ZCZ – Achieving n-bit SPRP Security with a Minimal Number of Tweakable-block-cipher Calls

Ritam Bhaumik[1], Eik List[2], and Mridul Nandi[1]⋆

[1] Indian Statistical Institute, Kolkata, India
[2] Bauhaus-Universität Weimar, Weimar, Germany
bhaumik.ritam@gmail.com, eik.list@uni-weimar.de, mridul.nandi@gmail.com

Abstract. Strong Pseudo-random Permutations (SPRPs) are important for various applications. In general, it is desirable to base an SPRP on a single-keyed primitive for minimizing the implementation costs. For constructions built on classical block ciphers, Nandi showed at ASIACRYPT'15 that at least two calls to the primitive per processed message block are required for SPRP security, assuming that all further operations are linear. The ongoing trend of using tweakable block ciphers as primitive has already led to MACs or encryption modes with high security and efficiency properties. Thus, three interesting research questions are hovering in the domain of SPRPs: (1) if and to which extent the bound of two calls per block can be reduced with a tweakable block cipher, (2) how concrete constructions could be realized, and (3) whether full n-bit security is achievable from primitives with n-bit state size.

The present work addresses all three questions. Inspired by Iwata et al.'s ZHash proposal at CRYPTO'17, we propose the ZCZ (ZHash-Counter-ZHash) construction, a single-key variable-input-length SPRP based on a single tweakable block cipher whose tweak length is at least its state size. ZCZ possesses close to optimal properties with regards to both performance and security: not only does it require only asymptotically $3\ell/2$ calls to the primitive for ℓ-block messages; we show that this figure is close to the minimum by an PRP distinguishing attack on any construction with tweak size of $\tau = n$ bits and fewer than $(3\ell - 1)/2$ calls to the same primitive. Moreover, it provides optimal n-bit security for a primitive with n-bit state and tweak size.

Keywords: Symmetric-key cryptography · provable security · variable-input-length SPRP · tweakable block cipher · encryption

1 Introduction

SPRPs. Strong Pseudo-Random Permutations (or wide-block ciphers), are important symmetric-key schemes for protecting the privacy of variable-length mes-

⋆ The research by Mridul Nandi has been supported by the Wisekey project at the R. C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata.

sages. Their tweakable variants (STPRPs) are useful to build strong authenticated encryption [25, 54] or onion AE [50]. During the previous two decades, the symmetric-key community proposed a considerable corpus of SPRPs. From a high-level point of view, existing constructions could be categorized into (1) Generalized Feistel networks, (2) Encrypt-Mix-Encrypt, (3) Hash-ECB-Hash, (4) Hash-Counter-Hash, and (5) miscellaneous designs.

Optimization Goals. The primary goals for optimizations in cryptographic schemes are, in general, low implementation costs, high provable security guarantees, and high performance. For the first criterion, it is desirable to construct higher-level schemes from a single well-analyzed primitive without large internal state and with a single key.

High security is essential in many domains that have to process large amounts of data without the ability of frequent re-keying. In most constructions, however, it comes at the cost of decreased performance. Unsurprisingly, the challenges of combining high security guarantees with high performance have been identified among the hot topics of symmetric-key cryptography at the ESC 2017 workshop [8]. Often, high security is associated with security *beyond the birthday bound*. In the areas of authentication (e.g., [33, 56, 57]), encryption, as well as authenticated encryption (e.g., [26, 27, 48]), beyond-birthday security has undergone a long line of research. In the area of SPRPs, however, the security of the vast majority of existing constructions is still limited by the birthday bound of $n/2$ bits, where n is the state size of the underlying primitive. So, the privacy guarantees are lost if $q \simeq 2^{n/2}$ message blocks have been encrypted under the same key. Assuming the AES as primitive, this would imply that significantly fewer than 2^{64} blocks could safely be encrypted under a single key.

Security of SPRPs: State of the Art. Among the earlier proposals, the LargeBlock1 and LargeBlock2 constructions by Minematsu and Iwata [39] as well as TCT_2 by Shrimpton and Terashima [54] are exceptional for their security guarantees. The LargeBlock designs can achieve optimal n-bit security, whereas TCT_2 is limited by $2n/3$ bits. Both share similarities to the Ψ_2 and Ψ_3 constructions from Coron et al. [17], which use two and three calls to a tweakable block cipher. Both LargeBlock2 and TCT_2 possess a sandwich structure, where an encryption layer is wrapped by two layers of hashing. In the former, the encryption layer is an application of Ψ_2 in ECB-mode; the hashing layers employs two calls to a polynomial hash of $2(\ell - 1)$multiplications each. TCT_2 can be seen as an unbalanced version of Ψ_3, where also $2(\ell - 1)$ of ℓ input blocks are hashed in each hashing layer. Both constructions are remarkable for their time. To be comparably efficient, however, they required two primitives, a block cipher and a universal hash function.

A different direction is followed by HHFHFH [5] and its instantiations (e.g., [6]), which is a four-round unbalanced Feistel network, built on a large-state primitive. Instead of providing beyond-birthday security, it possesses large security margins due to a larger birthday bound of their internal primitives. However, the large state size limits its efficiency.

The only approach we are aware of that almost combines both security and performance desiderata is Simpira (v2) [19], a family of Feistel-like constructions built upon the AES round function. Its authors claim 128-bit security and high performance on current processors with support for AES native instructions. However, Simpira's security claim stems purely from heuristics, which will demand intensive further cryptanalysis to increase trust into it.

Tweakable Block Ciphers. One established approach for achieving higher security without considerably sacrificing performance is to use a tweakable block cipher (TBC) [32] as underlying primitive. At the core, tweakable block ciphers employ an additional public input called tweak, which allows to efficiently separate the domains of different calls to the primitive. This fact can reduce the impact of internal collisions on the security of the scheme built around them. For message authentication codes (MACs), a series of recent works pushed the security bounds further [16, 28, 41], but a similar trend is also observable in the domain of encryption modes and authenticated encryption schemes [27, 31, 37, 48, 49]. This approach has also been used earlier for SPRPs [17, 36, 38, 39, 54] – those proposals, however, originate from at least half a decade ago where TBCs used to be constructed in cumbersome fashion from classical block ciphers. Nowadays, we have the option of using efficient dedicated TBCs, such as Deoxys-BC, Joltik-BC [30], or Skinny [3].

The application of TBCs can also boost the efficiency of constructions, as has been demonstrated recently for MACs. At CRYPTO'17, Iwata et al. [28] introduced ZMAC, a TBC-based parallelizable, single-key single-primitive MAC whose internal hash function ZHash processed the message in both the tweak and plaintext simultaneously. The additional message bits per primitive call render ZMAC more efficient than previous MACs and suggest the adoption of the approach to other domains.

Open Research Questions. When abstracting away the details of the primitive, the number of calls to it per input block becomes the main efficiency metric. From Encrypt-Mix-Encrypt-based constructions, it is well-known that the bound is at most two calls per block (plus some minor overhead), assuming all further operations are linear. Thus, it is an interesting question if SPRPs can be built from fewer calls to a single-keyed primitive. Moreover, a strongly related question is that for the minimal number of calls necessary for SPRP security.

From a theoretical perspective, Nandi [44] showed that constructions built from a classical single-keyed block cipher require 2ℓ calls for ℓ-block messages for SPRP security. Though, it seems as though this bound is reducible if one used a TBC instead as the underlying primitive. For Hash-Counter-Hash-based constructions, the most efficient (T)BC-based hash function we are aware of is ZHash. For a TBC with n-bit state and τ-bit tweak length, it would yield a construction of about $\ell + 2\lceil \sigma/(n+\tau) \rceil$ calls for messages of σ bits. For dedicated TBCs, such as Deoxys-BC-128-384 or Skinny-128-384, this figure still implies that approximately $5\ell/3$ calls are necessary. Regarding the other design principles, it is unclear if similar results are applicable to constructions based

Table 1: Asymptotic #primitive calls for SPRP paradigms. We assume that hash functions and encryption layers use a single-keyed (tweakable) block cipher with n-bit state and τ-bit tweak size to encrypt an ℓ-block message of σ bits in total. We assume the hashing layers use ZHash (as the most efficient blockcipher-based hash function we are aware of).

	#Block-cipher calls			
Paradigm	Top	Middle	Bottom	Total (asympt.)
LargeBlock2	$2\lceil(\ell-1)/2\rceil$	ℓ	$2\lceil(\ell-1)/2\rceil$	$4\lceil(\ell-1)/2\rceil+\ell$
TCT_2	$2\lceil(\ell-1)/2\rceil$	$2(\ell-1)$	$2\lceil(\ell-1)/2\rceil$	$4\lceil(\ell-1)/2\rceil+2\ell$
Encrypt-Mix-Encrypt	ℓ	$\lceil\ell/n\rceil$	ℓ	$2\ell+\lceil\ell/n\rceil$
Hash-ECB-Hash	ℓ	ℓ	ℓ	3ℓ
Hash-Counter-Hash	$\lceil\sigma/(n+\tau)\rceil$	ℓ	$\lceil\sigma/(n+\tau)\rceil$	$\ell+2\lceil\sigma/(n+\tau)\rceil$
ZCZ	$\ell/2$	$\ell/2+\lceil\ell/2n\rceil$	$\ell/2$	$3\ell/2+\lceil\ell/2n\rceil$

on the Encrypt-Mix-Encrypt or Hash-ECB-Hash paradigms. We estimate that Hash-ECB-Hash constructions would need about ℓ primitive calls in each hashing layer, plus ℓ calls in the encryption layer. An instantiation of LargeBlock2 with ZHash instead of multiplications would yield $2\lceil(\ell-1)/2\rceil$ calls in each hashing layer, plus ℓ calls in the middle, or 3ℓ calls in sum. TCT_2 could use a ZHash layer each for both top and bottom hashing layer. While further modifications could make it more efficient, its proposal employed $2\ell-2$ calls in the middle. We compare the approaches in Table 1. Altogether, three interesting research questions remain: (1) to which extent can the number of primitive calls be reduced when employing a tweakable block cipher, (2) how can a specific construction be realized, and (3) can it be built with high provable security guarantees.

Contribution. This work tries to answer all three questions above: for the theoretical interest, (1) we show that 1.5ℓ primitive calls per message block is close to minimal by a generic distinguisher on any construction that employs fewer than $(3\ell-1)/2$ calls to a single-keyed primitive per message block, where all further operations are linear. For the practitioner's interest, (2) we propose ZCZ (ZHash-Counter-ZHash), an almost fully parallelizable variable-input-length SPRP based on a single-keyed TBC with n-bit state and n-bit tweak size. ZCZ matches approximately the optimal number of 1.5ℓ calls to the primitive for an ℓ-block message, plus a small overhead. Finally, we show (3) that ZCZ achieves optimal n-bit security, i.e., the SPRP advantage of any adversary that asks at most q queries of σ blocks in total is in $O(\sigma^2/2^{2n})$.

