = \overline{A}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Game $\mathsf{CMT}^{\star}_{\mathsf{A}}$

$(K, N, (A, \overline{A}), (M, \overline{M})) \leftarrow \mathcal{A}()$
if $A = \overline{A}$
  return 0
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(K, N, \overline{A}, \overline{M})$
return $((C, T) = (\overline{C}, \overline{T}))$

Figure 13: Security games $\mathsf{CMT_K}$, $\mathsf{CMT_N}$, $\mathsf{CMT_A}$, $\mathsf{CMT}^{\star}_{\mathsf{K}}$, $\mathsf{CMT}^{\star}_{\mathsf{N}}$, and $\mathsf{CMT}^{\star}_{\mathsf{A}}$ for authenticated encryption schemes. Here, $((K, \overline{K}), N, A, (M, \overline{M}))$ is an abbreviation for $(K, N, A, M), (\overline{K}, N, A, \overline{M})$, likewise used for the other context components.

and nonce N used to generate the ciphertext), and the adversary is challenged to find associated data A which validly decrypts the ciphertext. Below we recall this security notion.

Definition 6. Let $\textsc{Ae} = (\textsc{Enc}, \textsc{Dec})$ be an authenticated encryption scheme and the game $\mathsf{CDY}^{\star}_{A}$ be defined as in Fig. 14. For any adversary $\mathcal{A}$, its $\mathrm{CDY}^{\star}_{A}$ advantage is defined as

$$\mathbf{Adv}^{\mathsf{CDY}^{\star}_{A}}_{\textsc{Ae}}(\mathcal{A}) \coloneqq \Pr[\mathsf{CDY}^{\star}_{A}(\mathcal{A}) \to 1]\,.$$

Next, we define the advantage of finding colliding tags. At its core, this is a weakened version of committing security as the ciphertexts are not required to agree. We use this to bound the committing security of Ascon and Schwaemm.

Definition 7. Let $\textsc{Ae} = (\textsc{Enc}, \textsc{Dec})$ be an authenticated encryption scheme and the game $\mathsf{TagColl}$ be defined as in Fig. 15. For any adversary $\mathcal{A}$, its TagColl advantage is defined as

$$\mathbf{Adv}^{\mathsf{TagColl}}_{\textsc{Ae}}(\mathcal{A}) \coloneqq \Pr[\mathsf{TagColl}(\mathcal{A}) \to 1]\,.$$

Game $\mathsf{CDY}^{\star}_{A}$

$$
\begin{array}{l}
(K, N, (C, T)) \leftarrow\!\!\$\, \mathcal{K} \times \mathcal{N} \times \mathcal{C} \\
A \leftarrow \mathcal{A}(K, N, (C, T)) \\
M \leftarrow \textsc{Dec}(K, N, A, (C, T)) \\
\textbf{if } M = \bot \\
\quad \textbf{return } 0 \\
\textbf{return } 1
\end{array}
$$

Figure 14: Security game $\mathsf{CDY}^{\star}_{A}$ for authenticated encryption schemes.

Game $\mathsf{TagColl}$

$$
\begin{array}{l}
(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M}) \leftarrow \mathcal{A}() \\
\textbf{if } (K, N, A) = (\overline{K}, \overline{N}, \overline{A}) \\
\quad \textbf{return } 0 \\
(C, T) \leftarrow \textsc{Enc}(K, N, A, M) \\
(\overline{C}, \overline{T}) \leftarrow \textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) \\
\textbf{return } (T = \overline{T})
\end{array}
$$

Figure 15: Security game $\mathsf{TagColl}$ for authenticated encryption schemes used in the proof of Theorem 4.

A.3 (Tweakable) Block-Ciphers

A block-cipher BC: $\{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ takes as input a key K of length κ and a message M of length n, and outputs a ciphertext C of the same length as the message. For every $K \in \{0,1\}^\kappa$, $\mathrm{BC}(K, \cdot)$ is a permutation over $\{0,1\}^n$. A tweakable block-cipher [LRW02] TBC: $\{0,1\}^\kappa \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ takes as input a key K of length κ, a tweak W (from some set of tweaks $\mathcal{T}$), and a message M of length n, and outputs a ciphertext C of the same length as the message. For every pair $(K, W) \in \{0,1\}^\kappa \times \mathcal{T}$, $\mathrm{TBC}(K, W, \cdot)$ is a permutation over $\{0,1\}^n$. We also use $\mathrm{TBC}_K^W(\cdot)$ as an alternative notation for $\mathrm{TBC}(K, W, \cdot)$.

For our results, we model the block-ciphers BC and tweakable block-ciphers TBC by an ideal cipher E and ideal tweakable cipher $\widetilde{\mathsf{E}}$, respectively.

A.4 Sponges

Sponges [BDPVA07] are a versatile tool for cryptographic primitives. Rather than just being relevant for cryptographic hash functions—as was their initial design goal—they turned out to be more powerful as one can construct numerous cryptographic primitives from sponges.

The underlying component of a sponge is a permutation $\rho\colon \{0,1\}^n \to \{0,1\}^n$. Here, n is the size of the sponge state. The sponge operates in a round-wise fashion, where each round it absorbs a part of the input and applies ρ. The rate r describes how many bits of the input can be absorbed in each round by XORing them to the first r bits of the sponge state. The higher the rate the faster the sponge as fewer rounds, hence fewer invocations of ρ, are required to absorb the input. The part of the sponge state that is not affected by the input absorption is called the inner state and its size is denoted by the capacity c, thus we have $r + c = n$. The capacity is related to the security of the sponge, the higher the capacity the better the security of the sponge. For several of the NIST LWC finalists, the capacity of the initial state differs from the one in the rest of the sponge, which is why

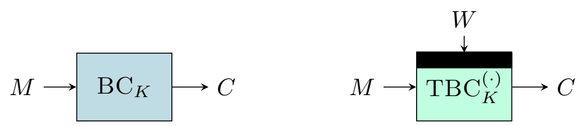


Figure 16: Block-cipher (left) and tweakable block-cipher (right). For tweakable block-ciphers, the black bar indicates that the incoming arrow (W) is the tweak.

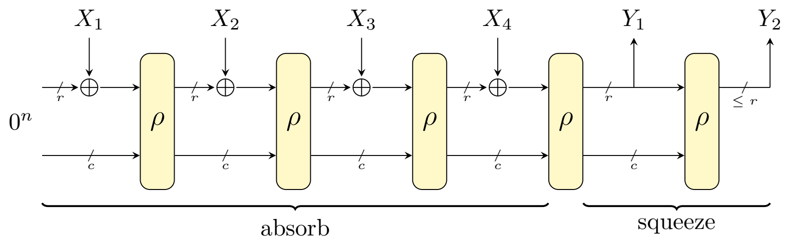


Figure 17: Illustration of a plain sponge construction with four rounds of absorbing and two rounds of squeezing.

we introduce $c^\star$ as a notation for this initial capacity.

We refer to sponges of the form described above by *plain sponges* and provide an illustration in Fig. 17. It was shown that—especially in the context of AE schemes—one can also deploy *full-state sponges* and *duplex sponges*. The former XORs the input to the entire state, i.e., $r = n$ and $c = 0$. The latter absorbs and squeezes in each round, in contrast to the plain sponge which squeezes only after the absorption is finished. Xoodyak uses a full-state sponge, while a duplex sponge is used, for instance, by Ascon.

Below we recall two results for the plain sponge construction that we will use later: First, a bound on the collision resistance of a simple sponge-based hash function and, second, the indifferentiability of sponges from a random function.

Theorem 9 ([BS23, Theorem 8.6])**.** *Let $\mathcal{H}$ be a hash function obtained from a permutation $\rho\colon \{0,1\}^n \to \{0,1\}^n$, with capacity c, rate r (so $n = r + c$), and output length $w \leq r$. For every adversary $\mathcal{A}$, if the number of ideal permutation queries plus the number of r-bit blocks in the output of $\mathcal{A}$ is bounded by q, it holds that*

$$\mathbf{Adv}_H^{\mathsf{CR}}(\mathcal{A}) \leq \frac{q(q-1)}{2^w} + \frac{q(q+1)}{2^c} .$$

Theorem 10 ([BDPV08, Theorem 2])**.** *Let $\mathcal{H}$ be a (padded) sponge construction obtained from a permutation $\rho\colon \{0,1\}^n \to \{0,1\}^n$, with capacity c and rate r (so $n = r + c$). Then, for any adversary $\mathcal{A}$, making significantly less than 2^c queries to ρ, $\mathcal{H}$ is indistinguishable from a random oracle* F*, except with probability at most* $\frac{(1-2^{-256})q^2+(1+2^{-256})q}{2^{129}}$.

An improved indifferentiability bound is given by Naito et al. [NO14]. This results allow for a smaller capacity in both the initial state of the sponge ($c^\star = \frac{c}{2}$ instead of $c^\star = c$) and the squeezing phase ($c^\star = \frac{c}{2} + \log_2 c$ instead of $c^\star = c$) while maintaining the security bound. Using this improved indifferentiability one can also obtain committing security of Ascon (in contrast to our analysis which uses collision resistance of plain sponges). For Isap and (our modified version of) Schwaemm, the older indifferentiability bound is sufficient as they both feature a larger IV.

A.5 Existing Results

The theorem below gives both upper and lower bounds on finding collisions for independent random variables.

Theorem 11 ([BS23, Theorem B.1])**.** *Let $\mathcal{M}$ be a set of size n and $X_1, \ldots, X_k$ be k independent random variables uniform in $\mathcal{M}$. Let C be the event that for some distinct $i, j \in \{1, \ldots, k\}$ we have that $X_i = X_j$. Then*

$$\Pr[\mathsf{C}] \geq 1 - \exp\left(\frac{-k(k-1)}{2n}\right) \geq \min\left\{\frac{-k(k-1)}{4n}, 0.63\right\} \text{ and}$$
$$\Pr[\mathsf{C}] \leq 1 - \exp\left(\frac{-k(k-1)}{n}\right) \text{ when } k < \frac{n}{2}\,.$$

We use the formulation of the birthday problem presented in [Wag02], but provide a more formal description using Theorem 11.

Lemma 5. *Consider two lists L_1, L_2 of elements drawn uniformly and independently at random from $\{0,1\}^\tau$. We denote the size of L_1 and L_2 by l_1 and l_2, respectively, and define $l = l_1 + l_2$. Then one finds $x_k \in L_1$ and $x_j \in L_2$ such that $x_k \oplus x_j = 0$ with a probability of at least*

$$\left(1 - \exp\left(\frac{-l(l-1)}{2^{\tau+1}}\right)\right) \cdot \frac{2\,l_1 l_2}{l^2 - l}\,.$$

Proof. Consider the concatenation of the two lists $L = L_1 \parallel L_2$ and write $x_1, \ldots, x_l$ for its elements. Denote by C the event that $x_k = x_j$ holds for some $k \neq j$. Since the x_i are drawn uniformly and independently, Theorem 11 yields the bound $\Pr[\mathsf{C}] \geq 1 - \exp\left(\frac{-l(l-1)}{2^{\tau+1}}\right)$. However, this probability also counts internal collisions of L_1 and L_2, respectively. For two elements of L, the probability that they are not both from either L_1 or L_2 is $\frac{l_1}{l}\frac{l_2}{l-1} + \frac{l_2}{l}\frac{l_1}{l-1} = \frac{2\,l_1 l_2}{l^2-l}$. Taking this into account, the probability that $x_k = x_j$ holds for some $x_k \in L_1$ and $x_j \in L_2$ is thus bounded above by $\left(1 - \exp\left(\frac{-l(l-1)}{2^{\tau+1}}\right)\right) \cdot \frac{2\,l_1 l_2}{l^2-l}$. $\square$

The following lemma contains two technical results needed for the committing security proof of Ascon (Theorem 4). While the computations are not hard, we give them here to provide a complete presentation.

Lemma 6. *Let $n, c, l \in \mathbb{N}$ such that $c, l \leq n$ and $\mathsf{IV} \in \{0,1\}^l$. Let further ρ be a random permutation over $\{0,1\}^n$ and $\mathcal{A}$ be an adversary making queries to ρ. Consider the following events:*

1. *Event $\mathsf{E_t}$ (target hitting query):*

   $\mathcal{A}$ makes a query Y to ρ such that $\lfloor \rho(Y) \rfloor_l = \mathsf{IV}$ or $\mathcal{A}$ makes a query S to ρ^{-1} such that $\lfloor \rho^{-1}(S) \rfloor_l = \mathsf{IV}$.

2. *Event $\mathsf{E_c}$ (colliding queries):*

   $\mathcal{A}$ makes queries $Y \neq \overline{Y}$ to ρ such that $\lfloor \rho(Y) \rfloor_c = \lfloor \rho(\overline{Y}) \rfloor_c$ or $\mathcal{A}$ makes queries Y to ρ and $\overline{S}$ to ρ^{-1} such that $\lfloor \rho(Y) \rfloor_c = \lfloor \rho^{-1}(\overline{S}) \rfloor_c$.

If $\mathcal{A}$ makes $q \leq 2^{n-1}$ queries, then

$$\Pr[\mathsf{E_t}] \leq \frac{q}{2^{l-1}} \quad \text{and} \quad \Pr[\mathsf{E_c}] \leq \frac{q(q-1)}{2^c}\,.$$

Proof. We start with the bound for $\mathsf{E_t}$. Let $\mathsf{X_i}$ be the event that $\mathcal{A}$ triggers event $\mathsf{E_t}$ with its i-t query. It holds that

$$\Pr[\mathsf{E_t}] \leq \sum_{i=1}^{q} \Pr[\mathsf{X_i}] = \sum_{i=1}^{q} \frac{2^{n-l}}{2^n - i + 1} \leq \frac{2^{n-l} q}{2^n - q} \leq \frac{2^{n-l} q}{2^{n-1}} = \frac{q}{2^{l-1}}\,,$$

where $q \leq 2^{n-1}$ is used for the last inequality. Next, we bound event $\mathsf{E_c}$. Let $\mathsf{X_{ij}}$ be the event that the j-th query by $\mathcal{A}$ forms a collision with the i-th query. Then it holds that

$$\Pr[\mathsf{E_c}] \leq \sum_{j=i}^{q} \sum_{i=1}^{j-1} \Pr[\mathsf{X_{ij}}] \leq \sum_{j=1}^{q} \frac{(j-1)2^{n-c}}{2^n - j + 1} \leq 2^{n-c} \left(\frac{q(q-1)}{2\,(2^n - q)} \right),$$

and using $q \leq 2^{n-1}$ again, we obtain

$$\Pr[\mathsf{E_c}] \leq 2^{n-c} \left(\frac{q(q-1)}{2^n} \right) = \frac{q(q-1)}{2^c},$$

which finishes the proof. □

B Additional Committing Security Analysis

Here, we analyze the CMT security of the six remaining NIST LWC finalists. For Elephant (cf. Appendix B.1), Gift-Cofb (cf. Appendix B.2), and Photon-Beetle (cf. Appendix B.3), we give—similar to Romulus (cf. Section 3.1)—attacks that break committing security with essentially no cost. For Xoodyak (cf. Appendix B.4), we give a committing attack requiring about 2^{17} queries using a birthday attack as done for TinyJambu (cf. Section 3.2). For Isap (cf. Appendix B.5) and Schwaemm (cf. Appendix B.6), we give formal proofs showing that the schemes achieve committing security of about 64-bit, similar to Ascon (cf. Section 3.3).