We note that instantiations of Hash-Counter-Hash with ZHash and a TBC with large tweaks of $\tau=3n$, the number of primitive calls could become equal to that of ZCZ. However, such primitives would introduce a significant slowdown, be it due to the requirements of more rounds in a TWEAKEY-like cipher, or due to the need of calling an additional universal hash function for compressing the tweak. Concerning practical tweak sizes $\tau<3n$, the number of calls is significantly lower for our construction.

YET ANOTHER ENCRYPTION SCHEME? It may appear that ZCZ is yet another encryption scheme after all, and with hundreds of encryption schemes already being present in the literature, it is difficult get excited about another one, notwithstanding small improvements in performance and security. We beg to differ on this point primarily for two reasons: (1) very few existing encryption schemes built upon a primitive with an n-bit output provide n-bit security — most in fact are only secure up to the birthday bound. As such, the improvement by ZCZ in terms of security is not a small step, but rather a leap. Since there is a considerable interest in the (still) small group of constructions that achieve this security, we believe that our encryption scheme is an exciting addition to this group. (2) Even more significant is the way that ZCZ exploits the randomness generated by a tweakable blockcipher. While most previous approaches were based on generic replacements of two or more blockcipher calls by a single call to a tweakable block cipher, the approach used by ZCZ is not a corollary of any previous work. Given its efficiency, we believe it can lead to exciting new directions in research on tweakable-blockcipher modes.

OUTLINE. The remainder is structured as follows: first, Section 2 briefly summarizes the necessary preliminaries. Given a primitive with an effective tweak size[3] $\tau = n$, Section 3 illustrates that every PRP with fewer than $3\ell - 1$ primitive calls for 2ℓ-block messages is insecure, which was the core motivation for our search for constructions with about 1.5ℓ calls. Subsequently, Section 4 defines our basic construction, which is first described for messages whose length is a positive multiple of $2n$ bits. Thereupon, Section 5 extends our definition to messages of more general lengths. Section 6 provides the details of our security analysis.

We provide further insights on the starting point of our research in the Appendix. Therein, we also discuss attacks on insecure preliminary variants that motivated our studies towards the final design of ZCZ.

2 Preliminaries

GENERAL NOTATION. We use lowercase letters x for indices and integers, uppercase letters X, Y for binary strings and functions, and calligraphic uppercase letters $\mathcal{X}, \mathcal{Y}$ for sets. We denote the concatenation of binary strings X and Y by $X \| Y$; we mostly treat bit strings as representations of elements in the finite field $\mathbb{F}_{2^n}$, which is the Galois Field $\mathbb{GF}(2^n)$ with a fixed irreducible polynomial $p(\mathtt{x})$. There, we interpret a bit string $(x_{n-1} \ldots x_1 x_0)$ as polynomial $\sum_{i=0}^{n-1} a_i \cdot \mathtt{x}^i$ in $\mathbb{F}_{2^n}$. Bit x_i represents the coefficient $a_i \in \{0, 1\}$, for $0 \leq i \leq n-1$, and the most significant bit is the leftmost, and the least significant bit is the rightmost bit. We denote the result of the addition of two elements as $X + Y$, which is equivalent to the XOR of X and Y. For tuples of bit strings $(X_1, \ldots, X_x)$, $(Y_1, \ldots, Y_x)$ of equal domain, we denote by $(X_1, \ldots, X_x) + (Y_1, \ldots, Y_x)$ the element-wise XOR, i.e., $(X_1 + Y_1, \ldots, X_x + Y_x)$. Unless stated otherwise, we consider all additions

[3] By effective tweak size, we mean the usable tweak domain without bits that are used for other purposes such as domain separation.

of n-bit values to be in $\mathbb{F}_2^n$. Moreover, we will use $\oplus$ for the XOR of bit strings in illustrations. However, all additions and subtractions in sub- and superscripts that denote indices represent integer additions. We indicate the length of a bit string X in bits by $|X|$, and write X_i for the i-th block. Moreover, we denote by $X \twoheadleftarrow \mathcal{X}$ that X is chosen independently uniformly at random from the set $\mathcal{X}$. We define three sets of particular interest: $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ be the set of all functions $F : \mathcal{X} \to \mathcal{Y}$, $\mathsf{Perm}(\mathcal{X})$ the set of all permutations over $\mathcal{X}$, and $\widetilde{\mathsf{Perm}}(\mathcal{T}, \mathcal{X})$ for the set of tweaked permutations over $\mathcal{X}$ with associated tweak space $\mathcal{T}$.

$(X_1, \ldots, X_x) \xleftarrow{n} X$ denotes that X is split into the minimal number of n-bit blocks possible i.e., $X_1 \parallel \ldots \parallel X_x = X$, and $|X_i| = n$ for $1 \leq i \leq x - 1$, and $|X_x| \leq n$. So, when $|X| > 0$, then $|X_x| > 0$. If $|X| = 0$, $Y \xleftarrow{x} X$ sets Y to the empty string. $\langle X \rangle_n$ denotes an encoding of an integer $X \in \mathbb{Z}_n$ as an n-bit string. For two sets $\mathcal{X}$ and $\mathcal{Y}$, a uniform random function $\rho : \mathcal{X} \to \mathcal{Y}$ maps inputs $X \in \mathcal{X}$ independently and uniformly at random to outputs $Y \in \mathcal{Y}$. For an event E, we denote by $\Pr[E]$ the probability of E; ε is the empty string. For a given set $\mathcal{X}$ and integer x, we define $\mathcal{X}^{\leq x} = \bigcup_{i=1}^{x} \mathcal{X}^i$ and $\mathcal{X}^+ = \bigcup_{j=1}^{\infty} \mathcal{X}^j$. For two integers n, k with $n \geq k \geq 1$, we denote the falling factorial as $(n)_k = \prod_{i=0}^{k-1}(n - i)$.

Adversaries. An adversary $\mathbf{A}$ is an efficient Turing machine that interacts with a given set of oracles that appear as black boxes to $\mathbf{A}$. We denote by $\mathbf{A}^{\mathcal{O}}$ the output of $\mathbf{A}$ after interacting with some oracle $\mathcal{O}$. We write $\Delta_{\mathbf{A}}(\mathcal{O}^1; \mathcal{O}^2) := |\Pr[\mathbf{A}^{\mathcal{O}^1} \Rightarrow 1] - \Pr[\mathbf{A}^{\mathcal{O}^2} \Rightarrow 1]|$ for the advantage of $\mathbf{A}$ to distinguish between oracles $\mathcal{O}^1$ and $\mathcal{O}^2$. All probabilities are defined over the random coins of the oracles and those of $\mathbf{A}$, if any. W.l.o.g., we assume that $\mathbf{A}$ never asks queries to which it already knows the answer.

A block cipher E with associated key space $\mathcal{K}$ and message space $\mathcal{M}$ is a mapping $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ such that for every key $K \in \mathcal{K}$, it holds that $E(K, \cdot)$ is a permutation over $\mathcal{M}$. A tweakable block cipher $\widetilde{E}$ with additional tweak space $\mathcal{T}$ is a mapping $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ such that for every key $K \in \mathcal{K}$ and tweak $T \in \mathcal{T}$, it holds that $\widetilde{E}(K, T, \cdot)$ is a permutation over $\mathcal{M}$. We also write $\widetilde{E}_K^T(\cdot)$ as short form. In this work, we assume that SPRPs allow variable-length inputs, i.e., there is no single fixed length, but the length of the ciphertext always equals that of the plaintext and vice versa; moreover, over all inputs of equal length, the construction is a permutation. The advantage is defined as follows.

Definition 1 (SPRP Advantage). Let $\mathcal{K}$ be a non-empty set and $\mathcal{M} \subset \{0,1\}^*$. Let $\Pi : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be a length-preserving permutation. Let $\pi \twoheadleftarrow \mathsf{Perm}(\mathcal{M})$ be sampled from the set of all length-preserving permutations of $\mathcal{M}$, and $K \twoheadleftarrow \mathcal{K}$. Then, the SPRP advantage of $\mathbf{A}$ with respect to Π is defined as $\mathbf{Adv}_{\Pi}^{\mathrm{SPRP}}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \Delta_{\mathbf{A}}(\Pi_K, \Pi_K^{-1}; \pi, \pi^{-1})$.

Definition 2 (STPRP Advantage). Let $\mathcal{K}$ and $\mathcal{T}$ be non-empty sets and let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ denote a tweakable block cipher. Let $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}, \{0,1\}^n)$ and $K \twoheadleftarrow \mathcal{K}$. Then, the STPRP advantage of $\mathbf{A}$ w.r.t. $\widetilde{E}$ is defined as $\mathbf{Adv}_{\widetilde{E}}^{\mathrm{STPRP}}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \Delta_{\mathbf{A}}(\widetilde{E}_K, \widetilde{E}_K^{-1}; \widetilde{\pi}, \widetilde{\pi}^{-1})$.

Definition 3 (Almost-XOR-Universal Hash Function). Let $\mathcal{K}$, $\mathcal{X}$, and $\mathcal{Y} \subseteq \{0,1\}^*$ be non-empty sets. Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a function keyed by $K \in \mathcal{K}$. We call H ϵ-almost-XOR-universal (ϵ-AXU) if, for all distinct $X, X' \in \mathcal{X}$ and any $\Delta \in \mathcal{Y}$, it holds that $\Pr_{K \twoheadleftarrow \mathcal{K}}\left[H_K(X) - H_K(X') = \Delta\right] \leq \epsilon$, where subtraction is in $\mathbb{F}_{2^n}$.

The H-Coefficient Technique. The H-coefficient technique is a proof method by Patarin [47]. It assumes that the results of the interaction of an adversary $\mathbf{A}$ with its oracles are collected in a transcript τ of the attack: $\tau = \langle (M_1, C_1, d_1), \ldots, (M_q, C_q, d_q) \rangle$. (M_i, C_i) denotes the in- and output of the i-th query of $\mathbf{A}$; a Boolean variable d_i denotes the query direction: $d_i = 1$ indicates that C_i was result of an encryption query, and $d_i = 0$that M_i was the result of a decryption query. The task of $\mathbf{A}$ is to distinguish the real world $\mathcal{O}_{\text{real}}$ from the ideal world $\mathcal{O}_{\text{ideal}}$. A transcript τ is called *attainable* if the probability to obtain τ in the ideal world is non-zero. We denote by Θ_{real} and Θ_{ideal} the distribution of transcripts in the real and the ideal world, respectively. Then, the fundamental Lemma of the H-coefficients technique, whose proof is given in [14, 47], states:

Lemma 1 (Fundamental Lemma of the H-coefficient Technique [47]). Assume that the set of attainable transcripts is partitioned into two disjoint sets GoodT and BadT. Further assume that there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in$ GoodT, it holds that

$$\frac{\Pr\left[\Theta_{\text{real}} = \tau\right]}{\Pr\left[\Theta_{\text{ideal}} = \tau\right]} \geq 1 - \epsilon_1, \quad \text{and} \quad \Pr\left[\Theta_{\text{ideal}} \in \textsc{BadT}\right] \leq \epsilon_2.$$

Then, for all adversaries $\mathbf{A}$, it holds that $\Delta_{\mathbf{A}}(\mathcal{O}_{\text{real}}; \mathcal{O}_{\text{ideal}}) \leq \epsilon_1 + \epsilon_2$.