B.1 Elephant

The AE scheme Elephant [BCDM21, BCDM20] relies on a cryptographic permutation which gets masked using linear feedback shift registers similar to the masked Even-Mansour construction [GJMN16]. The security of the Elephant-mode follows from the security of the masked Even-Mansour construction, which is shown to be indistinguishable from an ideal tweakable block-cipher [BCDM21]. As our results rely on the mode, we choose to present the scheme in terms of such an tweakable block-cipher, which we denote by TEM for tweakable Even-Mansour.

B.1.1 Description of Elephant

The pseudocode of Elephant is given in Fig. 19 and further illustration is provided in Fig. 18. Elephant follows the Encrypt-then-MAC paradigm, i.e., it first encrypts the message $C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, M)$ and afterwards computes the tag $T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$. Note that in $\textsc{Enc}_{\mathcal{T}}$ the nonce and associated data are padded together, i.e., the first associated data block contains the nonce and the first bits of the associated data. This is in contrast to all other schemes, where the associated data blocks do not contain the nonce. Furthermore, note that the underlying encryption is an involution, i.e., to decrypt a ciphertext, we simply compute $\textsc{Enc}_{\mathcal{M}}(K, N, C)$.

B.1.2 Committing Attack against Elephant

Since Elephant follows the EtM-paradigm, we only need to focus on the underlying function $\textsc{Enc}_{\mathcal{T}}$. If we can find two contexts that verify a ciphertext-tag pair, applying $\textsc{Enc}_{\mathcal{M}}$ to the ciphertext and each context, gives back two valid messages. The following attack, which is the simplest one in this work, shows that Elephant [BCDM21] does not achieve committing security.

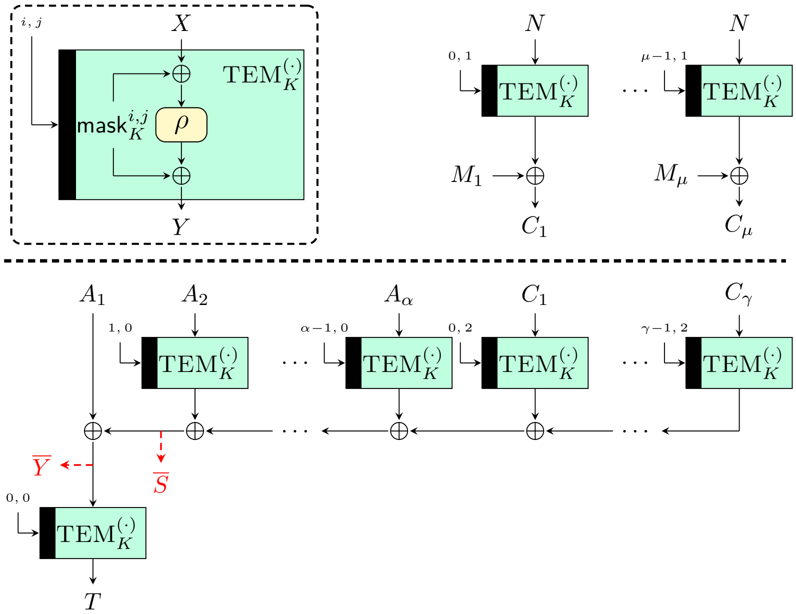


Figure 18: Illustration of Elephant in terms of $\textsc{Enc}_{\mathcal{M}}$ (top) and $\textsc{Enc}_{\mathcal{T}}$ (bottom). The states $\overline{S}$ and $\overline{Y}$, marked in red, are used in our CMT attack.

Theorem 12. *Consider* Elephant *which is illustrated and described in Fig. 18 and Fig. 19, respectively. Let* TEM *be modeled as an ideal tweakable cipher* $\widetilde{\mathsf{E}}$. *Then there exists an adversary* $\mathcal{A}$, *making* q *queries to* $\widetilde{\mathsf{E}}$, *such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Elephant}}(\mathcal{A}) = 1\,,$$

where $q = 2\mu + 2\gamma + \alpha + \overline{\alpha}$. *Here,* μ *is the number of message blocks while computing* $\textsc{Enc}_{\mathcal{M}}$ *and* γ *is the number of ciphertext blocks while computing* $\textsc{Enc}_{\mathcal{T}}$.[22] *Furthermore,* α *and* $\overline{\alpha}$ *are the number of associated data blocks for the two tuples that* $\mathcal{A}$ *outputs.*

Proof. We construct a CMT adversary $\mathcal{A}$ against Elephant as shown in Fig. 20. As a first step it samples a key K, a nonce N, associated data A, and a message M at random from the respective sets. It computes the ciphertext $C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, M)$ and the tag $T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$. The ciphertext is parsed into blocks $C_1, \ldots, C_\gamma \xleftarrow{n} \mathtt{pad}_{10^*}(C, n)$. Next, the adversary samples a second, different key $\overline{K} \leftarrow\!\!\$\ \mathcal{K}\backslash\{K\}$ and associated data blocks $\overline{A}_2, \ldots, \overline{A}_{\overline{\alpha}} \leftarrow\!\!\$\ \{0,1\}^n$.[23] The adversary then computes the state

$$\overline{S} \leftarrow \bigoplus_{i=2}^{\overline{\alpha}} (\widetilde{\mathsf{E}}(\overline{K}, (i-1, 0), \overline{A}_i)) \oplus \bigoplus_{i=1}^{\gamma} (\widetilde{\mathsf{E}}(\overline{K}, (i-1, 2), C_i))\,,$$

shown in Fig. 18. The value $\overline{Y}$ is computed by querying $\widetilde{\mathsf{E}}^{-1}$ on $\overline{K}$, $(0, 0)$, and T (padded with 0s to length n). Adversary $\mathcal{A}$ then computes $\overline{A}_1$ as the XOR of $\overline{S}$ and $\overline{Y}$. Together with the other associated data blocks, $\mathcal{A}$ computes $(\overline{N}, \overline{A}) \leftarrow \mathtt{pad}^{-1}_{10^*}(\overline{A}_1 \parallel \ldots \parallel \overline{A}_{\overline{\alpha}})$, i.e., removes the padding. It remains to compute the message $\overline{M}$ to which the ciphertext C decrypts under the context $(\overline{K}, \overline{N}, \overline{A})$. This can easily be achieved by setting $\overline{M} \leftarrow \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, C)$. Finally, $\mathcal{A}$ outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. Observe that $\mathcal{A}$ wins the game CMT, as we have

$$\textsc{Elephant}.\textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) = (C, T) = \textsc{Elephant}.\textsc{Enc}(K, N, A, M)\,.$$

[22] Note that μ and γ might not be the same.
[23] We assume that $\mathcal{A}$ chooses the last block to exhibit a valid padding.

$\textsc{Elephant}.\textsc{Enc}(K, N, A, M)$

$C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, M)$
$T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$
$\textbf{return } (C, T)$

$\textsc{Enc}_{\mathcal{M}}(K, N, M)$

$M_1, \ldots, M_\mu \xleftarrow{n} \mathtt{pad}_{0^*}(M, n)$
$\textbf{for } i = 1, \ldots, \mu$
$\quad C_i \leftarrow M_i \oplus \mathrm{TEM}^{(i-1,1)}(K, N)$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
$\textbf{return } C$

$\textsc{Enc}_{\mathcal{T}}(K, N, A, C)$

$A_1, \ldots, A_\alpha \xleftarrow{n} \mathtt{pad}_{10^*}(N \parallel A, n)$
$C_1, \ldots, C_\gamma \xleftarrow{n} \mathtt{pad}_{10^*}(C, n)$
$T \leftarrow A_1$
$\textbf{for } i = 2, \ldots, \alpha$
$\quad T \leftarrow T \oplus \mathrm{TEM}^{(i-1,0)}(K, A_i)$
$\textbf{for } i = 1, \ldots, \gamma$
$\quad T \leftarrow T \oplus \mathrm{TEM}^{(i-1,2)}(K, C_i)$
$T \leftarrow \mathrm{TEM}^{(0,0)}(K, T)$
$\textbf{return } \lceil T \rceil_\tau$

Figure 19: Pseudocode of Elephant [BCDM21] in terms of $\textsc{Enc}_{\mathcal{M}}$ and $\textsc{Enc}_{\mathcal{T}}$.

$\textsc{Elephant}$ adversary $\mathcal{A}$

$K, N, A, M \leftarrow\!\!\$\; \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$
$C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, M)$
$T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$
$C_1, \ldots, C_\gamma \xleftarrow{n} \mathtt{pad}_{10^*}(C, n)$
$\overline{K} \leftarrow\!\!\$\; \mathcal{K} \setminus \{K\}$
$\overline{A}_2, \ldots, \overline{A}_{\overline{\alpha}} \leftarrow\!\!\$\; \{0,1\}^n$
$\overline{S} \leftarrow \mathcal{B}(C_1, \ldots, C_\gamma, \overline{A}_2, \ldots, \overline{A}_{\overline{\alpha}})$
$\overline{Y} \leftarrow \widetilde{\mathsf{E}}^{-1}(\overline{K}, (0,0), \mathtt{pad}_{0^*}(T, n))$
$\overline{A}_1 \leftarrow \overline{Y} \oplus \overline{S}$
$(\overline{N}, \overline{A}) \leftarrow \mathtt{pad}^{-1}_{10^*}(\overline{A}_1, \ldots, \overline{A}_{\overline{\alpha}})$
$\overline{M} \leftarrow \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, C)$
$\textbf{return } (K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$

$\mathcal{B}(C_1, \ldots, C_\gamma, \overline{A}_2, \ldots, \overline{A}_{\overline{\alpha}})$

$\overline{S} \leftarrow 0^n$
$\textbf{for } i = 1, \ldots, \gamma$
$\quad \overline{S} \leftarrow \overline{S} \oplus \widetilde{\mathsf{E}}^{-1}(\overline{K}, (i-1, 2), C_i)$
$\textbf{for } i = 2, \ldots, \overline{\alpha}$
$\quad \overline{S} \leftarrow \overline{S} \oplus \widetilde{\mathsf{E}}^{-1}(\overline{K}, (i-1, 0), \overline{A}_i)$
$\textbf{return } \overline{S}$

Figure 20: Elephant adversary $\mathcal{A}$ from Theorem 12.

As for the queries to $\widetilde{\mathsf{E}}$, $\mathcal{A}$ makes μ queries to compute C and $\alpha + \gamma$ to compute T. Additionally, $\mathcal{A}$ makes μ queries to compute $\overline{M}$ and $\overline{\alpha} + \gamma$ queries to compute $\overline{S}$ and $\overline{Y}$, totalling up to $q = 2\mu + 2\gamma + \alpha + \overline{\alpha}$ queries. □

The attack easily extends to $\mathrm{CMT}_{\mathsf{K}}$, $\mathrm{CMT}_{\mathsf{N}}$, $\mathrm{CMT}_{\mathsf{A}}$, and $\mathrm{CMT}^{\star}_{\mathsf{A}}$ attacks. The reasoning follows the one given for Romulus. For a $\mathrm{CMT}^{\star}_{\mathsf{A}}$, the attack needs to target a different associated data block than the first as this one contains the nonce.

B.2 Gift-Cofb

The AE scheme Gift-Cofb uses the Cofb mode [CIMN17] for authenticated encryption and instantiates the block-cipher using Gift [BPP+17].

B.2.1 Description of Gift-Cofb

The pseudocode of Gift-Cofb is given in Fig. 22 while Fig. 21 provides an illustration of it. Gift-Cofb follows the CpP-approach, i.e., it first computes $S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$

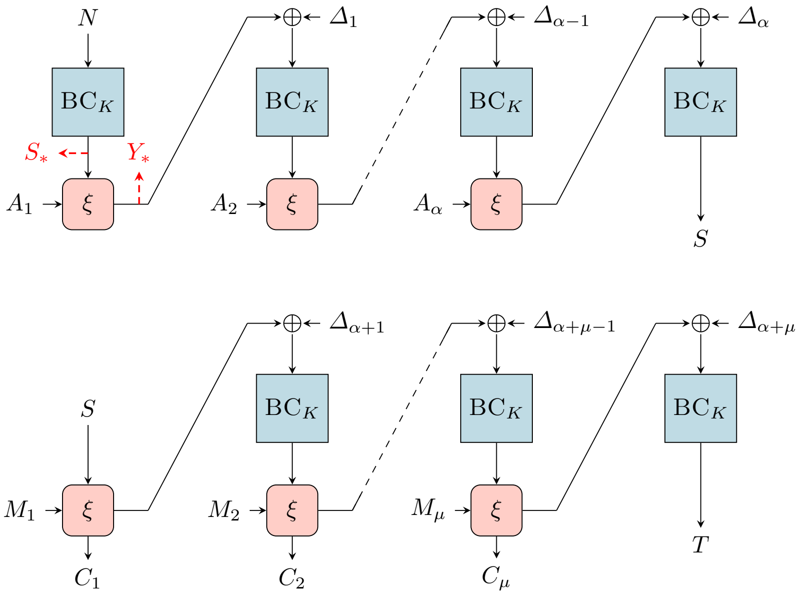


Figure 21: Illustration of Gift-Cofb in terms of $\textsc{Enc}_{\mathcal{C}}$ (top) and $\textsc{Enc}_{\mathcal{M}}$ (bottom). The different indices for Δ only indicate that the values are different—in the pseudocode the value Δ is constantly updated. The states S_* and Y_*, marked in red, are used in our CMT attack.

followed by $(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, S, M)$. The scheme features the state-update-function

$$\xi : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n,\ \xi(S, I) = (\widetilde{G}(S) \oplus I, S \oplus I)$$

that is invoked in an alternating manner with the block-cipher. This function makes use of the following matrix $\widetilde{G}$, which gets a bit string of length n, swaps the two halves, and additionally applies a bit rotation to the (new) second half:

$$\widetilde{G}\colon \{0,1\}^{n/2} \times \{0,1\}^{n/2} \to \{0,1\}^{n/2} \times \{0,1\}^{n/2},\ (S_1, S_2) \qquad \mapsto (S_2, S_1 \lll 1)\,.$$

In between the state-update-function and the block-cipher, Gift-Cofb applies some masking by XORing some value Δ to the state.

B.2.2 Committing Attack Against Gift-Cofb

The AE scheme Gift-Cofb does not achieve CMT security. The scheme is very similar to Romulus which allows to apply the same attack strategy. However, Gift-Cofb uses a different state-update-function ξ. It turns out that we cannot always invert ξ—inversion only works with probability $\frac{1}{2}$. This is the reason, why the advantage depends on the number of ciphertext blocks.

Theorem 13. *Consider* Gift-Cofb *which is illustrated and described in Fig. 21 and Fig. 22, respectively. Let* BC *be modeled as an ideal cipher* E*. Then there exists an adversary* $\mathcal{A}$*, making* q *queries to* E*, such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Gift-Cofb}}(\mathcal{A}) = \frac{1}{2^\mu}\,,$$

where $q = 2\mu + \alpha + \overline{\alpha} + 2$*. Here,* μ *is the number of message blocks,* α *is the number of associated data blocks for the first tuple, and* $\overline{\alpha}$ *for the second tuple that* $\mathcal{A}$ *outputs.*

$\underline{\textsc{Gift-Cofb}.\textsc{Enc}(K, N, A, M)}$

$(S, \Delta) \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, (S, \Delta), M)$
return (C, T)

$\underline{\textsc{Enc}_{\mathcal{M}}(K, (S, \Delta), M)}$

$M_1, \ldots, M_\mu \xleftarrow{n} \mathtt{pad}_{10^*}(M, n)$
for $i = 1, \ldots, \mu - 1$
  $\Delta \leftarrow 2\Delta$
  $(Y, C_i) \leftarrow \xi(S, M_i)$
  $Y \leftarrow Y \oplus (\Delta \parallel 0^{n/2})$
  $S \leftarrow \mathrm{BC}(K, Y)$
if $|M| \bmod n = 0$
  $\Delta \leftarrow 3\Delta$
if $|M| \bmod n \neq 0$
  $\Delta \leftarrow 3^2\Delta$
$(Y, C_\mu) \leftarrow \xi(S, M_\mu)$
$Y \leftarrow Y \oplus (\Delta \parallel 0^{n/2})$
$S \leftarrow \mathrm{BC}(K, Y)$
$T \leftarrow \lceil S \rceil_\tau$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
return (C, T)

$\underline{\textsc{Enc}_{\mathcal{C}}(K, N, A)}$

$Y \leftarrow N$
$S \leftarrow \mathrm{BC}(K, Y)$
$\Delta \leftarrow \lceil S \rceil_{n/2}$
$A_1, \ldots, A_\alpha \xleftarrow{n} \mathtt{pad}_{10^*}(A, n)$
for $i = 1, \ldots, \alpha - 1$
  $\Delta \leftarrow 2\Delta$
  $(Y, \cdot) \leftarrow \xi(S, A)$
  $Y \leftarrow Y \oplus (\Delta \parallel 0^{n/2})$
  $S \leftarrow \mathrm{BC}(K, Y)$
if $|A| \bmod n = 0$
  $\Delta \leftarrow 3\Delta$
  $j \leftarrow 1$
if $|A| \bmod n \neq 0$
  $\Delta \leftarrow 3^2\Delta$
  $j \leftarrow 2$
$(Y, \cdot) \leftarrow \xi(S, A_\alpha)$
$Y \leftarrow Y \oplus (\Delta \parallel 0^{n/2})$
$S \leftarrow \mathrm{BC}(K, Y)$
return (S, Δ)

$\underline{\xi(S, I)}$

$Y \leftarrow \widetilde{G}(S) \oplus I$
$O \leftarrow S \oplus I$
return (Y, O)

Figure 22: Pseudocode of Gift-Cofb [BCI+21] in term of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$.

While the advantage depends on μ, we stress that this vlaue is controlled by $\mathcal{A}$. In particular, choosing a short ciphertext of only one block leads to an attack with success probability of $\frac{1}{2}$.

For the proof of Theorem 13 we drop the XOR of the masking values. This avoids some very cumbersome and tedious bookkeeping. Subsequent to the proof, we discuss why the result also holds if the masking values are used.

Before giving the proof of Theorem 13, we give three lemmas that we will use to prove it. The first lemma, Lemma 7, shows that there is an algorithm that inverts the state-update-function of Gift-Cofb for random outputs with probability $\frac{1}{2}$. The second lemma, Lemma 8, shows that $\textsc{Enc}_{\mathcal{C}}$ can be inverted, i.e., given an arbitrary key K, a nonce N, and an output state S, there is an algorithm that outputs associated data A, such that $\textsc{Enc}_{\mathcal{C}}(K, N, A) = S$. The third lemma, Lemma 9, shows that there is an algorithm that inverts $\textsc{Enc}_{\mathcal{M}}$. The latter relies heavily on Lemma 7 which is why the success probability drops exponentially in the number of ciphertext blocks.

Lemma 7. *Let ξ be the state-update-function of* Gift-Cofb. *Let further $\mathcal{A}_\xi$ be the algorithm displayed in Fig. 23. Let C be an arbitrary bit string of length n. Then for*

$Y \leftarrow_{\$} \{0,1\}^n$, *it holds that*

$$\Pr[\xi(\mathcal{A}_\xi(Y,C)) = (Y,C)] = \frac{1}{2}\,.$$

Proof. We start by describing the algorithm $\mathcal{A}_\xi$ that is displayed in Fig. 23. We divide C and Y in two $\frac{n}{2}$-sized blocks, denoted by C_1, C_2 and Y_1, Y_2, respectively. Further, we denote the bits of $C_1 \oplus Y_1 \oplus C_2 \oplus Y_2$ by $z_1, \ldots, z_{\frac{n}{2}}$. Then, we randomly sample a bit s_1 and set $s_i \leftarrow s_{i-1} \oplus z_{i-1}$ for all $i = 2, \ldots, \frac{n}{2}$. By S_1 we denote the bit string resulting from concatenating the s_i's. Next, we define $S_2 \leftarrow C_1 \oplus Y_1 \oplus S_1$ and $S \leftarrow S_1 \parallel S_2$, which will be the first output of $\mathcal{A}_\xi$. Further, we set $M_1 \leftarrow Y_1 \oplus S_2$ and $M_2 \leftarrow Y_2 \oplus (S_1 \lll 1)$. The concatenated bit string $M \leftarrow M_1 \parallel M_2$ is the second output of $\mathcal{A}_\xi$, i.e., $\mathcal{A}_\xi(Y,C) = (S,M)$.

Next, we check that $\xi(\mathcal{A}_\xi(Y,C)) = (Y,C)$ holds with probability $\frac{1}{2}$. By definition of the state-update-function, the first component of $\xi(\mathcal{A}_\xi(Y,C)) = \xi(S,M)$ is given by

$$\begin{aligned}\widetilde{G}(S) \oplus M &= (S_2, S_1 \lll 1) \oplus M \\ &= (S_2 \oplus M_1, (S_1 \lll 1) \oplus M_2) \\ &= (Y_1, Y_2) = Y\,.\end{aligned}$$

The second component of $\xi(\mathcal{A}_\xi(Y,C)) = \xi(S,M)$ computes as

$$\begin{aligned}M \oplus S &= (M_1 \oplus S_1, M_2 \oplus S_2) \\ &= (Y_1 \oplus S_2 \oplus S_1, Y_2 \oplus (S_1 \lll 1) \oplus S_2)\,.\end{aligned}$$

Note that this expression is equal to C if and only if $S_1 \oplus (S_1 \lll 1) = C_1 \oplus Y_1 \oplus C_2 \oplus Y_2$, which results from solving $C_1 = Y_1 \oplus S_2 \oplus S_1$ for S_2 and plugging the result into $C_2 = Y_2 \oplus (S_1 \lll 1) \oplus S_2$. Breaking down this equation to the bit-level, yields

$$(s_1, s_2, \ldots, s_{\frac{n}{2}}) \oplus (s_2, s_3, \ldots, s_{\frac{n}{2}}, s_1) = (z_1, \ldots, z_{\frac{n}{2}})\,.$$

This gives the following equations

$$\begin{aligned}s_1 \oplus s_2 &= z_1 \\ s_2 \oplus s_3 &= z_2 \\ &\vdots \\ s_{\frac{n}{2}-1} \oplus s_{\frac{n}{2}} &= z_{\frac{n}{2}-1} \\ s_{\frac{n}{2}} \oplus s_1 &= z_{\frac{n}{2}}\end{aligned}$$

of which the first $\frac{n}{2} - 1$ equations hold by construction of the algorithm $\mathcal{A}_\xi$. Finally, we need to determine in which cases the last equation holds. For this, we replace $s_{\frac{n}{2}}$ by $s_{\frac{n}{2}-1} \oplus z_{\frac{n}{2}-1}$ which we get from the second to last equation. Then, in turn, we replace $s_{\frac{n}{2}-1}$ with $s_{\frac{n}{2}-2} \oplus z_{\frac{n}{2}-2}$ and continue this process until we get an equation depending only on the z_i's and s_1, namely

$$z_{\frac{n}{2}-1} \oplus \ldots z_2 \oplus z_1 \oplus s_1 \oplus s_1 = z_{\frac{n}{2}}\,,$$

which is equivalent to $z_{\frac{n}{2}} \oplus z_{\frac{n}{2}-1} \oplus \ldots z_2 \oplus z_1 = 0$. Thus we get $\xi(\mathcal{A}_\xi(Y,C)) = (\cdot, C)$ if and only if $C_1 \oplus Y_1 \oplus C_2 \oplus Y_2 = z_1 \parallel \ldots \parallel z_{\frac{n}{2}}$ contains an even number of 1s. Since Y is chosen uniformly at random, this is the case with probability $\frac{1}{2}$. This finishes the proof. □

Lemma 8. *Consider* GIFT-COFB *as described in Fig. 22. There exists an adversary $\mathcal{A}_\mathcal{C}$, making q queries to the ideal cipher* E *such that for any $(K,N,S) \in \mathcal{K} \times \mathcal{N} \times \{0,1\}^n$, it holds that*

$$\Pr[\text{ENC}_\mathcal{C}(K,N,A) = S \mid A \leftarrow \mathcal{A}_\mathcal{C}(K,N,S)] = 1\,.$$

The number of ideal cipher queries by $\mathcal{A}_\mathcal{C}$ is $q = \alpha + 1$ for α being the number of associated data blocks that $\mathcal{A}_\mathcal{C}$ outputs.

Proof. We construct the following adversary $\mathcal{A}_{\mathcal{C}}$ against GIFT-COFB as shown in Fig. 23. Its input is (K, N, S). First, $\mathcal{A}_{\mathcal{C}}$ randomly picks associated data blocks $A_2, \ldots, A_\alpha$, i.e., all except the first one.[24] Next, $\mathcal{A}_{\mathcal{C}}$ computes both S_* and Y_* (cf. Fig. 21): For the former, $\mathcal{A}_{\mathcal{C}}$ computes $S_* \leftarrow \mathsf{E}(K, N)$ and for the latter, $\mathcal{A}_{\mathcal{C}}$ consecutively inverts the ideal cipher E (starting from the input S) and the state-update-function ξ (by XORing an associated data block and inverting $\widetilde{G}$—note that we merely need to invert $\widetilde{G}$ as the output of ξ is discarded). Finally, $\mathcal{A}_{\mathcal{C}}$ computes $A_1 \leftarrow S_* \oplus Y_*$ and outputs $A \leftarrow \mathtt{pad}_{10^*}^{-1}(A_1 \parallel \ldots \parallel A_\alpha)$. By construction, it holds that

$$\textsc{Enc}_{\mathcal{C}}(K, N, A) = S\,.$$

The number of ideal cipher queries by $\mathcal{A}_{\mathcal{C}}$ is $\alpha + 1$. □

Lemma 9. *Consider* GIFT-COFB *described in Fig. 22. There exists an adversary $\mathcal{A}_{\mathcal{M}}$, making q queries to the ideal cipher E such that*

$$\Pr[\textsc{Enc}_{\mathcal{M}}(K, N, S, M) = (C, T) \mid (S, M) \leftarrow \mathcal{A}_{\mathcal{M}}(K, N, (C, T))] = \frac{1}{2^\mu}\,,$$

holds for any $(K, N, (C, T)) \in \mathcal{K} \times \mathcal{N} \times \mathcal{C}$. The number of ideal cipher queries by $\mathcal{A}_{\mathcal{M}}$ is $q = \mu$ for μ being the number of ciphertext blocks that $\mathcal{A}_{\mathcal{M}}$ receives as input.

Proof. We construct an adversary $\mathcal{A}_{\mathcal{M}}$ against GIFT-COFB as shown in Fig. 23. It gets $(K, N, (C, T))$ as input. For ease of exposition, we assume that the length of C is a multiple of the block size n, hence $C_1, \ldots, C_\mu \xleftarrow{n} C$ yields μ full blocks of length n. First, Adversary $\mathcal{A}_{\mathcal{M}}$ sets $S \leftarrow T$. Next, $\mathcal{A}_{\mathcal{M}}$ consecutively inverts the ideal cipher $Y \leftarrow \mathsf{E}^{-1}(K, S)$ and computes $(S, M_i) \leftarrow \mathcal{A}_\xi(Y, C_i)$. Finally, it sets $M \leftarrow M_1 \parallel \ldots \parallel M_\mu$ and outputs (S, M). Provided that $\mathcal{A}_\xi$ correctly inverts ξ, we obtain

$$\textsc{Enc}_{\mathcal{M}}(K, N, S, M) = (C, T)\,.$$

By Lemma 7—using Y is uniformly random due to E being an ideal cipher—every inversion of ξ succeeds with probability $\frac{1}{2}$. Since $\mathcal{A}_\xi$ is run μ times by $\mathcal{A}_{\mathcal{M}}$, we get

$$\Pr[\textsc{Enc}_{\mathcal{M}}(K, N, S, M) = (C, T) \mid (S, M) \leftarrow \mathcal{A}_{\mathcal{M}}(K, N, (C, T))] = \frac{1}{2^\mu}\,.$$

The number of queries to the ideal cipher E by $\mathcal{A}_{\mathcal{M}}$ is μ. □

We are now ready to give the proof of Theorem 13.

Proof (of Theorem 13). We construct $\mathcal{A}$ against GIFT-COFB as displayed in Fig. 23. It starts by sampling a context (K, N, A) together with a message M at random and computes $(C, T) \leftarrow$ GIFT-COFB.ENC(K, N, A, M). Next, it samples a fresh key-nonce pair $(\overline{K}, \overline{N})$, computes $(\overline{S}, \overline{M}) \leftarrow \mathcal{A}_{\mathcal{M}}(\overline{K}, \overline{N}, (C, T))$, and $\overline{A} \leftarrow \mathcal{A}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{S})$. Finally, $\mathcal{A}$ outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$.

By Lemma 8 and Lemma 9, we have the following equalities with probability $\frac{1}{2^\mu}$:

$$\begin{aligned}
\textsc{Gift-Cofb.Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M}) &= \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, \textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A}), \overline{M}) \\
&= \textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, \overline{S}, \overline{M}) && \text{(Lemma 8)} \\
&= (C, T) && \text{(Lemma 9)} \\
&= \textsc{Gift-Cofb.Enc}(K, N, A, M)\,.
\end{aligned}$$

[24] We assume that $\mathcal{A}_{\mathcal{C}}$ picks the blocks such that they exhibit a valid padding according to GIFT-COFB.

GIFT-COFB adversary $\mathcal{A}()$

$(K, N, A, M) \leftarrow\$ \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$
$(C, T) \leftarrow \textsc{Enc}(K, N, A, M)$
$(\overline{K}, \overline{N}) \leftarrow\$ \mathcal{K} \times \mathcal{N}$
$(\overline{S}, \overline{M}) \leftarrow \mathcal{A}_{\mathcal{M}}(\overline{K}, \overline{N}, (C, T))$
$\overline{A} \leftarrow \mathcal{A}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{S})$
return $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$

$\textsc{Enc}_{\mathcal{C}}$ adversary $\mathcal{A}_{\mathcal{C}}(K, N, S)$

$A_2, \ldots, A_\alpha \leftarrow\$ \{0,1\}^n$
$S_* \leftarrow \mathsf{E}(K, N)$
$Y \leftarrow \mathsf{E}^{-1}(K, S)$
for $i = \alpha, \ldots, 2$
  $S \leftarrow \widetilde{G}^{-1}(Y \oplus A_i)$
  $Y \leftarrow \mathsf{E}^{-1}(K, S)$
$Y_* \leftarrow Y$
$A_1 \leftarrow \widetilde{G}(S_*) \oplus Y_*$
$A \leftarrow A_1 \parallel \ldots \parallel A_\alpha$
return A

$\textsc{Enc}_{\mathcal{M}}$ adversary $\mathcal{A}_{\mathcal{M}}(K, N, (C, T))$

$S \leftarrow T$
for $i = \mu, \ldots, 1$
  $Y \leftarrow \mathsf{E}^{-1}(K, S)$
  $(S, M_i) \leftarrow \mathcal{A}_{\xi}(Y, C_i)$
$M \leftarrow M_1 \parallel \ldots \parallel M_\mu$
return (S, M)

$\mathcal{A}_{\xi}(Y, C)$

$C_1, C_2 \xleftarrow{\frac{n}{2}} C$
$Y_1, Y_2 \xleftarrow{\frac{n}{2}} Y$
$z_1, \ldots, z_{\frac{n}{2}} \xleftarrow{1} C_1 \oplus Y_1 \oplus C_2 \oplus Y_2$
$s_1 \leftarrow\$ \{0,1\}$
for $i = 2, \ldots, \frac{n}{2}$ **do**
  $s_i \leftarrow s_{i-1} \oplus z_{i-1}$
$S_1 \leftarrow s_1 \parallel \ldots \parallel s_{\frac{n}{2}}$
$S_2 \leftarrow C_1 \oplus Y_1 \oplus S_1$
$S \leftarrow S_1 \parallel S_2$
$M_1 \leftarrow Y_1 \oplus S_2$
$M_2 \leftarrow Y_2 \oplus (S_1 \lll 1)$
$M \leftarrow M_1 \parallel M_2$
return (S, M)

Figure 23: GIFT-COFB adversary $\mathcal{A}$ from Theorem 13 and the state-update-function adversary $\mathcal{A}_{\xi}$ from Lemma 7.