3 On the Minimal Number of Required Primitive Calls

This section shows that any PRP with fewer than $3\ell - 1$ calls for messages of 2ℓ blocks to a primitive with n-bit tweak size and n-bit state size is insecure. We follow the approach by [44], who proved that an SPRP based on a single-keyed classical block cipher needs at least 2ℓ calls to the primitive for ℓ-block messages.

3.1 Generic Construction

Define positive integers n, τ, and ℓ, and let $\mathcal{M} \subseteq \{0,1\}^*$ denote a space for which $(\{0,1\}^n)^{2\ell} \subseteq \mathcal{M}$. Let $r \leq 3\ell - 2$ and let $\widetilde{\pi}_i : \{0,1\}^\tau \times \{0,1\}^n \to \{0,1\}^n$, for all $1 \leq i \leq r$, denote tweakable permutations with tweak space $\{0,1\}^\tau$ and state size n. Let $\Pi[\widetilde{\pi}_1, \ldots, \widetilde{\pi}_r] : \mathcal{M} \to \mathcal{M}$ be a length-preserving cipher that employs as its only non-linear functions in total r calls to the permutations $\widetilde{\pi}_1, \ldots, \widetilde{\pi}_r$. For simplicity, we also write Π as short form, hereafter. All further components of Π are linear over $\mathbb{F}_{2^n}$. For any such construction, we can formulate this as follows. Let X_i denote the input to π_i, T_i the tweak to π_i, and let $Y_i \leftarrow \pi_i(X_i)$ denote its output. The linear operations in Π must be describable as non-zero

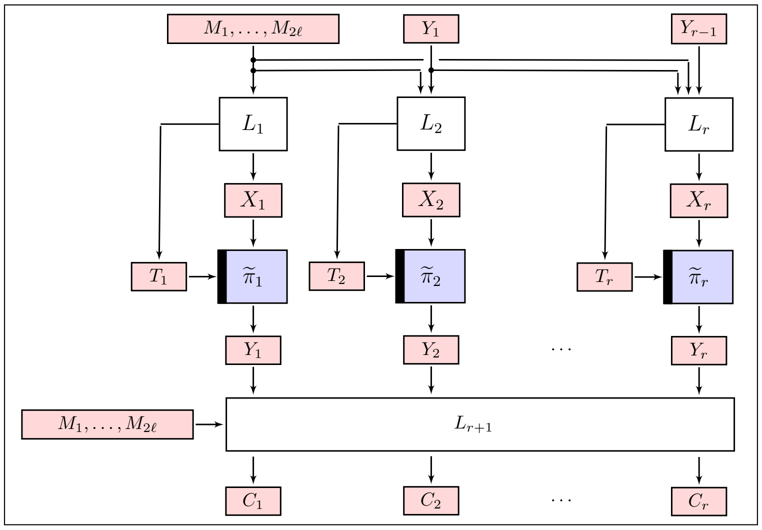


Fig. 1: Generic model of a PRP that consists of at most $r \le 3\ell - 2$ calls to tweakable block ciphers $\widetilde{\pi}_i$ for messages of 2ℓ blocks.

linear functions $L_i : \mathcal{M} \times (\{0,1\}^n)^{i-1} \to \{0,1\}^n \times \{0,1\}^\tau$, for $1 \le i \le r$, and an additional non-zero linear function $L_{r+1} : \mathcal{M} \times (\{0,1\}^n)^r \to \mathcal{M}$ that, for all given inputs $(M, Y_1, \dots, Y_r) \in \mathcal{M} \times (\{0,1\}^n)^r$, outputs C s.t. it holds that $|C| = |M|$. Then, we can describe the encryption with $\Pi(M)$ as

$$\begin{aligned} (X_i, T_i) &\leftarrow L_i\left(M, Y_1, \dots, Y_{i-1}\right), && \text{for all } 1 \le i \le r, \\ Y_i &\leftarrow \widetilde{\pi}^{T_i}(X_i), && \text{for all } 1 \le i \le r, \text{ and} \\ C &\leftarrow L_{r+1}(M, Y_1, \dots, Y_r). && \end{aligned}$$

Π must be correct for all inputs, i.e., for all $M, C \in \mathcal{M}$, it must hold that $\Pi^{-1}(\Pi(M)) = M$ and $\Pi(\Pi^{-1}(C)) = C$. Figure 1 gives an illustration.

Remark 1. It may not be instantaneously clear why the generic construction above covers all considered schemes. Note that it computes the values X_i and T_i by a non-zero linear function of M, Y_1, Y_2, ..., Y_{i-1}. So, the previous values Y_i can also be used to generate X_i. Indeed, it is generic enough to include all such constructions where the only non-linear components are the permutation calls.

For simplicity, we consider independent permutations with tweak domain $\mathbb{F}_2^\tau$ in this section. For efficiency, our proposal later in this work will employ only a single tweakable primitive with a composite tweak domain $\mathcal{T}_D = \mathcal{D} \times \mathbb{F}_2^\tau$, where $\mathcal{D}$ is a non-empty set of domains. So, this approach achieves the same goal of having independent permutations. We consider that τ is the effectively usable size of the tweaks without domains.

3.2 A PRP Attack on Constructions with At Most $3\ell - 2$ Calls

Case $\tau = n$. Let **A** be an adversary with the goal to distinguish the outputs of a variable-input-length PRP Π under a secret key as above from an ideal PRP. First, **A** chooses two messages M and M' of 2ℓ blocks each, i.e., $M = (M_1, \ldots, M_{2\ell})$ and $M' = (M'_1, \ldots, M'_{2\ell})$. We define the differences $\Delta M = M - M'$, and analogously the differences ΔX_i, ΔY_i, and ΔC in the obvious manner. Choose M and M' such that it holds that $\Delta X_i = 0$ and $\Delta T_i = 0$, for $1 \leq i \leq \ell - 1$. Note that such a choice of M and M' must be possible since these variables correspond to $2\ell - 2$ equations ($\ell - 1$ equations for adjusting the values ΔX_i and $\ell - 1$ equations for adjusting the values ΔT_i) and there exist 2ℓ blocks ΔM_i. For instance, the adversary can efficiently derive an element N from the null space of $L_1, \ldots, L_{2(\ell-1)}$. It chooses M arbitrarily and derives $M' = M + N$.

From $\Delta X_i = 0^n$ and $\Delta T_i = 0^\tau$ for $1 \leq i \leq \ell - 1$, it follows that $\Delta Y_i = \widetilde{\pi}^{T_i}(X_i) \oplus \widetilde{\pi}^{T'_i}(X'_i) = 0^n$, for all $1 \leq i \leq \ell - 1$. The non-linear layer of calls to the tweakable block cipher maps $(\Delta X_1, \ldots, \Delta X_r)$ to $(\Delta Y_1, \ldots, \Delta Y_r)$. We obtain

$$L_{r+1}(\Delta M, \underbrace{\Delta Y_1, \ldots, \Delta Y_{\ell-1}}_{=(0, \ldots, 0)}, \Delta Y_\ell, \ldots, \Delta Y_r) = \Delta C.$$

Since **A** fixed ΔM and chose M and M' so that $\Delta X_1 = \ldots = \Delta X_{\ell-1} = 0^n$ and $\Delta T_1 = \ldots = \Delta T_{\ell-1} = 0^\tau$, we obtain $\Delta Y_1, \ldots, \Delta Y_{\ell-1} = 0^n$. So, there are at most $2\ell - 1$ free variables $\Delta Y_\ell, \ldots \Delta Y_r$, and 2ℓ equations for $\Delta C_1, \ldots, \Delta C_{2\ell}$, which implies that 2ℓ blocks of ΔC are a linear combination of $2\ell - 1$ values $\Delta Y_\ell, \ldots, \Delta Y_r$. So, in the real construction, L_{r+1} defines a map from $2\ell - 1$ to 2ℓ n-bit variables, and **A** can efficiently derive a solution $\Delta Y_\ell, \ldots, \Delta Y_r$ from the null space of the equation system. This becomes a distinguishing event happening with probability one in the real construction and with probability $1/2^n$ in the ideal world for this example. The distinguishing advantage is hence $1 - 1/2^n$. **A** can query it with two messages as above and output real if such a non-zero linear function L exists and random otherwise, as summarized in Algorithm 1.

For general values of τ. A similar attack is applicable for general values of τ. Though, we have to consider linearity over $\mathbb{F}_2$ then. Define

$$s = \left\lfloor \frac{2\ell n}{n + \tau} \right\rfloor - 1.$$

The adversary chooses $M \in (\mathbb{F}_2^n)^{2\ell}$ arbitrarily, and $M' \in (\mathbb{F}_2^n)^{2\ell}$ with $M \neq M'$ s. t. $\Delta X_1 = \ldots \Delta X_s = 0^n$ and $\Delta T_1 = \ldots = \Delta T_s = 0^\tau$. Note that we consider the inputs $X_i \in \mathbb{F}_2^n$ and the tweaks $T_i \in \mathbb{F}_2^\tau$ as blocks. Again, such a choice of M' exists for the same reason as above and can be found efficiently from the null space of the linear functions $L_1, L_2, \ldots$ that are involved in the computation of $\Delta X_1, \ldots, \Delta X_s$ and $\Delta T_1, \ldots, \Delta T_s$. Again, we obtain $\Delta Y_i = 0^n$, for $1 \leq i \leq s$ for the real construction. We obtain the equation

$$L_{r+1}(\Delta M, \underbrace{\Delta Y_1, \ldots, \Delta Y_s}_{=(0, \ldots, 0)}, \Delta Y_{s+1}, \ldots, \Delta Y_r) = \Delta C.$$

Algorithm 1 PRP attack on generic constructions Π with at most $3\ell - 2$ primitive calls, here for $\tau = n$.

```
1: function A^Π
2:     Choose $M_i$ for $1 \leq i \leq 2\ell$ arbitrarily
3:     Choose $M'_i$ for $\ell \leq i \leq 2\ell$ s. t. it holds that
4:       $L_i(\Delta M_i) = (\Delta X_i, \Delta T_i) = (0^n, 0^\tau)$, for $1 \leq i \leq 2(\ell - 1)$
5:     Ask for the encryption of $C = \Pi(M)$ and $C' = \Pi(M')$
6:     Derive $\Delta C = C' - C$
7:     if there exists $(\Delta Y_\ell, \ldots, \Delta Y_r)$, s. t. $L_{r+1}(\Delta M, \Delta Y) = \Delta C$ then
8:         return "Real"
9:     return "Random"
```