This yields

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Gift-Cofb}}(\mathcal{A}) = \frac{1}{2^\mu}\,.$$

Adversary $\mathcal{A}$ makes $q = 2\mu + \alpha + \overline{\alpha} + 2$ queries to E: $\mu + \alpha + 1$ for computing the first tuple, μ for inverting $\textsc{Enc}_{\mathcal{M}}$, and $\overline{\alpha} + 1$ for inverting $\textsc{Enc}_{\mathcal{C}}$. □

When considering the masking values, inverting $\textsc{Enc}_{\mathcal{M}}$ might seem to be a problem, as the adversary needs to invert the function using the correct masking values. We observe that the masking values depend on the key, the nonce, and the length of associated data and message/ciphertext. In particular, they are independent of the exact values of A, M, and C. Thus, when inverting $\textsc{Enc}_{\mathcal{M}}$, the adversary merely has to choose how long the associated data will be, as this allows to use the correct masking values.

The attack easily extends to $\mathrm{CMT}_{\mathsf{K}}$, $\mathrm{CMT}_{\mathsf{N}}$, $\mathrm{CMT}_{\mathsf{A}}$, and $\mathrm{CMT}^{\star}_{\mathsf{A}}$ attacks, following the argument we gave for ROMULUS.

B.3 PHOTON-BEETLE

The authenticated encryption scheme PHOTON-BEETLE [BCD+21] is a (duplex) sponge-based AE scheme. It uses the PHOTON permutation [GPP11] as the underlying permutation and the BEETLE mode of operation [CDNY18]. In contrast to the plain duplex, the

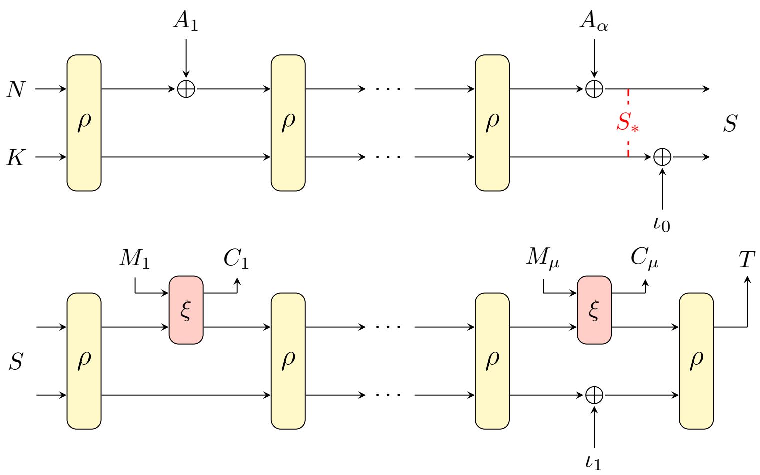


Figure 24: Illustration of Photon-Beetle in terms of $\text{Enc}_{\mathcal{C}}$ (top) and $\text{Enc}_{\mathcal{M}}$ (bottom). The state S_*, marked in red, is used in our CMT attack.

Beetle mode uses a state-update-function to determine the next input to the underlying permutation of the sponge.

B.3.1 Description of Photon-Beetle

Photon-Beetle is described in Fig. 25 and illustrated in Fig. 24. It is a CpP scheme, i.e., it processes the context (K, N, A) via the function $\text{Enc}_{\mathcal{C}}$ and subsequently processes the message together with the output of $\text{Enc}_{\mathcal{C}}$—an important property is that no part of the context is input to $\text{Enc}_{\mathcal{M}}$. In $\text{Enc}_{\mathcal{M}}$, the permutation and the state-update-function are applied in an alternating fashion. We omit the description of the latter, as our CMT attack is independent of it.

B.3.2 Committing Attack Against Photon-Beetle

In this section, we show that Photon-Beetle does not achieve committing security. The CMT attack is given in Theorem 14 below.

Theorem 14. *Consider* Photon-Beetle *which is illustrated and described in Fig. 24 and Fig. 25, respectively. Let ρ be modeled as an ideal permutation. Then there exists an adversary $\mathcal{A}$, making q queries to ρ, such that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\text{Photon-Beetle}}(\mathcal{A}) = 1\,,$$

where $q = \alpha + \overline{\alpha}$. Here, α is the number of blocks for the associated data of the first tuple, and $\overline{\alpha}$ is the number of blocks for the second tuple that $\mathcal{A}$ outputs.

Proof. We construct the following CMT adversary $\mathcal{A}$ against Photon-Beetle as shown in Fig. 26. It chooses (K, N, A) uniformly at random from the respective sets and computes $S \leftarrow \text{Enc}_{\mathcal{C}}(K, N, A)$. Let S_* denote the state before the domain separation is applied (see Fig. 24). Adversary $\mathcal{A}$ then chooses different associated data $\overline{A}$ and inverts $\text{Enc}_{\mathcal{C}}$ starting from S_* up to the initial state. This initial state is then used as the concatenation of nonce $\overline{N}$ and key $\overline{K}$.[25] As the final step, $\mathcal{A}$ picks a message M at random and outputs

[25]Note that these are likely to be different than K and N but not guaranteed to be.

$\textsc{Photon-Beetle.Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(S, M)$
return (C, T)

$\textsc{Enc}_{\mathcal{M}}(S, M)$

$M_1, \ldots, M_\mu \xleftarrow{r} \mathtt{pad}_{10^*}(M, r)$
for $i = 1, \ldots, \mu$
  $S \leftarrow \rho(S)$
  $(\lceil S \rceil_r, C_i) \leftarrow \xi(\lceil S \rceil_r, M_i)$
$S \leftarrow S \oplus (0^r \parallel \iota_1)$
$S \leftarrow \rho(S)$
$T \leftarrow \lceil S \rceil_\tau$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
return (C, T)

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$S \leftarrow N \parallel K$
$A_1, \ldots, A_\alpha \xleftarrow{r} \mathtt{pad}_{10^*}(A, r)$
for $i = 1, \ldots, \alpha$
  $S \leftarrow \rho(S)$
  $S \leftarrow S \oplus (A_i \parallel 0^c)$
$S \leftarrow S \oplus (0^r \parallel \iota_0)$
return S

$\xi(S, I)$

$O \leftarrow \mathsf{Shuffle}(S) \oplus I$
$Y \leftarrow S \oplus I$
return (Y, O)

Figure 25: Pseudocode of Photon-Beetle [BCD+21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$. Here, $\mathsf{Shuffle}(S) = S_2 \parallel (S_1 \ggg 1)$ for $S_1, S_2 \xleftarrow{\frac{r}{2}} S$.

$((K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, M))$. By construction, we have $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$ and it holds that

$$\begin{aligned}
\textsc{Photon-Beetle.Enc}(K, N, A, M) &= \textsc{Enc}_{\mathcal{M}}(\textsc{Enc}_{\mathcal{C}}(K, N, A), M) \\
&= \textsc{Enc}_{\mathcal{M}}(S_*, M) \\
&= \textsc{Enc}_{\mathcal{M}}(\textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A}), M) \\
&= \textsc{Photon-Beetle.Enc}(\overline{K}, \overline{N}, \overline{A}, M)\,,
\end{aligned}$$

thus $\mathcal{A}$ wins the game CMT.

$\mathcal{A}$ makes α queries to compute S_* and additional $\overline{\alpha}$ queries to compute $(\overline{K}, \overline{N})$, resulting in $q = \alpha + \overline{\alpha}$ queries in total. □

The attack is by construction also a valid $\mathrm{CMT}_{\mathsf{A}}$ attack as the associated data are chosen to be different. While the adversary does not choose the key $\overline{K}$ and nonce $\overline{N}$ for the second tuple, it is easy to see that the attack extends to $\mathrm{CMT}_{\mathsf{K}}$ and $\mathrm{CMT}_{\mathsf{N}}$—if $(\overline{K}, \overline{N}) = (K, N)$, the adversary simply chooses different associated data $\overline{A}$ and repeats the attack until the keys and nonces differ.

B.4 Xoodyak

The authenticated encryption scheme Xoodyak [DHP+21] is an AE scheme based on a full-state keyed duplex. Xoodyak uses the Xoodoo permutation [DHAK18] as the underlying permutation and the Cyclist mode of operation. The latter was introduced as part of the Xoodyak specification and is an adaption of the Keyak mode [BDP+16] to the lightweight setting.

B.4.1 Description of Xoodyak

The pseudocode of Xoodyak is given in Fig. 28 and further illustration is provided in Fig. 27. Xoodyak uses a state of size 384 and is a CpP scheme, i.e., first $S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$ is computed, followed by the computation of ciphertext and tag as $(C, T) \leftarrow$

PHOTON-BEETLE adversary $\mathcal{A}$	$\mathcal{B}(S_*, \overline{A})$
$(K, N) \leftarrow_\$ \mathcal{K} \times \mathcal{N}$	$\overline{A}_1, \ldots, \overline{A}_{\overline{\alpha}} \xleftarrow{r} \mathtt{pad}_{10^*}(\overline{A}, r)$
$S \leftarrow N \parallel K$	**for** $i = \overline{\alpha}, \ldots, 1$ **do**
$A \leftarrow_\$ \mathcal{A}$	$\quad S_* \leftarrow S_* \oplus (\overline{A}_i \parallel 0^c)$
$A_1, \ldots, A_\alpha \xleftarrow{r} \mathtt{pad}_{10^*}(A, r)$	$\quad S_* \leftarrow \rho^{-1}(S_*)$
for $i = 1, \ldots, \alpha$ **do**	$\overline{N} \parallel \overline{K} \leftarrow S_*$
$\quad S \leftarrow \rho(S)$	**return** $(\overline{K}, \overline{N})$
$\quad S \leftarrow S \oplus (A_i \parallel 0^c)$	
$S_* \leftarrow S$	
$\overline{A} \leftarrow_\$ \mathcal{A} \backslash \{A\}$	
$(\overline{K}, \overline{N}) \leftarrow \mathcal{B}(S_*, \overline{A})$	
$M \leftarrow_\$ \mathcal{M}$	
return $((K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, M))$	

Figure 26: PHOTON-BEETLE adversary $\mathcal{A}$ from Theorem 14.

$\textsc{Enc}_\mathcal{M}(S, M)$. In $\textsc{Enc}_\mathcal{C}$, the inputs are XORed onto the full-state (note that the last 32 bits are reserved for padding). In contrast to this, $\textsc{Enc}_\mathcal{M}$ uses a rate of 192 bits for the computation of ciphertext and tag.

XOODYAK exhibits a form of padding, that is used by none of the other NIST candidates and hence will be described shortly in the following: For a bit string X of length at most 352 and $p \in \{0,1\}^8$ define

$$\mathtt{pad}_\mathcal{C}(X, p) = X \parallel 00000001 \parallel 0^{368-|X|} \parallel p\,,$$

which is used for padding the context blocks. Further, for $M \in \{0,1\}^{\leq 192}$, we define $\mathtt{pad}_\mathcal{M}(M) = M \parallel 00000001$, which will be used to pad the message blocks.

B.4.2 Committing Attack Against XOODYAK

We show that XOODYAK does not achieve committing security. The attack is stated in the following theorem.

Theorem 15. *Consider* XOODYAK *which is illustrated and described in Fig. 27 and Fig. 28, respectively. Let ρ be modeled as an ideal permutation. Then there exists an adversary $\mathcal{A}$ that makes $q = 2^{17} + 1$ queries to ρ and fulfills*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Xoodyak}}(\mathcal{A}) \geq \frac{1}{2}\,.$$

Proof. We construct a CMT adversary $\mathcal{A}$ against XOODYAK. It uses a birthday attack to find a collision in the last 32 bits of the sponge state after the first application of ρ. For this, $q = 2^{17} + 1$ different key-nonce pairs are sampled randomly. For $i \in [q]$, we denote them by K_i and N_i and write $\boldsymbol{K}_i \parallel \boldsymbol{N}_i = \mathtt{pad}_\mathcal{C}((K_i \parallel N_i \parallel \mathtt{enc}_8(N_i)), 00000010)$ for their padded concatenation. Further, we consider the following random function

$$f : \{0,1\}^{384} \to \{0,1\}^{32}, \quad f(X) = \lfloor \rho(X) \rfloor_{32}\ ,$$

and compute $f(\boldsymbol{K}_i \parallel \boldsymbol{N}_i)$ for each $i \in [q]$. By the birthday attack [BS23, Section 8.3][26], a collision of f is found with probability at least $\frac{1}{2}$. Assume that such a collision has been found and write (K, N) and $(\overline{K}, \overline{N})$ for the key-nonce pairs that lead to it, i.e., have

[26] Note that the prerequisite, regarding the size of domain and codomain of f, is fulfilled as $2^{384} \geq 100 \cdot 2^{32}$.

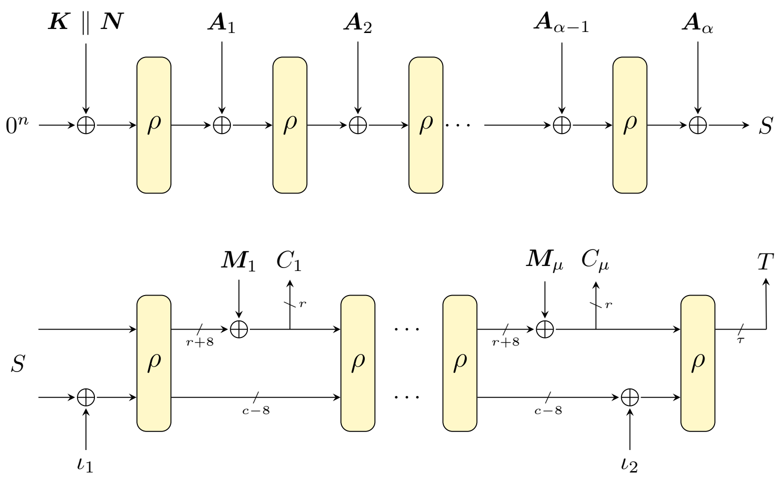


Figure 27: Illustration of XOODYAK in terms of $\text{ENC}_\mathcal{C}$ (top) and $\text{ENC}_\mathcal{M}$ (bottom). Here, $\boldsymbol{K} \parallel \boldsymbol{N} = \texttt{pad}_\mathcal{C}((K \parallel N \parallel \texttt{enc}_8(|N|)), 00000010)$, $\boldsymbol{M}_i = \texttt{pad}_\mathcal{M}(M_i)$, $\boldsymbol{A}_1 = \texttt{pad}_\mathcal{C}(A_1, 00000011)$, and for $i = 2, \ldots, \alpha$, $\boldsymbol{A}_i = \texttt{pad}_\mathcal{C}(A_i, 0^8)$. In $\text{ENC}_\mathcal{M}$, the increased rate $(r + 8)$ is required for the padding $\texttt{pad}_\mathcal{M}$ which appends 8 bits to the message blocks.

$\lfloor \rho(\boldsymbol{K} \parallel \boldsymbol{N}) \rfloor_{32} = \lfloor \rho(\overline{\boldsymbol{K}} \parallel \overline{\boldsymbol{N}}) \rfloor_{32}$, where $\boldsymbol{K} \parallel \boldsymbol{N}$ and $\overline{\boldsymbol{K}} \parallel \overline{\boldsymbol{N}}$ denote the corresponding padded values. Next, adversary $\mathcal{A}$ picks $A \in \{0,1\}^{352}$ at random and computes $\overline{A} \leftarrow \lceil \rho(\overline{\boldsymbol{K}} \parallel \overline{\boldsymbol{N}}) \rceil_{352} \oplus \lceil \rho(\boldsymbol{K} \parallel \boldsymbol{N}) \rceil_{352} \oplus A$. Together with the collision on the last 32 bits, $\mathcal{A}$ has produced a collision on the whole state. Then, the adversary wins the game CMT against XOODYAK by outputting (K, N, A, M) and $(\overline{K}, \overline{N}, \overline{A}, M)$ for M some randomly sampled message. This is the case, as $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$ and the states after the associated data is absorbed agree, from which point on only the same input (namely M) is processed for both tuples. Hence, we obtain $\text{XOODYAK.ENC}(K, N, A, M) = \text{XOODYAK.ENC}(\overline{K}, \overline{N}, \overline{A}, M)$ and in total have shown that $\mathcal{A}$ wins with probability at least $\frac{1}{2}$ for $q = 2^{17} + 1$ queries. $\square$

Note that, using the same strategy as presented above, we obtain an attacker that wins with probability 1 after making $2^{32} + 1$ queries. Further, the above attack is by construction also a valid CMT_K and CMT_N attack as keys and nonces, respectively, are chosen to be different. Moreover, it can be shown to be a CMT_A attack: The same associated data blocks are only chosen if the states after the first application of ρ already coincide in their rate part. Since they agree in the last 32 bits by construction, this would constitute a full-state collision of ρ, which is impossible for a permutation.

B.5 ISAP

The authenticated encryption scheme ISAP [DEM+21, DEM+17, DEM+20] is a sponge-based scheme designed to withstand side-channel leakage. It features a re-keying approach that guarantees that for each input a different session key is used. The re-keying function uses a small rate to prevent adversaries from obtaining too much leakage.

B.5.1 Description of ISAP

The pseudocode of ISAP is given in Fig. 31 and further illustration is provided in Fig. 30. ISAP follows the EtM-approach, i.e., first the message is encrypted via $\text{ENC}_\mathcal{M}$ resulting in a ciphertext C and afterwards, the tag T is computed using $\text{ENC}_\mathcal{T}$, which processes the context and the ciphertext. Both $\text{ENC}_\mathcal{M}$ and $\text{ENC}_\mathcal{T}$ internally uses the re-keying function

$\textsc{Xoodyak}.\textsc{Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(S, M)$
return (C, T)

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$S \leftarrow 0^n$
$X \leftarrow K \parallel N \parallel \mathtt{enc}_8(|N|)$
$Y \leftarrow S \oplus \mathtt{pad}_{\mathcal{C}}(X, 0^6 \parallel 10)$
$S \leftarrow \rho(Y)$
$A_1, \ldots, A_\alpha \xleftarrow{352} A$
$Y \leftarrow S \oplus \mathtt{pad}_{\mathcal{C}}(A_1, 0^6 \parallel 11)$
for $i = 2, \ldots, \alpha$
  $S \leftarrow \rho(Y)$
  $Y \leftarrow S \oplus \mathtt{pad}_{\mathcal{C}}(A_i, 0^8)$
$S \leftarrow Y$
return S

$\textsc{Enc}_{\mathcal{M}}(S, M)$

$Y \leftarrow S \oplus (0^{r+8} \parallel \iota_1)$
$M_1, \ldots, M_\mu \xleftarrow{r} M$
for $i = 1, \ldots, \mu$
  $\boldsymbol{M}_i \leftarrow \mathtt{pad}_{\mathcal{M}}(M_i)$
  $S \leftarrow \rho(Y)$
  $Y \leftarrow S \oplus (\boldsymbol{M}_i \parallel 0^{c-8})$
  $C_i \leftarrow \lceil Y \rceil_r$
$Y \leftarrow Y \oplus (0^{r+8} \parallel \iota_2)$
$S \leftarrow \rho(Y)$
$T \leftarrow \lceil S \rceil_\tau$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
return (C, T)

Figure 28: Pseudocode of Xoodyak [DHP+21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$.

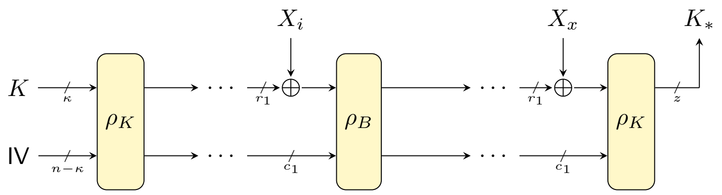


Figure 29: Illustration of Isap.Rk.

Isap.Rk to derive the session key. For the security analysis [DEM+21], $\textsc{Enc}_{\mathcal{M}}$ is viewed as a keyed duplex construction while $\textsc{Enc}_{\mathcal{T}}$ is viewed as a suffix keyed sponge (SuKS) [DM19].

In Isap, the underlying permutation is applied several times between two absorptions, the precise number depending on the position in the Isap sponge. For Isap.Rk, for instance, more rounds are applied when the key is processed and when the tag is generated, while fewer rounds are used in between. In summary, Isap uses four permutations ρ_K, ρ_H, ρ_B, and ρ_E, each based on the same permutation but applied a different number of times.

B.5.2 Committing Security of Isap

We show that Isap achieves CMT security. In the security proof for Isap [DEM+20], the two permutations used in Isap.Rk (namely ρ_K and ρ_B) are modeled as one permutation. We adopt this for our proof and denote the permutation used in Isap.Rk by ρ_1 and the one used in $\textsc{Enc}_{\mathcal{T}}$ by ρ_2.[27] In conformity with this, the rate in Isap.Rk is denoted by r_1 and the rate in $\textsc{Enc}_{\mathcal{T}}$ by r_2. Furthermore, we consider a slightly different domain separation in $\textsc{Enc}_{\mathcal{T}}$, by XORing $1 \parallel 0^{c_2-1}$ instead of $0^{c_2-1} \parallel 1$. By the same reasoning as given for Ascon, this is a purely cosmetic change which allows us to view $\textsc{Enc}_{\mathcal{T}}$ as a

[27] The proof is independent of $\textsc{Enc}_{\mathcal{M}}$ which is why we do not need a third permutation.

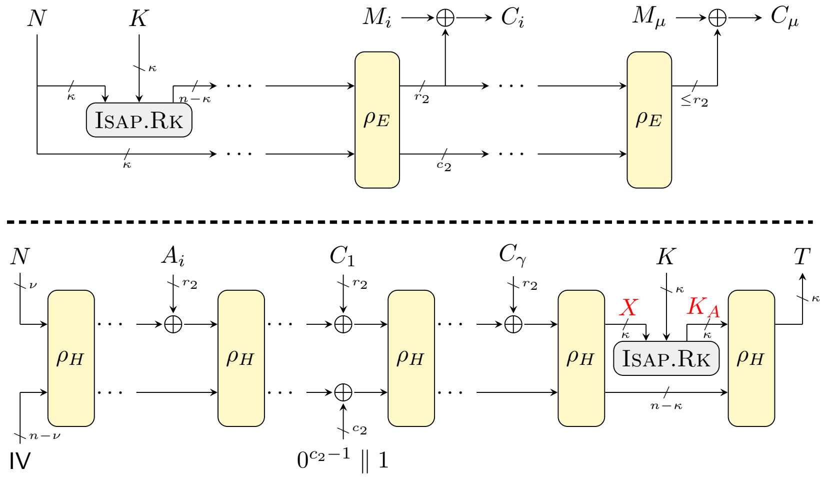


Figure 30: Illustration of ISAP in terms of $\text{ENC}_{\mathcal{M}}$ (top) and $\text{ENC}_{\mathcal{T}}$ (bottom), which both rely on the re-keying function ISAP.RK. The values X and K_A, marked in red, are used in our CMT proof.

sponge with increased rate of $r + 1$. In particular, it neither affects the security of ISAP nor defeats the purpose of the domain separation.

Lastly, note that the scheme comes in two variants, using either ASCON-P or KECCAK-P as a permutation. The theorem given below holds for both instances, considering each time the main parameter sets as described in Table 3. Since for all parameter sets under consideration the key-length equals the tag-length ($\kappa = \tau = 128$), we exclusively use the variable κ in the following proof.

The proof for ISAP follows the same idea as the one for ASCON, again using the collision resistance of plain sponges. There are two main differences, though. First, ISAP feastures a larger capacity in the initial state which allows to directly apply Theorem 9.[28] Second, some extra care is necessary to handle the re-keying mechanism of ISAP, which essentially results in two applications of Theorem 9.

Theorem 16. *Consider* ISAP *which is illustrated and described in Fig. 30 and Fig. 31, respectively. Let ρ_1 and ρ_2 be modeled by ideal permutations ρ_1 and ρ_2, respectively. Then for any adversary $\mathcal{A}$ making q_1 and q_2 queries to ρ_1 and ρ_2, respectively, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\text{ISAP}}(\mathcal{A}) \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}} .$$

Proof. Let $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ be the output of a CMT adversary $\mathcal{A}$ against ISAP. Further note that IV denotes the initialization vector used in ISAP. We assume that $\mathcal{A}$ makes queries to ρ_1 and ρ_2 that correspond to its output, i.e., querying all states that occur during the evaluation of ISAP for the two output tuples of $\mathcal{A}$. This assumption is without loss of generality, as we can easily transform any adversary $\mathcal{A}$ into one that runs $\mathcal{A}$ to obtain $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ and—before outputting the same—makes all queries to ρ corresponding to the evaluation of $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$.

The authentication component $\text{ENC}_{\mathcal{T}}$ uses a session key denoted by K_A (resp. $\overline{K}_A$), which results from an application of ISAP.RK to the key K (resp. $\overline{K}$) and the intermediate state X (resp. $\overline{X}$) computed during $\text{ENC}_{\mathcal{T}}$. We consider the event E that $(N, A) = (\overline{N}, \overline{A})$

[28]In a similar vein, one can use the old indifferentiability bound [BDPV08] for ISAP, whereas ASCON requires the newer version [NO14].

$\textsc{Isap}.\textsc{Enc}(K, N, A, M)$

$C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, M)$
$T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$
$\textbf{return } (C, T)$

$\textsc{Enc}_{\mathcal{T}}(K, N, A, C)$

$A_1, \ldots, A_\alpha \xleftarrow{r_2} \mathtt{pad}_{10^*}(A, r_2)$
$C_1, \ldots, C_\gamma \xleftarrow{r_2} \mathtt{pad}_{10^*}(C, r_2)$
$Y \leftarrow N \parallel \mathsf{IV}$
$S \leftarrow \rho_H(Y)$
$\textbf{for } i = 1, \ldots, \alpha$
$\quad Y \leftarrow S \oplus (A_i \parallel 0^{c_2})$
$\quad S \leftarrow \rho_H(Y)$
$S \leftarrow S \oplus 0^{n-1} \parallel 1$
$\textbf{for } i = 1, \ldots, \gamma$
$\quad Y \leftarrow S \oplus (C_i \parallel 0^{c_2})$
$\quad S \leftarrow \rho_H(Y)$
$K_A \leftarrow \textsc{Isap}.\textsc{Rk}(K, \lceil S \rceil_\kappa)$
$S \leftarrow \rho_H(K_A, \lceil S \rceil_\kappa)$
$T \leftarrow \lceil S \rceil_\tau$
$\textbf{return } T$

$\textsc{Enc}_{\mathcal{M}}(K, N, M)$

$M_1, \ldots, M_\mu \xleftarrow{r_2} \mathtt{pad}_{0^*}(M, r_2)$
$K_E \leftarrow \textsc{Isap}.\textsc{Rk}(K, N)$
$S \leftarrow K_E \parallel N$
$\textbf{for } i = 1, \ldots, \mu$
$\quad S \leftarrow \rho_E(S)$
$\quad C_i \leftarrow \lceil S \rceil_{r_2} \oplus M_i$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
$\textbf{return } C$

$\textsc{Isap}.\textsc{Rk}(K, X)$

$X_1, \ldots, X_z \xleftarrow{r_1} X$
$Y \leftarrow K \parallel \mathsf{IV}$
$S \leftarrow \rho_K(Y)$
$\textbf{for } i = 1, \ldots, z-1$
$\quad Y \leftarrow S \oplus (X_i \parallel 0^{n-r_1})$
$\quad S \leftarrow \rho_B(Y)$
$Y \leftarrow S \oplus (X_z \parallel 0^{n-r_1})$
$S \leftarrow \rho_K(Y)$
$\textbf{return } \lceil S \rceil_\kappa$

Figure 31: Pseudocode of Isap [DEM+21] in terms of $\textsc{Enc}_{\mathcal{M}}$ and $\textsc{Enc}_{\mathcal{T}}$.

and $K_A = \overline{K}_A$. Using this, the CMT advantage can be divided up as follows

$$\begin{aligned}\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Isap}}(\mathcal{A}) &= \Pr[\mathsf{CMT}(\mathcal{A}) \to 1] \\ &= \Pr[\mathsf{E} \wedge \mathsf{CMT}(\mathcal{A}) \to 1] + \Pr[\neg\mathsf{E} \wedge \mathsf{CMT}(\mathcal{A}) \to 1]\,.\end{aligned}$$

We start by giving an upper bound for the second summand. For this, we construct a CR (see Definition 4) adversary $\mathcal{B}$ against a sponge-based hash function $\mathcal{H}_2$ obtained from the permutation ρ_2 with rate $\overline{r}_2 = \max\{\kappa, r_2 + 1\}$[29], capacity $\overline{c}_2 = n - \overline{r}_2$, and output length κ. Further $0^\kappa \parallel \mathsf{IV}$ is chosen as the initial state of $\mathcal{H}_2$. First, $\mathcal{B}$ runs $\mathcal{A}$, which outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. For every query that $\mathcal{A}$ makes to ρ_2, the adversary $\mathcal{B}$ makes the same query to its own permutation and sends the response back to $\mathcal{A}$. Further, Adversary $\mathcal{B}$ simulates ρ_1 for $\mathcal{A}$. Using this, $\mathcal{B}$ computes for both output tuples of $\mathcal{A}$, the state in $\textsc{Enc}_{\mathcal{T}}$ after the associated data and the ciphertext blocks are absorbed. We denote the states for the first tuple and second tuple by X and $\overline{X}$, respectively and the session keys for $\textsc{Enc}_{\mathcal{T}}$ by K_A and $\overline{K}_A$, respectively (cf. Fig. 32). The states obtained after XORing these together are denoted by $Z = X \oplus K_A$ and $\overline{Z} = \overline{X} \oplus \overline{K}_A$.[30]

Let $A_1, \ldots, A_\alpha \xleftarrow{r_2} \mathtt{pad}_{10^*}(A, r_2)$ and $\overline{A}_1, \ldots, \overline{A}_{\overline{\alpha}} \xleftarrow{r_2} \mathtt{pad}_{10^*}(\overline{A}, r_2)$ be the division of A and $\overline{A}$ into blocks of length r_2. Analogously, the ciphertext $C = \textsc{Enc}_{\mathcal{M}}(K, N, M) =$

[29] The definition of the rate over the maximum ensures that the argument works for both Isap variants (Isap-K and Isap-A). More precisely, it guarantees that all inputs can still be fully absorbed after the adjustment of the rate.