The blocks $\Delta Y_{s+1}, \ldots, \Delta Y_r$ contain $(r-s)n$ bits, that are mapped through L_{r+1} to $\Delta C_{2\ell n}$ bits. For all schemes Π that use r calls to the primitive with

$$(r - s) \cdot n < 2\ell n, \qquad \text{which leads to} \qquad r < 2\ell \left(1 + \frac{n}{n + \tau}\right) - 1,$$

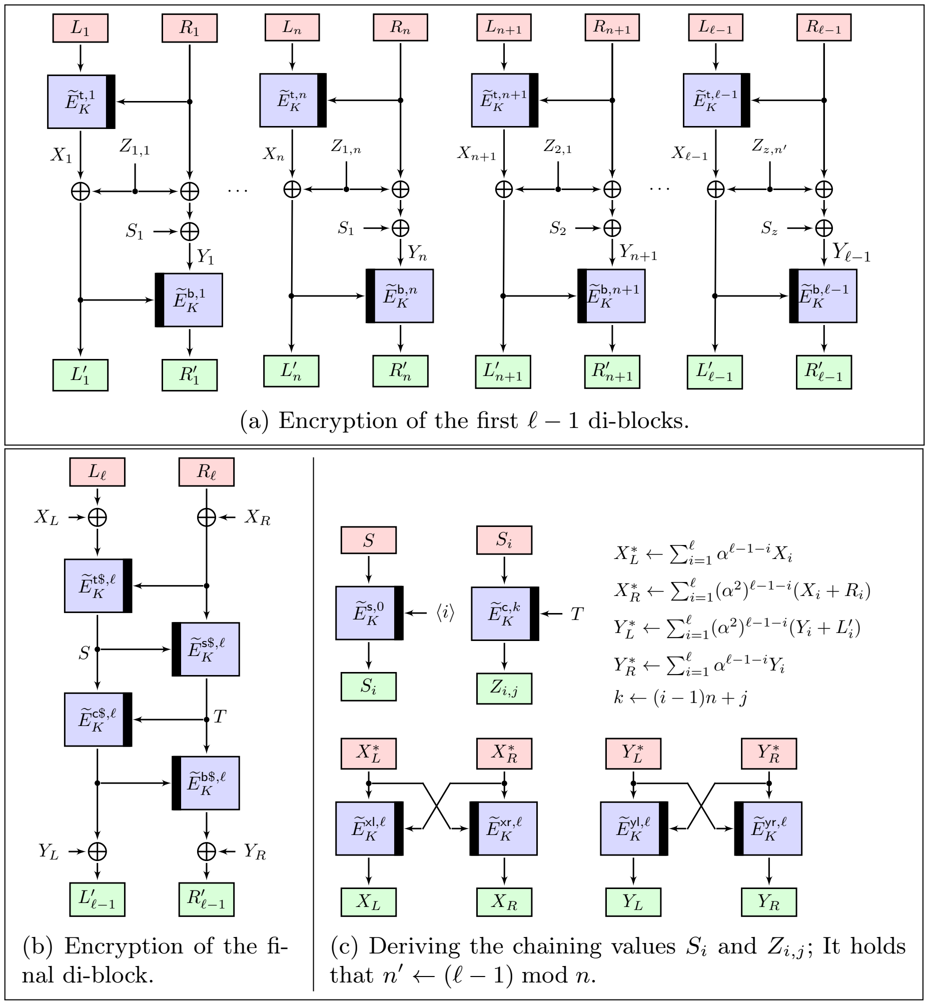
we obtain a compressing mapping. Then, there exist are more equations than variables, and the distinguisher as before applies. However, the advantage may be smaller and depends on the values of r, n, and τ.

4 Definition of The Basic ZCZ Construction

This section defines the basic ZCZ scheme. First, we consider messages that consist of at most $2n$ blocks, and will extend it thereupon to all messages whose length is a multiple of $2n$ bits. The subsequent section will then further define it for messages whose lengths are not necessarily multiples of $2n$ bits.

Parameters. Let $n, \tau, k, d \geq 1$ be integers with $d \ll n$ and $n = \tau$; we define $N \overset{\text{def}}{=} 2^n$ as an alias. Let $\mathcal{B} = \{0,1\}^{2n}$ define a *di-block* (or dual block, double block), i.e., $2n$ bits. We define non-empty sets of tweaks $\mathcal{T} = \{0,1\}^\tau$, keys $\mathcal{K} = \{0,1\}^k$, domains $\mathcal{D} = \{\mathsf{t}, \mathsf{s}, \mathsf{c}, \mathsf{b}, \mathsf{t\$}, \mathsf{s\$}, \mathsf{c\$}, \mathsf{b\$}, \mathsf{xl}, \mathsf{xr}, \mathsf{yl}, \mathsf{yr}, \mathsf{p}, \mathsf{kd}\} \subseteq \{0,1\}^d$, and a set of indices $\mathcal{I} \subseteq \{1, \ldots, 2^n - 1\}$. The purpose of domains and indices is to define an extended tweak set $\mathcal{T}_{D,I} = \mathcal{D} \times \mathcal{I} \times \mathcal{T}$ for a tweakable block cipher $\widetilde{E} : \mathcal{K} \times \mathcal{T}_{D,I} \times \{0,1\}^n \to \{0,1\}^n$.

Overview. The basic ZCZ$[\widetilde{E}_K]$ construction takes as input a secret key $K \in \mathcal{K}$ and a plaintext $M \in \mathcal{B}^{\leq n}$ that is split into $\ell \in [1..n]$ di-blocks. The design can be split into a top, middle, and a bottom layer. In the top layer, the first $\ell - 1$ complete di-blocks (L_i, R_i) are processed similarly as in the $\mathbb{ZHASH}$ construction by Iwata et al. [28]. The TBC outputs X_i are accumulated by an MDS code to two values X^*_L and X^*_R using the Horner rule, which are finally encrypted in a butterfly-like structure [41] to $X_L \leftarrow \widetilde{E}_K^{\mathsf{xl}, \ell, X^*_R}(X^*_L)$ and $X_R \leftarrow \widetilde{E}_K^{\mathsf{xr}, \ell, X^*_L}(X^*_R)$. X_L


L_1 R_1 L_n R_n L_{n+1} R_{n+1} $L_{\ell-1}$ $R_{\ell-1}$
$\widetilde{E}_K^{\mathsf{t},1}$ $\widetilde{E}_K^{\mathsf{t},n}$ $\widetilde{E}_K^{\mathsf{t},n+1}$ $\widetilde{E}_K^{\mathsf{t},\ell-1}$
X_1 $Z_{1,1}$ X_n $Z_{1,n}$ X_{n+1} $Z_{2,1}$ $X_{\ell-1}$ $Z_{z,n'}$
S_1 S_1 S_2 S_z
Y_1 Y_n Y_{n+1} $Y_{\ell-1}$
$\widetilde{E}_K^{\mathsf{b},1}$ $\widetilde{E}_K^{\mathsf{b},n}$ $\widetilde{E}_K^{\mathsf{b},n+1}$ $\widetilde{E}_K^{\mathsf{b},\ell-1}$
L'_1 R'_1 L'_n R'_n L'_{n+1} R'_{n+1} $L'_{\ell-1}$ $R'_{\ell-1}$
L_ℓ R_ℓ X_L X_R
$\widetilde{E}_K^{\mathsf{t\$},\ell}$ S $\widetilde{E}_K^{\mathsf{s\$},\ell}$ $\widetilde{E}_K^{\mathsf{c\$},\ell}$ T $\widetilde{E}_K^{\mathsf{b\$},\ell}$
Y_L Y_R $L'_{\ell-1}$ $R'_{\ell-1}$
S $\widetilde{E}_K^{\mathsf{s},0}$ $\langle i \rangle$ S_i
S_i $\widetilde{E}_K^{\mathsf{c},k}$ T $Z_{i,j}$
$X_L^* \leftarrow \sum_{i=1}^{\ell} \alpha^{\ell-1-i} X_i$
$X_R^* \leftarrow \sum_{i=1}^{\ell} (\alpha^2)^{\ell-1-i} (X_i + R_i)$
$Y_L^* \leftarrow \sum_{i=1}^{\ell} (\alpha^2)^{\ell-1-i} (Y_i + L'_i)$
$Y_R^* \leftarrow \sum_{i=1}^{\ell} \alpha^{\ell-1-i} Y_i$
$k \leftarrow (i-1)n + j$
X_L^* X_R^* $\widetilde{E}_K^{\mathsf{xl},\ell}$ $\widetilde{E}_K^{\mathsf{xr},\ell}$ X_L X_R
Y_L^* Y_R^* $\widetilde{E}_K^{\mathsf{yl},\ell}$ $\widetilde{E}_K^{\mathsf{yr},\ell}$ Y_L Y_R

(a) Encryption of the first $\ell - 1$ di-blocks.

(b) Encryption of the final di-block.

(c) Deriving the chaining values S_i and $Z_{i,j}$; It holds that $n' \leftarrow (\ell - 1) \bmod n$.

Fig. 2: Encryption of a message with ℓ complete di-blocks with $\mathrm{ZCZ}[\widetilde{E}_K]$.

and X_R are used to mask the branches of the final di-block, L_ℓ and R_ℓ. The final di-block is processed by a four-round Feistel-like network of four TBC calls in the spirit of the constructions by Coron et al. [17].

This four-round network generates two intermediate values S and T after the first and second call to $\widetilde{E}$. The middle layer derives from S and T a value $S_1 \leftarrow \widetilde{E}_K^{\mathsf{s},0,1}(S)$ and a series of $\ell - 1$ chaining values $Z_{1,j} \leftarrow \widetilde{E}^{\mathsf{c},j,T}(S_1)$. For the j-th di-block, the chaining value $Z_{1,j}$ is added to both branches of the j-th block. Moreover, S_1 is also added to the right branch of each di-block: $L'_j \leftarrow X_j + Z_{i,j}$ and $Y_j \leftarrow R_j + Z_{1,j} + S_1$. So, this middle layer ensures that each di-block depends on all others. Finally, the middle layer generates from the blocks Y_j and L'_j two values Y_L and Y_R symmetrically as X_L and X_R, from the values Y_j.

The bottom layer is then a symmetric version of the top layer. The $\ell - 1$ di-blocks are processed by another $\mathbb{ZHASH}$ layer to compute the ciphertext blocks: $L'_j \leftarrow X_j$ and $R'_j \leftarrow \widetilde{E}_K^{\mathsf{b},i,L'_j}(Y_j)$. The final complete di-block is processed by two further Feistel rounds before Y_L added to the left branch, and Y_R is added to the right branch of the ℓ-th di-block. The resulting values L'_i, R'_i, for $1 \leq i \leq \ell$, are concatenated and returned as the ciphertext. The details of the encryption with $\mathrm{ZCZ}[\widetilde{E}_K]$ is given in Algorithm 2, and is illustrated in parts in Figure 2, already for more than n complete di-blocks.

Rationale. The structure is inspired by ZHash [28] and AEZ [25]. The use of α and α^2 prevents that a collision in X_L would automatically lead to a collision also in X_R and vice versa; considering also the tweak values R_i for X_R renders birthday collisions in X_i from separate tweaks ineffective. Encrypting X_L^*, X_R^*, Y_L^*, and Y_R^* avoids that differences in the masks cancel differences in the final di-block. Finally, adding S_i and $Z_{i,j}$ prevents adversaries from observing differences $\Delta Z_{1,j}$. Using the masks X_L, X_R, Y_L, and Y_R in the final block makes its outputs depend on all blocks; Using S and T for the counter mode in the middle layer creates a dependency of each di-block on all others. We elaborate on attacks on preliminary versions of ZCZ in Appendix D. We employ pairwise distinct domains for all calls to $\widetilde{E}$ to prevent dependencies between the calls.

Extension to Longer Messages. Messages with more than n di-blocks are partitioned into *chunks*. The i-th (complete) chunk denotes the series of the n consecutive di-blocks $(L_{(i-1)n+1}, R_{(i-1)n+1}, \ldots L_{i \cdot n}, R_{i \cdot n})$, and employs the chaining values S_i and $Z_{i,j}$. We derive all chaining values under distinct domains as before. Furthermore, we derive $\ell - 1$ chaining values $Z_{i,j}$ by a TBC call each from S. For the i-th chunk, S_i is computed as $S_i \leftarrow \widetilde{E}_K^{\mathsf{s},0,i}(S)$. Then, for $j \in [1..n]$, $Z_{i,j}$ for the j-th block of the i-th chunk is generated as $Z_{i,j} \leftarrow \widetilde{E}_K^{\mathsf{c},0,n(i-1)+j}(S_i)$. $Y_{n(i-1)+j}$ is then computed as $Y_{n(i-1)+j} \leftarrow R_{n(i-1)+j} + S_i + Z_{n(i-1)+j}$. The rest of the computations remain unchanged. Letting j take any value in $[1..\ell]$, we can rewrite this as

$$Y_j \leftarrow R_j + S_{\lceil j/n \rceil} + Z_j. \tag{2'}$$

The encryption of $\mathrm{ZCZ}[\widetilde{E}_K]$ is defined in Algorithm 2, and illustrated in parts in Figure 2, already for more than n complete di-blocks. The figure employs bold bars in the blocks of $\widetilde{E}$ to indicate the parts of the tweaks that stem from $\mathcal{T}$. The decryption is defined in the obvious way.

5 ZCZ* for Messages with Partial Final Di-block

We extend the definition of ZCZ to messages whose length is not a multiple of $2n$ bits. We denote the last $r \leftarrow |M| \bmod 2n$ bits as *partial di-block*. Our approach for ZCZ* is inspired by the DE domain extender from [43]. Therefore, we briefly recap it.

Algorithm 2 Definition of the encryption algorithm of $\mathrm{ZCZ}[\widetilde{E}]$ given a tweakable block cipher $\widetilde{E}$. The code in the boxes is only part of $\mathrm{ZCZ}^*[\widetilde{E}]$ in Algorithm 3.

```
10: function $\mathrm{ZCZ}[\widetilde{E}_K](M)$
11:     $r \leftarrow |M| \bmod 2n$
12:     $\ell \leftarrow (|M| - r)/2n$
13:     $z \leftarrow \lceil (\ell - 1)/n \rceil$
14:     $L'_* \leftarrow \varepsilon$; $R'_* \leftarrow \varepsilon$
15:     $\textsc{Parse}(M, \ell)$
16:     $\textsc{TopEnc}[\widetilde{E}_K]()$
17:     if $r > 0$ then
18:         $\boxed{\textsc{PartialTopEnc}[\widetilde{E}_K]()}$
19:     $\textsc{LastTopEnc}[\widetilde{E}_K](X_L, X_R)$
20:     $\textsc{MidLayer}[\widetilde{E}_K](S, T)$
21:     $\textsc{BotEnc}[\widetilde{E}_K]()$
22:     $\textsc{LastBotEnc}[\widetilde{E}_K](Y_L, Y_R)$
23:     if $r > 0$ then
24:         $\boxed{\textsc{PartialBotEnc}[\widetilde{E}_K]()}$
25:     $C \leftarrow (L'_1 \| R'_1 \| \cdots \| L'_\ell \| R'_\ell \| L'_* \| R'_*)$
26:     return $C$

30: procedure $\textsc{TopEnc}[\widetilde{E}_K]$
31:     $X^*_L \leftarrow X^*_R \leftarrow 0^n$
32:     for $i \leftarrow 1 \ldots \ell - 1$ do
33:         $X_i \leftarrow \widetilde{E}_K^{\mathsf{t},i,R_i}(L_i)$
34:         $X^*_L \leftarrow X^*_L + \alpha^{\ell-1-i} X_i$
35:         $X^*_R \leftarrow X^*_R + (\alpha^2)^{\ell-1-i}(X_i + R_i)$
36:     $X_L \leftarrow \widetilde{E}_K^{\mathsf{xl},\ell,X^*_R}(X^*_L)$
37:     $X_R \leftarrow \widetilde{E}_K^{\mathsf{xr},\ell,X^*_L}(X^*_R)$

40: procedure $\textsc{MidLayer}[\widetilde{E}_K](S, T)$
41:     $S_0 \leftarrow S$
42:     for $i \leftarrow 1 \ldots z$ do
43:         $S_i \leftarrow \widetilde{E}_K^{\mathsf{s},0,i}(S_{i-1})$
44:     for $i \leftarrow 1 \ldots z$ do
45:         for $j \leftarrow 1 \ldots n$ do
46:             $Z_{i,j} \leftarrow \widetilde{E}_K^{\mathsf{c},(i-1)n+j,T}(S_i)$

50: procedure $\textsc{LastTopEnc}[\widetilde{E}_K](X_L, X_R)$
51:     $S \leftarrow \widetilde{E}_K^{\mathsf{t\$},\ell,R_\ell+X_R}(L_\ell + X_L)$
52:     $T \leftarrow \widetilde{E}_K^{\mathsf{s\$},\ell,S}(R_\ell + X_R)$

60: procedure $\textsc{BotEnc}[\widetilde{E}_K]$
61:     $Y^*_L \leftarrow 0^n$
62:     $Y^*_R \leftarrow 0^n$
63:     for $i \leftarrow 1 \ldots z - 1$ do
64:         for $j \leftarrow 1 \ldots n$ do
65:             $k \leftarrow (i-1)n + j$
66:             $L'_k \leftarrow X_k + Z_{i,j}$
67:             $Y_k \leftarrow R_k + Z_{i,j} + S_i$
68:             $R'_k \leftarrow \widetilde{E}_K^{\mathsf{b},k,L'_k}(Y_k)$
69:             $Y^*_L \leftarrow Y^*_L + (\alpha^2)^{\ell-1-k}(Y_k + L'_k)$
70:             $Y^*_R \leftarrow Y^*_R + (\alpha)^{\ell-1-k} Y_k$
71:     for $j \leftarrow 1 \ldots \ell - 1 - (z-1)n$ do
72:         $k \leftarrow (z-1)n + j$
73:         $L'_k \leftarrow X_k + Z_{z,j}$
74:         $Y_k \leftarrow R_k + Z_{z,j} + S_z$
75:         $R'_k \leftarrow \widetilde{E}_K^{\mathsf{b},k,L'_k}(Y_k)$
76:         $Y^*_L \leftarrow Y^*_L + (\alpha^2)^{\ell-1-k}(Y_k + L'_k)$
77:         $Y^*_R \leftarrow Y^*_R + \alpha^{\ell-1-k} Y_k$
78:     $Y_L \leftarrow \widetilde{E}_K^{\mathsf{yl},\ell,Y^*_R}(Y^*_L)$
79:     $Y_R \leftarrow \widetilde{E}_K^{\mathsf{yr},\ell,Y^*_L}(Y^*_R)$

80: procedure $\textsc{LastBotEnc}[\widetilde{E}_K](Y_L, Y_R)$
81:     $L'_\ell \leftarrow \widetilde{E}_K^{\mathsf{c\$},\ell,T}(S) + Y_L$
82:     $R'_\ell \leftarrow \widetilde{E}_K^{\mathsf{b\$},\ell,T}(L'_\ell + Y_L) + Y_R$

90: procedure $\textsc{Parse}(M, \ell)$
91:     $i \leftarrow \ell \cdot 2n$
92:     $(L_1, R_1, \cdots, L_\ell, R_\ell) \xleftarrow{n} M[0..i-1]$
93:     if $r > 0$ then
94:         $(L_*, R_*) \xleftarrow{n} M[i..|M|]$
```

The Domain Extender $\mathrm{DE}[\Pi, F, H] : \{0,1\}^{\geq n} \to \{0,1\}^{\geq n}$ [43] takes a block-wise-operating length-preserving permutation $\Pi : (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$, a PRF $F : \{0,1\}^n \to \{0,1\}^n$, and an XOR-universal hash function $H : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^n$. It produces a length-preserving permutation over bit strings of any length $\geq n$ bits. A message $M \in \{0,1\}^{\geq n}$ is split into blocks $(M_1, \ldots, M_\ell)$; $\mathrm{DE}[\Pi, F, H]$ computes the corresponding ciphertext $C = (C_1, \ldots, C_\ell)$ as: (1) $M^*_{\ell-1} \leftarrow H(M_{\ell-1}, M_\ell)$, (2) $(C_1, \ldots, C_{\ell-2}, C^*_{\ell-1}) \leftarrow \Pi(M_1, \ldots, M_{\ell-2}, M^*_{\ell-1})$, (3) $C_\ell \leftarrow F(M^*_{\ell-1} + C^*_{\ell-1}) +_{|M_\ell|} M_\ell$, and (4) $C_{\ell-1} \leftarrow H(C^*_{\ell-1}, C_\ell)$. where