[30] Note that $\mathcal{B}$ can compute these values by looking up the queries and responses from $\mathcal{A}$'s queries—using the assumption that it makes permutation queries corresponding to its output. Thus, this step does not require any additional permutation queries by $\mathcal{B}$.

$\textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, \overline{M})$ is parsed as $C_1, \ldots, C_\gamma \xleftarrow{r_2} \mathtt{pad}_{10^*}(C, r_2)$. The adversary $\mathcal{B}$ then outputs

$$O = (N \parallel 0^*, A_1 \parallel 0^*, \ldots, A_\alpha \parallel 0^*, C_1 \parallel 1 \parallel 0^*, \ldots, C_\gamma \parallel 0^*, Z \parallel 0^*)$$
$$\overline{O} = (\overline{N} \parallel 0^*, \overline{A}_1 \parallel 0^*, \ldots, \overline{A}_{\overline{\alpha}} \parallel 0^*, C_1 \parallel 1 \parallel 0^*, \ldots, C_\gamma \parallel 0^*, \overline{Z} \parallel 0^*)\,,$$

where $\parallel 0^*$ denotes the padding with 0s up to length $\overline{r}_2$. A visualization for this is provided in Fig. 32. We show that if $\mathcal{A}$ wins the game CMT against $\textsc{Isap}$ and the event $\neg\mathsf{E}$ holds, then the constructed adversary $\mathcal{B}$ wins the game CR against $\mathcal{H}_2$. Note that $\mathcal{A}$ winning the game CMT implies that $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$ and $\textsc{Enc}_{\mathcal{T}}(K, N, A, C) = T = \textsc{Enc}_{\mathcal{T}}(\overline{K}, \overline{N}, \overline{A}, C)$. Hence, the output tuples of $\mathcal{B}$ are mapped to the same result under $\mathcal{H}_2$ (namely T) and it is only left to check that $O \neq \overline{O}$ to guarantee a collision. As event $\neg\mathsf{E}$ holds, we know that $(N, A) \neq (\overline{N}, \overline{A})$ or $K_A \neq \overline{K}_A$. In the case that $(N, A) \neq (\overline{N}, \overline{A})$, we have $O \neq \overline{O}$. Hence, we can assume from now on that $(N, A) = (\overline{N}, \overline{A})$. Next, consider the case that $K_A \neq \overline{K}_A$. As $(N, A) = (\overline{N}, \overline{A})$, we know that $X = \overline{X}$ and hence $Z = \lceil X \rceil_\kappa \oplus K_A \neq \lceil \overline{X} \rceil_\kappa \oplus \overline{K}_A = \overline{Z}$. Thus, $O \neq \overline{O}$ is also given in this case. So far, we have shown that

$$\Pr[\neg\mathsf{E} \wedge \mathsf{CMT}(\mathcal{A}) \to 1] \leq \Pr[\mathsf{CR}(\mathcal{B}) \to 1] \leq \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\overline{r}_2}}\,,$$

where the last inequality holds by Theorem 9, which bounds the probability of finding a collision in a general sponge-based hash function. Here, we exploit the fact that $\mathcal{B}$ makes the same number of queries to ρ_2 as $\mathcal{A}$.

Next, we give a bound for the first summand $\Pr[\mathsf{E} \wedge \mathsf{CMT}(\mathcal{A}) \to 1]$. For this, construct a CR adversary $\mathcal{C}$ against a sponge-based hash function $\mathcal{H}_1$ obtained from the permutation ρ_1 with rate $\overline{r}_1 = \kappa$, capacity $\overline{c}_1 = n - \overline{r}_2$, and output length κ. Further its initial state is given by $0^\kappa \parallel \mathsf{IV}$. The adversary $\mathcal{C}$ starts by running $\mathcal{A}$, which outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. For every query that $\mathcal{A}$ makes to ρ_1, the adversary $\mathcal{C}$ makes the same query to its own permutation and sends the response back to $\mathcal{A}$. Further, adversary $\mathcal{C}$ simulates ρ_2 for $\mathcal{A}$. Then, it computes X and $\overline{X}$ (analoguously to adversary $\mathcal{B}$), and we write $X_1, \ldots, X_\kappa \xleftarrow{1} X$ and $\overline{X}_1, \ldots, \overline{X}_\kappa \xleftarrow{1} \overline{X}$. Lastly, adversary $\mathcal{C}$ outputs $(K, X_1 \parallel 0^{\kappa-1}, \ldots, X_\kappa \parallel 0^{\kappa-1})$ and $(\overline{K}, \overline{X}_1 \parallel 0^{\kappa-1}, \ldots, \overline{X}_\kappa \parallel 0^{\kappa-1})$. Next, we show that if $\mathcal{A}$ wins the game CMT against $\textsc{Isap}$ and event E holds, then the constructed adversary $\mathcal{C}$ wins the game CR against $\mathcal{H}_1$. First observe that $\mathcal{A}$ winning the game CMT, implies that $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$, and $\textsc{Isap}.\textsc{Enc}(K, N, A, M) = \textsc{Isap}.\textsc{Enc}(\overline{K}, \overline{N}, \overline{A}, \overline{M})$. If at the same time event E holds, i.e., $(N, A) = (\overline{N}, \overline{A})$ and $K_A = \overline{K}_A$ hold, then $K \neq \overline{K}$, as otherwise $\mathcal{A}$ would not be a valid CMT adversary. Further note that $X = \overline{X}$ as $(N, A) = (\overline{N}, \overline{A})$. Hence, $\mathcal{C}$ wins the game CR, because the tuples it outputs are different, but their image under $\mathcal{H}_1$ agrees (as $K_A = \overline{K}_A$). This implies that

$$\Pr[\mathsf{E} \wedge \mathsf{CMT}(\mathcal{A}) \to 1] \leq \Pr[\mathsf{CR}(\mathcal{C}) \to 1] \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}}\,,$$

where the last inequality holds by Theorem 9. Using $\overline{r}_2 = \max\{\kappa, r_2+1\}$, we obtain in total

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Isap}}(\mathcal{A}) \leq \frac{q_1(q_1-1)}{2^\kappa} + \frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2-1)}{2^\kappa} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}}\,,$$

which finishes the proof. □

The dominant term in the bound from Theorem 16 is $\frac{q_1(q_1-1)}{2^\kappa} + \frac{q_2(q_2-1)}{2^\kappa}$, thus by increasing κ (i.e., the tag and key length), we can increase the committing security. Note however, that—for $\textsc{Isap}$-A—we can only increase κ up to 160 as for larger values the other term $\frac{q_1(q_1+1)}{2^{n-\kappa}} + \frac{q_2(q_2+1)}{2^{n-\max\{\kappa, r_2+1\}}}$ becomes the dominant term. This would result in about 80-bit committing security. A similar argument applies for $\textsc{Isap}$-K, which deploys $\textsc{Keccak}$-P as the underlying permutation. For this variant, we have $n = 400$ which allows to increase κ up to 200, allowing for about 100-bit committing security.

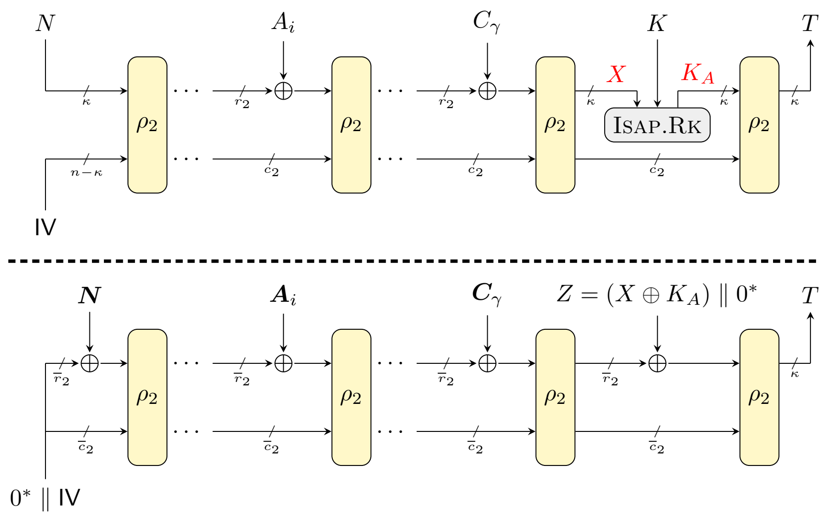


Figure 32: Illustration of a proof step for ISAP (Theorem 16).. Here, $\bar{r}_2 = \max\{\kappa, r_H + 1\}$ (i.e., 129 for ISAP-A and 145 for ISAP-K) and $\bar{c}_2 = n - \bar{r}_2$; further write $\boldsymbol{N} = N \parallel 0^*$, $\boldsymbol{A}_i = A_i \parallel 0^*$, $\boldsymbol{C}_1 = C_1 \parallel 1 \parallel 0^*$ and $\boldsymbol{C}_i = C_i \parallel 0^*$ for $i \in \{2, \ldots, \gamma\}$.

B.6 SCHWAEMM

SCHWAEMM [BBC+20, BBdS+21] is a sponge-based AE scheme. The permutation used to instantiate SCHWAEMM is SPARKLE which is inspired by the block-cipher SPARX [DPU+16]. The authentication mode is a variant of the BEETLE mode [CDNY18].

B.6.1 Description of SCHWAEMM

The pseudocode of SCHWAEMM is given in Fig. 34 and further illustration is provided in Fig. 33. SCHWAEMM follows the CpP-approach, i.e., first the context is processed resulting in $S \leftarrow \mathrm{ENC}_{\mathcal{C}}(K, N, A)$ and afterwards the ciphertext is computed as $(C, T) \leftarrow \mathrm{ENC}_{\mathcal{M}}(K, S, M)$. In SCHWAEMM the underlying permutation ρ is applied a varying number of times, depending on the position in the sponge (ρ^a and ρ^b for $a = 11$ and $b = 7$), similar to ISAP and ASCON.

Like some of the other schemes, SCHWAEMM features a state-update-function that is defined as follows:

$$\begin{aligned}\xi\colon \{0,1\}^r \times \{0,1\}^r &\to \{0,1\}^r \times \{0,1\}^r,\\ (S, I) &\mapsto (\xi_1(S, I), \xi_2(S, I)) = (\mathsf{FeistelSwap}(S) \oplus I, S \oplus I)\,.\end{aligned}$$

for $\mathsf{FeistelSwap} : \{0,1\}^r \to \{0,1\}^r$, $\mathsf{FeistelSwap}(S) = S_2 \parallel (S_2 \oplus S_1)$ with $S_1 = \lceil S \rceil_{\frac{r}{2}}$ and $S_2 = \lfloor S \rfloor_{\frac{r}{2}}$. Furthermore, SCHWAEMM deploys a so-called rate-whitening function, given by

$$\omega_{c,r}\colon \{0,1\}^c \to \{0,1\}^r, \quad \omega_{c,r}(I) = (I_1, I_2, I_1, I_2) \text{ for } I_1 = \lceil I \rceil_{\frac{c}{2}},\ I_2 = \lfloor I \rfloor_{\frac{c}{2}}\ .$$

In each round, it is applied between the state-update-function and the permutation. After the final permutation, the last κ bits are XORed with the key to yield the tag. As for ASCON, we refer to this as output-blinding.

B.6.2 Committing Security of SCHWAEMM

We show that SCHWAEMM achieves committing security. At the first glance, it looks like one can apply the same attack used against PHOTON-BEETLE: Invert $\mathrm{ENC}_{\mathcal{C}}$ for some S

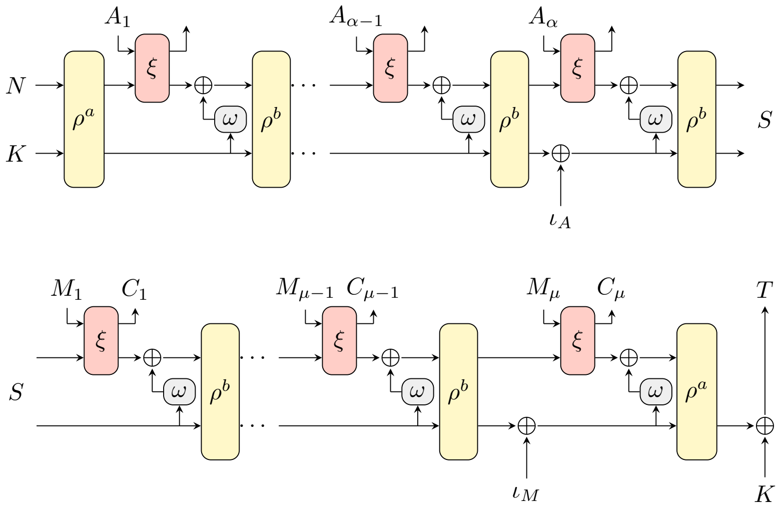


Figure 33: Illustration of SCHWAEMM in terms of $\text{ENC}_\mathcal{C}$ (top) and $\text{ENC}_\mathcal{M}$ (bottom).

and take the result as the concatenation of key and nonce. However, SCHWAEMM deploys output-blinding in $\text{ENC}_\mathcal{M}$ (the last XOR of the key in Fig. 33), that makes the attack unlikely to succeed. Output-blinding is a feature we have also encountered in ASCON, as an important feature for achieving committing security. Despite that, we cannot show committing security in the same way, as SCHWAEMM lacks the state-blinding, that ASCON has, and SCHWAEMM's initial state does not contain a fixed component. However, we noticed that introducing an IV to SCHWAEMM's initial state suffices to obtain a committing secure scheme—despite the weaker blinding mechanism. More precisely, we decrease the length of the nonce from 256 to 128 bits[31] and instead incorporate a fixed IV (of length 128 bits into the initial state. For the resulting modified version of SCHWAEMM, denoted by $\text{SCHWAEMM}_{\mathsf{IV}}$, we can show about 64-bit committing security. For the proof, we model the two permutations ρ^a and ρ^b by one ideal permutation ρ, as it was done for ISAP and ASCON. We further drop the domain separation in our proof for sake of simplicity. This part can easily be incorporated at the cost of reducing the committing security by the number of bits required for the domain separation.

Theorem 17. *Consider* SCHWAEMM *which is illustrated and described in Fig. 33 and Fig. 34, respectively. and* $\text{SCHWAEMM}_{\mathsf{IV}}$*, its modified version described above. Let ρ^a and ρ^b be modeled as a random permutation ρ. Then for any adversary $\mathcal{A}$ making $q \leq 2^{127}$ queries to ρ, it holds that*

$$\mathbf{Adv}^{\mathsf{CMT}}_{\text{SCHWAEMM}_{\mathsf{IV}}}(\mathcal{A}) \leq 1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right) + \epsilon\,,$$

for $\epsilon > \frac{(1-2^{-256})q^2 + (1+2^{-256})q}{2^{129}}$.

Proof. Firstly, we observe that finding different inputs to $\text{SCHWAEMM}_{\mathsf{IV}}.\text{ENC}$ that result in the same ciphertext is easy. However, breaking CMT security also includes finding colliding tags, which is what we focus on in the following. An adversary $\mathcal{A}$ that wins the game CMT against $\text{SCHWAEMM}_{\mathsf{IV}}$ its output denoted by $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$, in particular finds a tag collision, i.e., wins the game TagColl (see Fig. 15). Hence we can

[31] Note that the modified scheme is still in accordance to the NIST requirements [NIST15] that nonces are at least 96 bits long.

$\textsc{Schwaemm}.\textsc{Enc}(K, N, A, M)$

$S \leftarrow \textsc{Enc}_{\mathcal{C}}(K, N, A)$
$(C, T) \leftarrow \textsc{Enc}_{\mathcal{M}}(K, S, M)$
return (C, T)

$\textsc{Enc}_{\mathcal{C}}(K, N, A)$

$A_1, \ldots, A_\alpha \xleftarrow{r} \mathtt{pad}_{10^*}(A, r)$
$Y \leftarrow K \parallel N$
$S \leftarrow \rho^a(Y)$
for $i = 1, \ldots, \alpha - 1$
  $(X, \cdot) \leftarrow \xi(\lceil S \rceil_r, A_i)$
  $Y \leftarrow (X \oplus \omega(\lfloor S \rfloor_c)) \parallel \lfloor S \rfloor_c$
  $S \leftarrow \rho^b(Y)$
$(X, \cdot) \leftarrow \xi(\lceil S \rceil_r, A_\alpha)$
$Y \leftarrow X \parallel (\lfloor S \rfloor_c \oplus \iota_A)$
$Y \leftarrow (\lceil Y \rceil_r \oplus \omega(\lfloor Y \rfloor_c)) \parallel \lfloor Y \rfloor_c$
$S \leftarrow \rho^a(Y)$
return S

$\textsc{Enc}_{\mathcal{M}}(K, S, M)$

$M_1, \ldots, M_\mu \xleftarrow{r} \mathtt{pad}_{10^*}(M, r)$
for $i = 1, \ldots, \mu - 1$
  $(X, C_i) \leftarrow \xi(\lceil S \rceil_r, M_i)$
  $Y \leftarrow (X \oplus \omega(\lfloor S \rfloor_c)) \parallel \lfloor S \rfloor_c$
  $S \leftarrow \rho^b(Y)$
$(X, C_\mu) \leftarrow \xi(\lceil S \rceil_r, M_\mu)$
$Y \leftarrow X \parallel (\lfloor S \rfloor_c \oplus \iota_M)$
$Y \leftarrow (\lceil Y \rceil_r \oplus \omega(\lfloor Y \rfloor_c)) \parallel \lfloor Y \rfloor_c$
$C \leftarrow \lceil C_1 \parallel \ldots \parallel C_\mu \rceil_{|M|}$
$S \leftarrow \rho^a(Y)$
$T \leftarrow \lceil S \rceil_\tau \oplus K$
return (C, T)

$\xi(S, I)$

$Y \leftarrow \mathsf{FeistelSwap}(S) \oplus I$
$O \leftarrow S \oplus I$
return (Y, O)

Figure 34: Pseudocode of SCHWAEMM [BBdS+21] in terms of $\textsc{Enc}_{\mathcal{C}}$ and $\textsc{Enc}_{\mathcal{M}}$.

deduce

$$\mathbf{Adv}^{\mathsf{CMT}}_{\textsc{Schwaemm}_{\mathsf{IV}}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{TagColl}}_{\textsc{Schwaemm}_{\mathsf{IV}}}(\mathcal{A})\,.$$

As a next step, we pass over to a plain sponge construction. For this, we construct a $\mathsf{ShiftedColl}_{128}$ adversary $\mathcal{B}$ against a sponge-based hash function $\mathcal{H}$ obtained from the permutation ρ with rate 256, capacity 128, and output length 128. Further, its initial state is given by $0^{256} \parallel \mathsf{IV}$. First, $\mathcal{B}$ runs $\mathcal{A}$, which outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. For every query that $\mathcal{A}$ makes to ρ, the adversary $\mathcal{B}$ makes the same query to its own permutation and sends the response back to $\mathcal{A}$. Then $\mathcal{B}$ computes the state S_i (and respectively $\overline{S}_i$) after the i-th application of the permutation in $\textsc{Schwaemm}_{\mathsf{IV}}$ evaluated in (K, N, A, M) (and respectively $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$). Denote by $S_{i,r}$ and $S_{i,c}$ (and respectively $\overline{S}_{i,r}$ and $\overline{S}_{i,c}$) the rate and the capacity part of the state S_i (and respectively $\overline{S}_i$). The adversary $\mathcal{B}$ then outputs

$$\begin{aligned}
X = K \parallel N \parallel \\
&(S_{1,r} \oplus \xi_2(S_{1,r}, A_1) \oplus \omega_{c,r}(S_{1,c})) \parallel \ldots \parallel \\
&(S_{\alpha,r} \oplus \xi_2(S_{\alpha,r}, A_\alpha) \oplus \omega_{c,r}(S_{\alpha,c})) \parallel \\
&(S_{\alpha+1,r} \oplus \xi_2(S_{\alpha+1,r}, M_1) \oplus \omega_{c,r}(S_{\alpha+1,c})) \parallel \ldots \parallel \\
&(S_{\alpha+\mu,r} \oplus \xi_2(S_{\alpha+\mu,r}, M_\mu) \oplus \omega_{c,r}(S_{\alpha+\mu,c}) \\
\overline{X} = \overline{K} \parallel \overline{N} \parallel \\
&(\overline{S}_{1,r} \oplus \xi_2(\overline{S}_{1,r}, \overline{A}_1) \oplus \omega_{c,r}(\overline{S}_{1,c})) \parallel \ldots \parallel \\
&(\overline{S}_{\overline{\alpha},r} \oplus \xi_2(\overline{S}_{\overline{\alpha},r}, \overline{A}_{\overline{\alpha}}) \oplus \omega_{c,r}(\overline{S}_{\overline{\alpha},c})) \parallel \\
&(\overline{S}_{\overline{\alpha}+1,r} \oplus \xi_2(\overline{S}_{\overline{\alpha}+1,r}, \overline{M}_1) \oplus \omega_{c,r}(\overline{S}_{\overline{\alpha}+1,c})) \parallel \ldots \parallel \\
&(\overline{S}_{\overline{\alpha}+\overline{\mu},r} \oplus \xi_2(\overline{S}_{\overline{\alpha}+\overline{\mu},r}, \overline{M}_{\overline{\mu}}) \oplus \omega_{c,r}(\overline{S}_{\overline{\alpha}+\overline{\mu},c})\,,
\end{aligned}$$

which guarantees that $\mathcal{H}(X)$ (and $\mathcal{H}(\overline{X})$, respectively) models $\textsc{Schwaemm}_{\mathsf{IV}}$ evaluated on

(K, N, A, M) (and $(\overline{K}, \overline{N}, \overline{A}, \overline{M})$, respectively). More precisely, instead of the state-update-function, XORing of the input, and rate-whitening applied in $\textsc{Schwaemm}_{\mathsf{IV}}$, for $\mathcal{H}$ we XOR a suitable value which imitates these operations. A visualization for this is provided in Fig. 35.

Next, we show that if $\mathcal{A}$ wins the game $\mathsf{TagColl}$ against $\textsc{Schwaemm}_{\mathsf{IV}}$, then the constructed adversary $\mathcal{B}$ wins the game $\mathsf{ShiftedColl}_{128}$ against $\mathcal{H}$. First observe that $\mathcal{A}$ winning implies that $(K, N, A, M) \neq (\overline{K}, \overline{N}, \overline{A}, \overline{M})$ and the corresponding tags T and $\overline{T}$—computed with $\textsc{Schwaemm}_{\mathsf{IV}}$—agree. Note that the latter implies that $S_{\alpha+\mu+1,c} \oplus K = \overline{S}_{\overline{\alpha}+\overline{\mu}+1,c} \oplus \overline{K}$, hence by choice of X and $\overline{X}$ it holds that $\mathcal{H}(X) \oplus \lceil X \rceil_{128} = \mathcal{H}(\overline{X}) \oplus \lceil \overline{X} \rceil_{128}$. Further, the fact that $(K, N, A) \neq (\overline{K}, \overline{N}, \overline{A})$, implies that $X \neq \overline{X}$: for $(K, N) \neq (\overline{K}, \overline{N})$ this is obvious while for $(K, N) = (\overline{K}, \overline{N})$ and $A \neq \overline{A}$, a simple analysis shows that X and $\overline{X}$ differ at the point where the associated data blocks differ for the first time. This implies that $\mathcal{B}$ wins the game $\mathsf{ShiftedColl}_{128}$.

Using the indifferentiability of sponges (cf. Theorem 10), we can replace $\mathcal{H}$ by a random function F, as there exists an efficient simulator for the underlying permutation such that $\mathcal{A}$ cannot distinguish between $\mathcal{H}$ and F. This yields

$$\mathbf{Adv}_{\mathcal{H}}^{\mathsf{ShiftedColl}_{128}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathrm{F}}^{\mathsf{ShiftedColl}_{128}}(\mathcal{A}) + \epsilon\,,$$

for $\epsilon > \frac{(1-2^{-256})q^2+(1+2^{-256})q}{2^{129}}$, which results from the application of Theorem 10.

As a last step, we observe that for a random function $\mathrm{F} : \{0,1\}^{\geq 128} \to \{0,1\}^{128}$, it is unlikely that $\mathcal{B}$ wins the game $\mathsf{ShiftedColl}_{128}$. For this, note that an adversary that wins the game $\mathsf{ShiftedColl}_{128}$ against F with q queries, finds a collision in the following list of uniformly distributed elements

$$L = \{\mathrm{F}(X_1) \oplus \lceil X_1 \rceil_{128}, \mathrm{F}(X_2) \oplus \lceil X_2 \rceil_{128}, \ldots, \mathrm{F}(X_q) \oplus \lceil X_q \rceil_{128}\}\,,$$

for $X_i \in \{0,1\}^{\geq 128}$ being the inputs $\mathcal{B}$ queries to F. By Theorem 11, the probability for this is bounded above by

$$1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right)\,,$$

for $q \leq 2^{127}$. In total, we obtain

$$\begin{aligned}\mathbf{Adv}_{\mathcal{H}}^{\mathsf{ShiftedColl}_{128}}(\mathcal{A}) &\leq \mathbf{Adv}_{\mathrm{F}}^{\mathsf{ShiftedColl}_{128}}(\mathcal{A}) + \epsilon \\ &\leq 1 - \exp\left(\frac{-q(q-1)}{2^{128}}\right) + \epsilon\,,\end{aligned}$$

which finishes the proof of the theorem. □

C Deferred Proofs

C.1 Proof of Lemma 2

Proof. We construct $\mathcal{A}_{\mathcal{C}}$ as shown in Fig. 5. As input it receives (K, N, S). It chooses an arbitrary even number of associated data block α and chooses all except the first block at random, i.e., $A_2, \ldots, A_\alpha \leftarrow_{\$} \{0,1\}^n$.[32] In addition, $\mathcal{A}$ sets $A_{\alpha+1} \leftarrow 0^n$.[33] Then $\mathcal{A}$ proceeds by inverting S using the ideal tweakable cipher to obtain Y. For $i \in \{1, \ldots, \frac{\alpha}{2}\}$, $\mathcal{A}$ first computes $S \leftarrow Y \oplus A_{2i+1}$ followed by computing $Y \leftarrow \widetilde{\mathsf{E}}^{-1}(K, A_{2i}, S)$, inverting

[32] We assume that these blocks are chosen to exhibit a valid padding.

[33] This corresponds to the input of the last application of ξ in the upper part of Fig. 3.

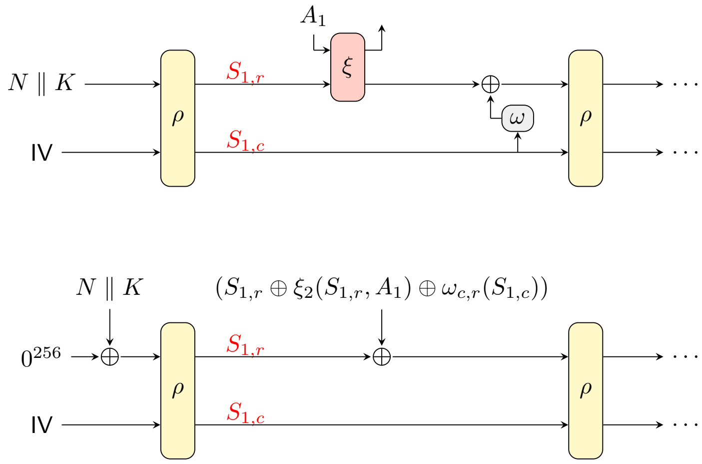


Figure 35: Illustration of a proof step for $\text{SCHWAEMM}_{\mathsf{IV}}$ (Theorem 17). $\text{SCHWAEMM}_{\mathsf{IV}}$ is represented as a plain sponge as is shown for the first XOR in the above figure.

Game $\mathsf{ShiftedColl}_\kappa$

$X, \overline{X} \leftarrow \mathcal{A}()$
if $X = \overline{X}$
  return 0
return $(\mathrm{F}(X) \oplus \mathrm{F}(\overline{X}) = \lceil X \rceil_\kappa \oplus \lceil \overline{X} \rceil_\kappa)$

Figure 36: Security game $\mathsf{ShiftedColl}_\kappa$ defined for a function $\mathrm{F}\colon \{0,1\}^{\geq\kappa} \rightarrow \{0,1\}^\kappa$ and used in the proof for $\text{SCHWAEMM}_{\mathsf{IV}}$ (Theorem 17).

the state-update-function ξ and the ideal tweakable cipher. Denote the resulting state by Y_* (see also the state marked in red in Fig. 3). Adversary $\mathcal{A}$ sets the first associated data as $A_1 \leftarrow Y_*$ which ensures that $\text{ENC}_{\mathcal{C}}(K, N, A) = S$.

The number of queries to the ideal tweakable cipher $\widetilde{\mathsf{E}}$ by $\mathcal{A}$ is $\lfloor \frac{\alpha}{2} \rfloor + 1$: the initial one plus $\lfloor \frac{\alpha}{2} \rfloor$ for the for-loop. □

C.2 Proof of Lemma 3

Proof. We construct adversary $\mathcal{A}_{\mathcal{M}}$ as shown in Fig. 5 which gets $(K, N, (C, T))$ as input. For ease of exposition, we assume that the length of C is a multiple of the block size n.[34] Let $C_1, \ldots, C_\gamma \xleftarrow{n} C$. The adversary $\mathcal{A}_{\mathcal{M}}$ first sets $Y \leftarrow T$ and then computes $S \leftarrow G^{-1}(Y)$. For $i \in \{1, \ldots, \gamma\}$, $\mathcal{A}_{\mathcal{M}}$ computes $Y \leftarrow \widetilde{\mathsf{E}}^{-1}(K, N, S)$[35] followed by the computation of $(S, M_i) \leftarrow \mathcal{A}_\xi(Y, C_i)$ from Lemma 1. Finally, $\mathcal{A}_{\mathcal{M}}$ outputs (S, M), where $M = M_1 \parallel \ldots \parallel M_\gamma$. By construction, it holds that $\text{ENC}_{\mathcal{M}}(K, N, S, M) = (C, T)$ as $\mathcal{A}_{\mathcal{M}}$ inverted all invocations of $\widetilde{\mathsf{E}}$ and ξ during $\text{ENC}_{\mathcal{M}}$—using Lemma 1 for the latter.

$\mathcal{A}$ queries the ideal tweakable cipher $\widetilde{\mathsf{E}}$ a total of μ times. □

C.3 Proof of Lemma 4

Proof. For the proof of the claim, we define the following auxiliary events:

1. Event $\mathsf{E_t}$ (target hitting query):

[34] This is justified by letting $\mathcal{A}$ choose a message satisfying this.
[35] Note that we drop the counter which is part of the tweak for simplicity.

$\mathcal{A}$ makes a query Y to ρ such that $\lfloor \rho(Y) \rfloor_{c^\star} = \mathsf{IV}$ or $\mathcal{A}$ makes a query S to ρ^{-1} such that $\lfloor \rho^{-1}(S) \rfloor_{c^\star} = \mathsf{IV}$.