$$x +_n y \overset{\text{def}}{=} \mathsf{msb}_n(x) + y$$

Algorithm 3 Functions of the encryption algorithm of ZCZ$^*[\widetilde{E}]$ for messages whose length is not necessarily a multiple of $2n$ bit (but at least $2n$ bit). Recall that $r = |M| \bmod 2n$.

```
10: procedure PartialTopEnc[$\widetilde{E}_K$]
11:     $M_\ell \leftarrow L_\ell \,\|\, R_\ell$
12:     $M_* \leftarrow \mathsf{pad}_{2n}(L_* \,\|\, R_*)$
13:     $(\overline{L}_*, \overline{R}_*) \xleftarrow{n} M_*$
14:     $(U_\ell, V_\ell) \leftarrow \mathcal{H}[\widetilde{E}_K, 0](\overline{L}_*, \overline{R}_*)$
15:     $L_\ell \leftarrow L_\ell + U_\ell$
16:     $R_\ell \leftarrow R_\ell + V_\ell$

20: function $\mathsf{msb}_x(X)$
21:     return $X[0..x-1]$

30: function $\mathsf{pad}_x(X)$
31:     return $X \,\|\, 1 \,\|\, 0^{x-|X|-1}$

40: procedure PartialBotEnc[$\widetilde{E}_K$]
41:     $(P, Q) \leftarrow \mathcal{H}[\widetilde{E}_K, 2](L_\ell + L'_\ell, R_\ell + R'_\ell)$
42:     $W \leftarrow \mathsf{msb}_r(P \,\|\, Q) \,\|\, 0^{2n-r}$
43:     $(P_*, Q_*) \xleftarrow{n} W$
44:     $L'_* \leftarrow L_* + P_*$
45:     $R'_* \leftarrow R_* + Q_*$
46:     $(\overline{L}'_*, \overline{R}'_*) \xleftarrow{n} \mathsf{pad}_{2n}(L'_* \,\|\, R'_*)$
47:     $(U'_\ell, V'_\ell) \leftarrow \mathcal{H}[\widetilde{E}_K, 4](\overline{L}'_*, \overline{R}'_*)$
48:     $L'_\ell \leftarrow L'_\ell + U'_\ell$
49:     $R'_\ell \leftarrow R'_\ell + V'_\ell$

50: function $\mathcal{H}[\widetilde{E}_K, i](U, V)$
51:     $U' \leftarrow \widetilde{E}_K^{\mathsf{p},i,V}(U)$
52:     $V' \leftarrow \widetilde{E}_K^{\mathsf{p},i+1,V}(U)$
53:     return $(U', V')$
```

for any $x, y \in \{0,1\}^*$ and integer n. To obtain that DE is a permutation, the hash function H must satisfy $H(H(M_{\ell-1}, M_\ell), M_\ell) = M_{\ell-1}$ for any allowed input $M_{\ell-1}$, M_ℓ (see [43, Remark 2]).

Overview of ZCZ*. Our extension ZCZ* requires that the message length is still at least $2n$ bits. Let $M_* = (L_*, R_*)$ be the partial message di-block that follows after ℓ complete di-blocks. Further assume that the partial di-block consists of $\geq n$ bits that are split into $|L_*| = n$ and $|R_*| < n$. The right part is padded to n bits by a single 1 and as many zero bits as necessary to extend it to n bits: $\overline{R}_* \leftarrow \mathsf{pad}_n(R_*)$. The values are given as inputs to a hash function $\mathcal{H}[\widetilde{E}_K, i]$, with $i = 0$, that is illustrated on the right side of Figure 3. $\{H\}$ uses one of the two n-bit values as state and the other one as tweak input for two calls to $\widetilde{E}_K$ under distinct tweaks: $U' \leftarrow \widetilde{E}_K^{\mathsf{p},i,V}(U)$ and $V' \leftarrow \widetilde{E}_K^{\mathsf{p},i+1,V}(U)$. The $2n$-bit output (U', V') is added to the final complete di-block. The resulting final di-block (L_ℓ, R_ℓ) is then processed by ZCZ$[\widetilde{E}_K]$. The sum of $(L_\ell, R_\ell) + (L'_\ell, R'_\ell)$ is then given again into $\mathcal{H}[\widetilde{E}_K, i]$, with $i = 2$ to produce a $2n$-bit value (P'_ℓ, Q'_ℓ). The most significant r bits of it are added to the final partial di-block to obtain the partial ciphertext di-block M'_*. M'_* is again padded to $2n$ bits and given as input to a third call to $\mathcal{H}[\widetilde{E}_K, i]$, with $i = 4$. The hash output is added to the final ciphertext di-block to produce M'_ℓ. If the partial di-block consists of less than n bits, it is also padded to $2n$ bits and processed analogously. So, the hash function H from the original definition of DE$[\mathit{\Pi}, F, H]$ is given by $H(M_\ell, M_*) \overset{\text{def}}{=} M_\ell + \mathcal{H}[\widetilde{E}_K, i](\mathsf{pad}_{2n}(M_*))$. One can see that the requirement from above holds for arbitrary M_ℓ and M_*: $H\left(H\left(M_\ell, M_*\right), M_*\right) = M_\ell$.

Remark 2. Note that ZCZ* still requires messages to consist of at least $2n$ bits. A further minor improvement in future work could be the integration of smaller

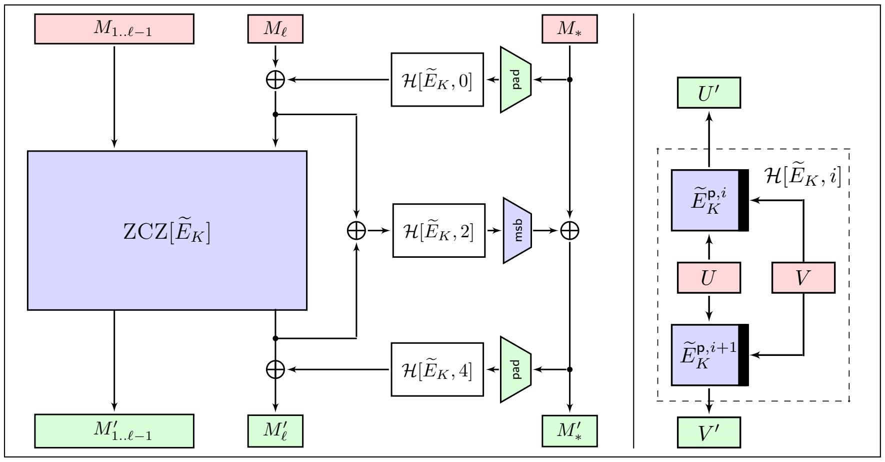

$M_{1..\ell-1}$
M_ℓ
M_*
$\mathcal{H}[\widetilde{E}_K, 0]$
pad
$\mathrm{ZCZ}[\widetilde{E}_K]$
$\mathcal{H}[\widetilde{E}_K, 2]$
msb
$\mathcal{H}[\widetilde{E}_K, 4]$
pad
$M'_{1..\ell-1}$
M'_ℓ
M'_*
U'
$\mathcal{H}[\widetilde{E}_K, i]$
$\widetilde{E}_K^{\mathsf{p},i}$
U
V
$\widetilde{E}_K^{\mathsf{p},i+1}$
V'

Fig. 3: Encryption of a partial message $M_1, \ldots, M_\ell, M^*$ whose length is not a multiple of $2n$ bit with $\mathrm{ZCZ}^*[\widetilde{E}_K]$. All preceeding di-blocks $M_1, \ldots, M_\ell$ are processed with $\mathrm{ZCZ}[\widetilde{E}_K]$ as before.

messages. For instance, the use of the very recent length-doubling construction LDT [15] could reduce the minimal message length to $n + 1$ bits. Though, this step would require an appropriate integration and ZCZ* is already a variable-input-length SPRP for lengths $\geq 2n$ bit.

6 Security Analysis of ZCZ and ZCZ*

This section studies the SPRP security of ZCZ and ZCZ*. Figure 4 provides a high-level overview on ZCZ. A given message M is split an input message into (M_L, M_R), where M_R consists of one $2n$-bit di-block, and M_L of the remaining di-blocks; the major part M_L is then processed by a variant of ZHash, that is denoted ZHash* here. It differs from ZHash in two aspects: ZHash* omits the XOR of the TBC output to the tweak input blocks. More prominently, ZHash* does not compress the input to two hash values, but is a permutation over $(n + \tau)^*$. So, the top layer returns the TBC outputs and the tweaks. $\widetilde{V}_1$ and $\widetilde{V}_2$ represent tweakable permutations. Internally, they can use the same primitive as also for ZHash*, and the tweakable variant of Counter mode, CTR*. H symbolizes an error-correcting code that sums up the inputs to $2n$ bits.

This high-level view allows to give a rationale for a dedicated analysis. A straight-forward use of a rate-1 counter mode would allow to apply a standard generic proof as for HCTR. Though, such an approach would yield 2ℓ calls to the primitive alone in the counter mode. In combination with ZHash*, this approach would need 4ℓ calls to the primitive for messages of 2ℓ blocks. ZCZ considers a special variant of counter mode that uses only ℓ blocks of entropy to mask 2ℓ blocks, similar as has been used in AEZ from version 2 [25]. However, this counter mode disallows to simply adopt the analysis from HCTR-like constructions when

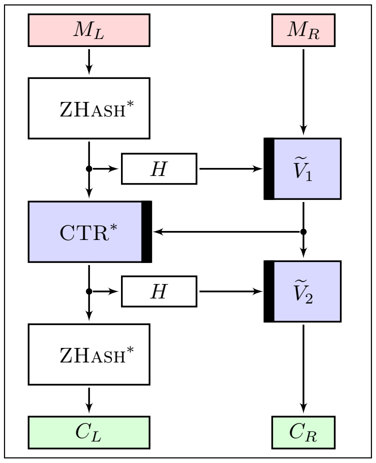


Fig. 4: High-level view on our proposal of ZCZ.

the goal is showing n-bit security. So, a dedicated analysis is needed, which is a major contribution of the present work. In the following, we study the security of the basic construction before we consider the extensions for inputs whose length is not necessarily a multiple of $2n$ bits, but at least $2n$ bits. We show the security of the extension ZCZ* at the end of this section.

6.1 Security of The Basic Construction

Theorem 1. Let $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}_{D,I}, \{0,1\}^n)$. Let $\mathbf{A}$ be an SPRP adversary on $\mathrm{ZCZ}[\widetilde{\pi}]$, s.t. $\mathbf{A}$ asks at most q queries of domain $\mathcal{B}^{\leq n}$, that sum up to at most σ di-blocks in total. Then

$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathrm{ZCZ}[\widetilde{\pi}]}(\mathbf{A}) \leq \frac{3\sigma^2 + 9q^2}{2N^2}.$$

Proof. The queries 1 through q by $\mathbf{A}$ are collected in a transcript τ where we define two disjoint sets of indices E and D s.t. $[1..q] = E \sqcup D$, and it holds that E consists of exactly those indices i s.t. the i-th query of $\mathbf{A}$ is an encryption query; similarly, D consists of exactly those indices i s.t. the i-th query of $\mathbf{A}$ is an decryption query. We define ℓ^i to be the number of di-blocks in the i-query, where $\ell^i \leq n$.