2. Event $\mathsf{E_c}$ (colliding queries):

   $\mathcal{A}$ makes queries $Y \neq \overline{Y}$ to ρ such that $\lfloor \rho(Y) \rfloor_c = \lfloor \rho(\overline{Y}) \rfloor_c$ or $\mathcal{A}$ makes queries Y to ρ and $\overline{S}$ to ρ^{-1} such that $\lfloor \rho(Y) \rfloor_c = \lfloor \rho^{-1}(\overline{S}) \rfloor_c$.

Next, we show that if $\mathcal{A}$ triggers PP, then it triggers one of the events defined above. For the proof assume that PP holds and denote the problematic paths $\mathcal{A}$ finds by $(P, \overline{P})$. We first consider the case that there is at least one backward edge in $(P, \overline{P})$. We assume without loss of generality that P contains at least one backward edge. Note that for each PS-path that contains at least one backward edge, we can define a corresponding minimal PS-path containing exactly one backward edge. To do this—starting from the end of the path—all edges are removed, until the last edge of the path is a backward edge and all other edges (if any remain) are forward edges. For sake of simplicity, we write P also for the minimal path corresponding to P in the following and write it as

$$P = Z_0|Y_0 \to \cdots \to S_{s-2}|Z_{s-2}|Y_{s-2} \to S_{s-1}|Z_{s-1}|Y_{s-1} \to S_s\,.$$

We further distinguish the following two sub-cases:

Case 1: $s = 1$

The path is simply $Z_0|Y_0 \to S_1$ and $\mathcal{A}$ queried S_1 to ρ^{-1}. By construction, we have $\lfloor Y_0 \rfloor_{c^\star} = \mathsf{IV}$ and $\rho^{-1}(S_1) = Y_0$, hence in particular $\lfloor \rho^{-1}(S_1) \rfloor_{c^\star} = \lfloor Y_0 \rfloor_{c^\star} = \mathsf{IV}$. Thus $\mathcal{A}$'s query triggered event $\mathsf{E_t}$.

Case 2: $s \geq 2$

The path is $Z_0|Y_0 \to \cdots \to S_{s-2}|Z_{s-2}|Y_{s-2} \to S_{s-1}|Z_{s-1}|Y_{s-1} \to S_s$. Except for the last edge, all edges are forward edges. By construction, we have $\lfloor S_{s-1} \rfloor_c = \lfloor Y_{s-1} \rfloor_c$. Furthermore, S_{s-1} is the result of querying Y_{s-2} to ρ (forward edge) and Y_{s-1} is the result of querying S_s to ρ^{-1} (backward edge). This yields that these two queries trigger event $\mathsf{E_c}$.

We now consider the case that the penultimate states Y_{s-1} and $\overline{Y}_{\overline{s}-1}$ are equal, but P and $\overline{P}$ contain no backward edges. For such a pair of paths $(P, \overline{P})$, we define the corresponding minimal pair of PS-paths by choosing $s + \overline{s}$ minimal such that $(Z_0, \ldots, Z_{s-1}) \neq (\overline{Z}_0, \ldots, \overline{Z}_{\overline{s}-1})$ and $Y_{s-1} = \overline{Y}_{\overline{s}-1}$ still hold. We consider the minimal pair of paths corresponding to $(P, \overline{P})$ and—for sake of simplicity—also denote them by $(P, \overline{P})$. As before we use the following representation

$$\begin{aligned} P &= Z_0|Y_0 \to \cdots \to S_{s-2}|Z_{s-2}|Y_{s-2} \to S_{s-1}|Z_{s-1}|Y_{s-1} \to S_s \\ \overline{P} &= \overline{Z}_0|\overline{Y}_0 \to \cdots \to \overline{S}_{\overline{s}-2}|\overline{Z}_{\overline{s}-2}|\overline{Y}_{\overline{s}-2} \to \overline{S}_{\overline{s}-1}|\overline{Z}_{\overline{s}-1}|\overline{Y}_{\overline{s}-1} \to \overline{S}_{\overline{s}}\,. \end{aligned}$$

Without loss of generality, we further assume $s \leq \overline{s}$ and distinguish between the following three sub-cases:

Case 1: $s = 1 \wedge \overline{s} = 1$

The paths are simply $Z_0|Y_0 \to S_1$ and $\overline{Z}_0|\overline{Y}_0 \to \overline{S}_1$. By construction, we have $\lceil Y_0 \rceil_r = Z_0$ and $\lceil \overline{Y}_0 \rceil_r = \overline{Z}_0$ which leads to a contradiction as $Z_0 \neq \overline{Z}_0$ (recall that problematic paths have different inputs) but $Y_0 = \overline{Y}_0$. Thus, this case cannot occur.

Case 2: $s = 1 \wedge \overline{s} \geq 2$

The paths are

$$\begin{aligned} &Z_0|Y_0 \to S_1 \\ &\overline{Z}_0|\overline{Y}_0 \to \cdots \to \overline{S}_{\overline{s}-2}|\overline{Z}_{\overline{s}-2}|\overline{Y}_{\overline{s}-2} \to \overline{S}_{\overline{s}-1}|\overline{Z}_{\overline{s}-1}|\overline{Y}_{\overline{s}-1} \to \overline{S}_{\overline{s}}\,. \end{aligned}$$

We have $\lfloor Y_0 \rfloor_{c^\star} = \mathsf{IV}$ by construction and $Y_0 = \overline{Y}_{\overline{s}-1}$ by assumption. This allows to deduce $\lfloor \overline{S}_{\overline{s}-1} \rfloor_{c^\star} = \mathsf{IV}$. Since $\overline{S}_{\overline{s}-1}$ is the result of querying $\overline{Y}_{\overline{s}-2}$ to ρ, this query triggered event $\mathsf{E_t}$.

Case 3: $s \geq 2 \wedge \overline{s} \geq 2$

The paths are

$$Z_0|Y_0 \to \cdots \to S_{s-2}|Z_{s-2}|Y_{s-2} \to S_{s-1}|Z_{s-1}|Y_{s-1} \to S_s$$
$$\overline{Z}_0|\overline{Y}_0 \to \cdots \to \overline{S}_{\overline{s}-2}|\overline{Z}_{\overline{s}-2}|\overline{Y}_{\overline{s}-2} \to \overline{S}_{\overline{s}-1}|\overline{Z}_{\overline{s}-1}|\overline{Y}_{\overline{s}-1} \to \overline{S}_{\overline{s}}\,.$$

Consider the penultimate edges of both paths, i.e., the edges from Y_{s-2} to S_{s-1} and from $\overline{Y}_{\overline{s}-2}$ to $\overline{S}_{\overline{s}-1}$, which are both forward edges. By assumption, $Y_{s-1} = \overline{Y}_{\overline{s}-1}$ holds. We have to distinguish two more cases based on the inputs Z_{s-1} and $\overline{Z}_{\overline{s}-1}$:

Case 3.1: $Z_{s-1} = \overline{Z}_{\overline{s}-1}$

From $Y_{s-1} = \overline{Y}_{\overline{s}-1}$ and $Z_{s-1} = \overline{Z}_{\overline{s}-1}$, we can conclude that $S_{s-1} = \overline{S}_{\overline{s}-1}$. Then, however, one can obtain a pair of shorter paths by dropping the last edges of both P and $\overline{P}$ while maintaining the desired property. Thus this case is impossible as it contradicts the minimality of the paths.

Case 3.2: $Z_{s-1} \neq \overline{Z}_{\overline{s}-1}$

As $Y_{s-1} = \overline{Y}_{\overline{s}-1}$ and $Z_{s-1} \neq \overline{Z}_{\overline{s}-1}$, we can deduce $S_{s-1} \neq \overline{S}_{\overline{s}-1}$ and $\lfloor S_{s-1} \rfloor_c = \lfloor \overline{S}_{\overline{s}-1} \rfloor_c$. Then $Y_{s-2} \neq \overline{Y}_{\overline{s}-2}$ and the queries on Y_{s-2} and $\overline{Y}_{\overline{s}-2}$ triggered event $\mathsf{E_c}$.

Collecting the above, yields

$$\Pr[\mathsf{PP}] \leq \Pr[\mathsf{E_t} \vee \mathsf{E_c}] \leq \Pr[\mathsf{E_t}] + \Pr[\mathsf{E_c}]\,.$$

We apply Lemma 6 for $c^\star$ and obtain $\Pr[\mathsf{E_t}] \leq \frac{q}{2^{c^\star-1}}$ and once again for c, which gives $\Pr[\mathsf{E_c}] \leq \frac{q(q-1)}{2^c}$. □

C.4 Proof of Theorem 3

Proof. In the proof of [BS23, Theorem 8.6], the advantage is decomposed as follows

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{H}}(\mathcal{A}) = \Pr[\mathsf{CP}] \leq \Pr[\mathsf{CP} \wedge \neg\mathsf{PP}] + \Pr[\mathsf{PP}]\,,$$

for CP the event that $\mathcal{A}$ finds a pair of colliding paths. Bounding the first summand works as in the original proof—hence we do not repeat it here—and results in $\Pr[\mathsf{CP} \wedge \neg\mathsf{PP}] \leq \frac{q(q-1)}{2^w}$. For the second summand, we apply Lemma 4, which gives $\Pr[\mathsf{PP}] \leq \frac{q}{2^{c^\star-1}} + \frac{q(q-1)}{2^c}$. □

C.5 Proof of Theorem 5

Proof. Adversary $\mathcal{B}$ simply runs $(K, N, A), (\overline{K}, \overline{N}, \overline{A}) \leftarrow \mathcal{A}$, picks an arbitrary message M, and outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, M)$. Since $\mathcal{A}$ wins the game $\mathsf{KeyColl}$, it holds that $K \neq \overline{K}$ and

$$\textsc{Enc}_{\mathcal{C}}(K, N, A) = \textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A}) =: S_*\,.$$

Using the fact that $\textsc{Ae}$ is a full-CpP scheme, we get

$$\begin{aligned}
\textsc{zp-Ae.Enc}(K, N, A, M) &= \textsc{Enc}_{\mathcal{M}}(\textsc{Enc}_{\mathcal{C}}(K, N, A), 0^z \parallel M) \\
&= \textsc{Enc}_{\mathcal{M}}(S_*, 0^z \parallel M) \\
&= \textsc{Enc}_{\mathcal{M}}(\textsc{Enc}_{\mathcal{C}}(\overline{K}, \overline{N}, \overline{A}), 0^z \parallel M) \\
&= \textsc{zp-Ae.Enc}(\overline{K}, \overline{N}, \overline{A}, M)\,.
\end{aligned}$$

Thus, $\mathcal{B}$ wins game $\mathsf{CMT_K}$. □

C.6 Proof of Theorem 7

Proof. For sake of simplicity, we give the proof for the case $z = n$. The adversary $\mathcal{A}$ generates an arbitrary context-message-pair (K, N, A, M) and computes the ciphertext $(C, T) \leftarrow$ zp-Elephant.Enc(K, N, A, M). Let $M_1, \ldots, M_\mu \xleftarrow{n} \mathtt{pad}_{0^*}(0^z \parallel M, n)$ be the μ message blocks, where it holds that $M_1 = 0^n$. Therefore, the ciphertext consists of μ n-bit blocks $C = C_1 \parallel \ldots \parallel C_\mu$, where $C_1 = \widetilde{\mathsf{E}}(K, (0,1), N) \oplus M_1 = \widetilde{\mathsf{E}}(K, (0,1), N)$. In order to do these computations, $\mathcal{A}$ needs to make μ queries to $\widetilde{\mathsf{E}}$.

Next, $\mathcal{A}$ chooses $\overline{K} \neq K$ at random to make sure that the attack is a valid CMT$_{\mathsf{K}}$ attack. To find the second nonce $\overline{N}$, $\mathcal{A}$ queries $(\overline{K}, (0,1), C_1)$ to $\widetilde{\mathsf{E}}^{-1}$ and sets the response to be $\overline{N}$. This guarantees that the first ciphertext block C_1 is an encryption of the all-zero message when using $\overline{N}$. To find $\overline{M}$, $\mathcal{A}$ computes $\overline{M}_i \leftarrow \widetilde{\mathsf{E}}(\overline{K}, (i-1, 1), \overline{N}) \oplus C_i$, for $i \in \{2, \ldots, \mu\}$. This then yields that (K, N, M) and $(\overline{K}, \overline{N}, \overline{M})$ yield the same ciphertext, i.e.,

$$\textsc{Enc}_{\mathcal{M}}(\overline{K}, \overline{N}, 0^z \parallel \overline{M}) = C = \textsc{Enc}_{\mathcal{M}}(K, N, 0^z \parallel M)\,.$$

Next, $\mathcal{A}$ chooses arbitrary A and computes $T \leftarrow \textsc{Enc}_{\mathcal{T}}(K, N, A, C)$. It remains to find $\overline{A}$ that result in the same tag, i.e., $\textsc{Enc}_{\mathcal{T}}(\overline{K}, \overline{N}, \overline{A}, C) = T$. The attack is effectively the existing committing attack against Elephant (cf. Theorem 12); the mere difference is that $\mathcal{A}$ targets the second associated data block since it has already fixed $\overline{N}$ which determines part of the first associated data block. Adversary $\mathcal{A}$ then outputs $(K, N, A, M), (\overline{K}, \overline{N}, \overline{A}, \overline{M})$. From the above we can conclude that the outputs by $\mathcal{A}$ result in the same ciphertext (C, T), which, together with the fact that $K \neq \overline{K}$, yields that

$$\mathbf{Adv}^{\mathsf{CMT_K}}_{\text{zp-Elephant}}(\mathcal{A}) = 1\,.$$

The number of queries to $\widetilde{\mathsf{E}}$ by $\mathcal{A}$ accumulate to $2\mu + 2\gamma + \alpha + \overline{\alpha}$: 2μ for computing (K, N, M) and $(\overline{K}, \overline{N}, \overline{M})$; and $2\gamma + \alpha + \overline{\alpha}$ for computing A and $\overline{A}$.

The proof generalizes to the case $z \leq n$ by letting $\mathcal{A}$ choose the message such that the first bits are 0^{n-z}, which ensures that the first message block is 0^n. □

C.7 Proof of Theorem 8

Proof. The adversary $\mathcal{A}$ picks a key-nonce pair (K, N) at random. It then picks an arbitrary message M and computes $C \leftarrow \textsc{Enc}_{\mathcal{M}}(K, N, 0^z \parallel M)$—making μ queries to ρ_3 and $\nu + 1$ queries to ρ_1. Subsequently, the adversary applies a birthday attack on the tag, by computing $\textsc{Enc}_{\mathcal{T}}(K, N, \cdot, C)$ for different associated data. By [BS23, Section 8.3], a collision is found with probability $\frac{1}{2}$ after trying $2^{\frac{\tau}{2}+2} + 1$ different associated data. Let A and $\overline{A}$ be the associated data that led to a collision as part of the birthday attack. Finally, $\mathcal{A}$ outputs $(K, N, A, M), (K, N, \overline{A}, M)$.

For each associated data, $\mathcal{A}$ makes $\alpha + \gamma + 2$ queries to ρ_2 (evaluating $\textsc{Enc}_{\mathcal{T}}$) and $\kappa + 1$ queries to ρ_1 (evaluating Isap.Rk). In total, the queries by $\mathcal{A}$ are $q_1 = (2^{\frac{\tau}{2}+1} + 1)(\kappa + 1) + \nu + 1$ queries to ρ_1, $q_2 = (2^{\frac{\tau}{2}+1} + 1)(\alpha + \gamma + 2)$ queries to ρ_2, and $q_3 = \mu$ queries to ρ_3. □