In both worlds, the adversary's queries are answered immediately with the corresponding outputs; certain internal parts of the transcript will be revealed to the adversary after it made all its queries, but before it outputs its decision bit that represents its guess of which world it interacted with. The internal parts consist of $S^i, T^i, S^i_1, X^i_L, X^i_R, Y^i_L, Y^i_R$ for $i \in [1..q]$ and $Z^i_{1,j}$ for $i \in [1..q], j \in [1..\ell^i - 1]$; for ease of notation, we write Z^i_j to refer to $Z^i_{1,j}$.

We will subsequently define certain transcripts to be *good*. More specifically, we describe a mechanism for the ideal oracle to sample the internal values to be given to the adversary at the end of the query phase, and define the event bad as the union of five events badA, badB, badC, badD and badE. We call a transcript good if it can be obtained by the ideal oracle without encountering the event bad. Now we state two lemmas.

Lemma 2. $\Pr[\mathsf{bad}] \leq \dfrac{3\sigma^2 + 8q^2}{N^2}$.

Lemma 3. For any good transcript τ,

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \geq 1 - \frac{q^2}{N^2}.$$

Then, the proof follows from Lemmas 1, 2, and 3. □

For proving Lemmas 2 and 3, we first define the sampling mechanism of the ideal oracle and the bad events.

Equations. First, we write the internal variables X_j^i, Y_j^i for $i \in [1..q], j \in [1..\ell^i]$ and $U_L^i, U_R^i, V_L^i, V_R^i$ for $i \in [1..q]$ in terms of $S^i, T^i, S_1^i, X_L^i, X_R^i, Y_L^i, Y_R^i, Z_j^i$:

$$X_j^i = L_j'^i + Z_j^i, \tag{1}$$

$$Y_j^i = R_j^i + Z_j^i + S_1^i, . \tag{2}$$

Moreover, we define four auxiliary variables to easier referral:

$$U_L^i = L_\ell^i + X_L^i, \tag{3}$$

$$U_R^i = R_\ell^i + X_R^i, \tag{4}$$

$$V_L^i = L_\ell'^i + Y_L^i, \tag{5}$$

$$V_R^i = R_\ell'^i + Y_R^i. \tag{6}$$

Identifying A Basis. A basis is the set of variables (internal to the constructions) which can be sampled uniformly and independently in the ideal oracles after fixing the inputs and outputs that are known to adversary. By looking at the construction and eliminating the relationships between the internal variables, plaintexts, and ciphertexts, some internal variables can be chosen almost freely, and still the real construction will behave indistinguishable from the ideal world for the adversary even after observing the plain- and ciphertexts. We call those variables a basis. For $i \in [1..q], j \in [1..\ell^i]$, we define (i, j) to be *fresh* if either of the following is true:

– $i \in E$, and for any $i' \in [1..i-1]$: $(L_j^{i'}, R_j^{i'}) \neq (L_j^i, R_j^i)$;
– $i \in D$, and for any $i' \in [1..i-1]$: $(L_j'^{i'}, R_j'^{i'}) \neq (L_j'^i, R_j'^i)$.

For $i \in [2..q], i' \in [1..i-1]$, we say i is *akin* to i' if either of the following holds:

– $\ell^i = \ell^{i'}$, $i \in E$, and for any $j \in [1..\ell^i - 1]$: $(L_j^{i'}, R_j^{i'}) = (L_j^i, R_j^i)$;

– $\ell^i = \ell^{i'}$, $i \in D$, and for any $j \in [1..\ell^i - 1]$: $(L'^{i'}_j, R'^{i'}_j) = (L'^i_j, R'^i_j)$;

We say i is *new* if it is not akin to any $i' \in [1..i-1]$. Now we define the basis as follows: for $i \in [1..q]$,

– For $j \in [1..\ell^i - 1]$, Z^i_j is in the basis if (i,j) is fresh;
– X^i_L and X^i_R are in the basis if $i \in D$, or if $i \in E$ and i is new;
– Y^i_L and Y^i_R are in the basis if $i \in E$, or if $i \in D$ and i is new;
– S^i, T^i, and S^i_1 are in the basis.

Let σ_F represent the total number of fresh pairs in the set $\{(i,j) \mid i \in [q], j \in [\ell^i - 1]\}$. Moreover, let q_ν be the total number of new queries in $[1..q]$. Then, the size of the basis is $\sigma_F + 2q_\nu + 5q$.

Extension from Basis. Now we show how all the internal variables X^i_j, Y^i_j for $i \in [1..q], j \in [1..\ell^i]$ and $U^i_L, U^i_R, V^i_L, V^i_R$ for $i \in [1..q]$ can be written in terms of basis variables. Since we have already seen how to write them in terms of $S^i, T^i, S^i_1, X^i_L, X^i_R, Y^i_L, Y^i_R$ for $i \in [1..q]$ and Z^i_j for $i \in [1..q], j \in [1..\ell^i - 1]$, and S^i, T^i, S^i_1 for $i \in [1..q]$ are already in the basis, it suffices to show that Z^i_j for $i \in [1..q], j \in [1..\ell^i - 1]$ and $X^i_L, X^i_R, Y^i_L, Y^i_R$ for $i \in [1..q]$ can be written in terms of basis variables. An expression of an internal variable in terms of basis variables and the oracle inputs and outputs will be called the *extension expression* of the basis variable. Thus, whenever we sample all the basis elements, we can extend this through these equations to assign values to all the internal variables.

For $i \in E, j \in [1..\ell^i]$, let i' be such that (i', j) is fresh, and $(L^{i'}_j, R^{i'}_j) = (L^i_j, R^i_j)$. Then, i' is called the *j-predecessor* of i, denoted $i : j$. Similarly, for $i \in D, j \in [1..\ell^i]$, if for some i' we have (i', j) fresh and $(L'^{i'}_j, R'^{i'}_j) = (L'^i_j, R'^i_j)$, we set $i : j = i'$. (Thus, when (i,j) is fresh, $i : j$ is i itself.) For $i \in E, j \in [1..\ell^i]$ we have from (1) that $X^i_j = X^{i:j}_j = L'^{i:j}_j + Z^{i:j}_j$, so

$$Z^i_j = L'^{i:j}_j + L'^i_j + Z^{i:j}_j; \tag{7}$$

and for $i \in D, j \in [1..\ell^i]$ we have from (2)

$$Y^i_j = Y^{i:j}_j = R^{i:j}_j + Z^{i:j}_j + S^{i:j}_1,$$

so

$$Z^i_j = R^{i:j}_j + R^i_j + Z^{i:j}_j + S^{i:j}_1 + S^i_1. \tag{8}$$

Now if i and $i : j$ are both in E or both in D, $Z^{i:j}_j$ is a basis element. (In particular, when $i : j = i$, Z^i_j is a basis element.) Otherwise, we can go back one step further to $(i : j) : j$, the j-predecessor of $i : j$, denoted $i : j^2$. We call (1) and (2) the *extension equations*. They will serve useful in the later proofs. Note that it does not hold in general that $(i : j) : j = i : j$. This holds only if $i : j$ and i are both in E or both in D, or when $i : j$ points to a fresh input block.

For $i \in [2..q]$, the smallest query index in $[1..i-1]$ which i is akin to is called the *origin* of i, denoted $\bar{i}$. We also define the origin of 1 to be 1 itself. Thus, for $i \in E$,

$$X_L^i = X_L^{\bar{i}}, \tag{9}$$

$$X_R^i = X_R^{\bar{i}}; \tag{10}$$

and for $i \in D$,

$$Y_L^i = Y_L^{\bar{i}}, \tag{11}$$

$$Y_R^i = Y_R^{\bar{i}}. \tag{12}$$

Since for $i \in E$, $X_L^{\bar{i}}$ and $X_R^{\bar{i}}$ are in the basis, and for $i \in D$, $Y_L^{\bar{i}}$ and $Y_R^{\bar{i}}$ are in the basis, this completes the extensions.

Oracles and Bad Events. The real oracle employs ZCZ$[\widetilde{\pi}]$ to answer the queries of $\mathbf{A}$. In the ideal world, the encryption oracle samples and returns $L_j'^i, R_j'^i$ for $i \in E, j \in [1..\ell^i]$ uniformly at random; the decryption oracle samples and returns L_j^i, R_j^i for $i \in D$, $j \in [1..\ell^i]$ uniformly at random. Once the interaction phase is over, the ideal world oracle samples and returns each basis element uniformly at random from $\{0,1\}^n$, with two exceptions:

- For $i \in E$, S^i is drawn uniformly from the set $\{0,1\}^n \setminus \left\{S^{i'} \mid i \text{ is akin to } i', R^i = R^{i'}\right\}$;
- For $i \in D$, T^i is drawn uniformly from the set $\{0,1\}^n \setminus \left\{T^{i'} \mid i \text{ is akin to } i', L'^i = L'^{i'}\right\}$.

The real world releases the values of the basis variables to the adversary. (thus, from the extension equations, $\mathbf{A}$ can calculate the values of the inputs, tweaks, and outputs of all internal TBC calls.) $\mathbf{A}$ shall distinguish the real world $\mathcal{O}_{\text{real}}$ from the ideal world $\mathcal{O}_{\text{ideal}}$, given a transcript τ of its interaction with the available oracles. We say that the event bad occurs when one of the following occurs:

- badA occurs when one of the following holds:
  — For some $i \in E, j \in [1..\ell^i]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j$ such that $(L_j'^{i'}, R_j'^{i'}) = (L_j'^i, R_j'^i)$;
  — For some $i \in D, j \in [1..\ell^i]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j$ such that $(L_j^{i'}, R_j^{i'}) = (L_j^i, R_j^i)$;
- badB occurs when for some $i \in [2..q]$ there exists $i' \in [1..i-1]$ with $\ell^i = \ell^{i'}$ such that one of the following holds:
  — $(U_L^i, U_R^i) = (U_L^{i'}, U_R^{i'})$;
  — $(S^i, U_R^i) = (S^{i'}, U_R^{i'})$;
  — $(S^i, T^i) = (S^{i'}, T^{i'})$;
  — $(V_L^i, T^i) = (V_L^{i'}, T^{i'})$;
  — $(V_L^i, V_R^i) = (V_L^{i'}, V_R^{i'})$;

– badC occurs when one of the following holds:
  — For some $i \in [1..q]$, there exists $i' \in [1..i-1]$ such that $(S_1^i, T^i) = (S_1^{i'}, T^{i'})$;
  — For some $i \in [1..q], j \in [1..\ell^i - 1]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j+1$ such that $(Z_j^i, T^i) = (Z_j^{i'}, T^{i'})$;
– badD occurs when one of the following holds:
  — For some $i \in E, j \in [1..\ell^i - 1]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j+1$ such that $(L'^i_j, Y^i_j) = (L'^{i'}_j, Y^{i'}_j)$;
  — For some $i \in D, j \in [1..\ell^i - 1]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j+1$ such that $(R^i_j, X^i_j) = (R^{i'}_j, X^{i'}_j)$;
– badE occurs when for some $i \in [2..q]$, there exists $i' \in [1..i-1]$ such that i is not akin to i' and yet one of the following holds:
  — $(X_L^{*i}, X_R^{*i}) = (X_L^{*i'}, X_R^{*i'})$;
  — $(Y_L^{*i}, Y_R^{*i}) = (Y_L^{*i'}, Y_R^{*i'})$;

Thus, $\mathsf{bad} \overset{\text{def}}{=} \mathsf{badA} \vee \mathsf{badB} \vee \mathsf{badC} \vee \mathsf{badD} \vee \mathsf{badE}$. Clearly,

$$\Pr[\mathsf{bad}] \leq \Pr[\mathsf{badA}] + \Pr[\mathsf{badB}] + \Pr[\mathsf{badC}] + \Pr[\mathsf{badD}] + \Pr[\mathsf{badE}]\,. \tag{13}$$

Now, we are in a position to prove Lemmas 2 and 3.

Proof of Lemma 2. Below, we show that each of the collision-pairs that would result in one of the bad events has a joint probability of $\leq 1/N^2$. Clearly, we need the assumption that all basis elements are uniformly sampled from $\{0,1\}^n$ for this purpose. Moreover, the values S^i and T^i are sampled without replacement under certain circumstances, their bound is at most $1/N(N-1)$, which can be upper bounded by $1/N(N-1) < 2/N^2$. Thus, for bounding the bad events, we simply need to bound the number of candidate collision-pairs.

For badA, there can be:

– at most $\sigma_E^2/2$ collision events of the form $(L'^{i'}_j, R'^{i'}_j) = (L'^i_j, R'^i_j)$;
– at most $\sigma_D^2/2$ collision events of the form $(L^{i'}_j, R^{i'}_j) = (L^i_j, R^i_j)$;

where σ_E is the total number of encryption query blocks and σ_D is the total number of decryption query blocks, so that $\sigma_E^2 + \sigma_D^2 \leq \sigma^2$. Thus

$$\Pr[\mathsf{badA}] \leq \frac{\sigma^2}{N^2}. \tag{14}$$

For badB, there can be:

– at most $q^2/2$ collision events of the form $(U_L^i, U_R^i) = (U_L^{i'}, U_R^{i'})$;
– at most $q^2/2$ collision events of the form $(S^i, U_R^i) = (S^{i'}, U_R^{i'})$;
– at most $q^2/2$ collision events of the form $(S^i, T^i) = (S^{i'}, T^{i'})$;
– at most $q^2/2$ collision events of the form $(V_L^i, T^i) = (V_L^{i'}, T^{i'})$;
– at most $q^2/2$ collision events of the form $(V_L^i, V_R^i) = (V_L^{i'}, V_R^{i'})$;

Thus

$$\Pr[\mathsf{badB}] \leq \frac{5q^2}{N^2}. \tag{15}$$

For badC, there can be:

- at most $q^2/2$ collision events of the form $(S_1^i, T^i) = (S_1^{i'}, T^{i'})$.
- at most $\sigma^2/2$ collision events of the form $(Z_j^i, T^i) = (Z_j^{i'}, T^{i'})$;

Thus

$$\Pr[\mathsf{badC}] \leq \frac{q^2 + \sigma^2}{N^2}. \tag{16}$$

For badD, there can be:

- at most $\sigma_E^2/2$ collision events of the form $(L_j'^i, Y_j^i) = (L_j'^{i'}, Y_j^{i'})$;
- at most $\sigma_D^2/2$ collision events of the form $(R_j^i, X_j^i) = (R_j^{i'}, X_j^{i'})$.

Thus

$$\Pr[\mathsf{badD}] \leq \frac{\sigma^2}{N^2}. \tag{17}$$

For badE, there can be:

- at most $q^2/2$ collision events of the form $(X_L^{*i}, X_R^{*i}) = (X_L^{*i'}, X_R^{*i'})$;
- at most $q^2/2$ collision events of the form $(Y_L^{*i}, Y_R^{*i}) = (Y_L^{*i'}, Y_R^{*i'})$.

Thus

$$\Pr[\mathsf{badE}] \leq \frac{2q^2}{N^2}. \tag{18}$$

The lemma follows from (13)–(18).

Now, all that is left to do is to establish our claim that each of the collision-pairs that would result in one of the bad events has a joint probability $\leq 1/N^2$. This is to be done by examining each bad event separately. badA, badB and badC are fairly straightforward, and we leave out the proofs. badD is more interesting; we provide below a complete analysis of it. The trickiest case is badE; here, due to space constraints, we only examine two of its main subcases in detail. The complete case-by-case analysis, along with a short analysis of badA, badB and badC, can be found in the Appendix.

Full Analysis of badD. We consider the two cases separately:

- $(L_j'^i, Y_j^i) = (L_j'^{i'}, Y_j^{i'})$, $i \in E$, $i' < i$: We will show that $Y_j^i = Y_j^{i'}$ always leads to an equation containing at least one basis variable that cannot get canceled out. The required bound follows since the basis variable and $L_j'^i$ are independently sampled. From (2) we have

$$R_j^i + Z_j^i + S_1^i = R_j^{i'} + Z_j^{i'} + S_1^{i'}. \tag{19}$$

  Note that S_1^i cannot occur in the expansion of $Z_j^{i:j}$, since $i \in E$. Now we have two options of i':

- $i' \in E$: From (7) and (19) we have

$$R^i_j + L'^{i:j}_j + L'^i_j + Z^{i:j}_j + S^i_1 = R^{i'}_j + L'^{i':j}_j + L'^{i'}_j + Z^{i':j}_j + S^{i'}_1.$$

  Here the basis element S^i_1 cannot be canceled out, since $i' < i$.
- $i' \in D$: From (7), (8) and (19), we have

$$R^i_j + L'^{i:j}_j + L'^i_j + Z^{i:j}_j + S^i_1 = R^{i'}_j + R^{i':j}_j + R^{i'}_j + Z^{i':j}_j + S^{i':j}_1.$$

  Again, the basis element S^i_1 cannot be canceled out since $i' : j \leq i' < i$.

– $(R^i_j, X^i_j) = (R^{i'}_j, X^{i'}_j)$, $i \in D$, $i' < i$: As above, we show that $X^i_j = X^{i'}_j$ always leads to an equation containing at least one basis variable that cannot get canceled out, and the required bound follows since the basis variable and R^i_j are independently sampled. From (1), we have

$$L'^i_j + Z^i_j = L'^{i'}_j + Z^{i'}_j. \tag{20}$$

Now, we have two options of i':

- $i' \in E$: From (8), (7) and (20), we have

$$L'^i_j + R^{i:j}_j + R^i_j + Z^{i:j}_j + S^{i:j}_1 + S^i_1 = L'^{i':j}_j + Z^{i':j}_j.$$

  When $i : j < i$, the basis element S^i_1 cannot be canceled out, and when $i = i : j$, we have $i' : j \leq i' < i = i : j$, so the basis element $Z^{i:j}_j = Z^i_j$ cannot be canceled out.
- $i' \in D$: From (8) and (19), we have

$$L'^i_j + R^{i:j}_j + R^i_j + Z^{i:j}_j + S^{i:j}_1 + S^i_1 = L'^{i'}_j + R^{i':j}_j + R^{i'}_j + Z^{i':j}_j + S^{i':j} + S^{i'},$$

  Here again, either S^i_1 or the basis element Z^i_j cannot be canceled out, and the argument is identical to the above.

Partial Analysis of badE. This is trickier than the other bad events, and requires some careful case analysis. We examine the two most difficult sub-cases here. Let $i' < i$ and $\ell \overset{\text{def}}{=} \ell^{i'} = \ell^i$, and let $\alpha_j(\cdot)$ and $\alpha^2_j(\cdot)$ be linear functions defined as

$$\alpha_j(x) \overset{\text{def}}{=} \alpha^{\ell-1-j} \cdot x \quad \text{and} \quad \alpha^2_j(x) \overset{\text{def}}{=} (\alpha^2)^{\ell-1-j} \cdot x.$$

Both the sub-cases we examine here fall under the case of $(X^{*i}_L, X^{*i}_R) = (X^{*i'}_L, X^{*i'}_R)$. We can write this collision as

$$\sum_{j=0}^{\ell-1} \alpha_j(X^i_j + X^{i'}_j) = 0 \quad \text{and} \quad \sum_{j=0}^{\ell-1} \alpha^2_j(X^i_j + X^{i'}_j) = \sum_{j=0}^{\ell-1} \alpha^2_j(R^i_j + R^{i'}_j).$$

Using (1) we can rewrite these as

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^i + Z_j^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j (L_j'^i + L_j'^{i'}), \tag{21}$$

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^i + Z_j^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^i + L_j'^{i'} + R_j^i + R_j^{i'}). \tag{22}$$

We first observe that since i is not akin to i', $X_j^i + X_j^{i'}$ cannot trivially disappear for all $j \in [1, .., \ell-1]$. Also, since $\alpha_j(X_j^i + X_j^{i'})$ sum to 0, there must be at least two indices in $[1, .., \ell-1]$ where $X_j^i + X_j^{i'}$ does not trivially disappear; let j_0 and j_1 be the two largest such indices, with $j_0 > j_1$. Now, we first consider the sub-case $i \in E, i' \in E$. From (7), (21) and (22) we have

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j (L_j'^{i:j} + L_j'^{i':j}), \tag{23}$$

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}). \tag{24}$$

By choice of j_0, $i : j_0 \neq i' : j_0$. Suppose $i : j_0 > i' : j_0$. If $i : j_0 \in D$, using (8), we replace $Z_{j_0}^{i:j_0}$ by $R_{j_0}^{i:j_0^2} + R_{j_0}^{i:j_0} + Z_{j_0}^{i:j_0^2} + S_1^{i:j_0^2} + S_1^{i:j_0}$. The basis element $S_1^{i:j_0}$ does not get canceled out; moreover, $R_{j_0}^{i:j_0}$ remains only in the top equation, while it gets canceled out in the bottom equation. Since $i : j = i' : j$ for all $j > j_0$, none of the adversary-queried blocks remaining in either equation came after $R_{j_0}^{i:j_0}$, so it is independent of the rest of the equation; along with the basis element $S_1^{i:j_0}$ (which appears in both equations), this makes the two collisions independent, thus occurring jointly with a probability $1/N^2$.

If $i : j_0 \in E$, $Z_{j_0}^{i:j_0}$ is in the basis, and does not cancel out. On the right hand side of both equations, $L_{j_0}'^{i:j_0}$ remains uncanceled as well, while all later adversary queries get canceled. Thus, the two equations can become dependent with probability at most $1/N$; then, the common collision can occur with probability at most $1/N$. Thus, in either case, the joint collision can occur with a probability of more than $1/N^2$. The analysis is similar when $i : j_0 < i : j_0$; then we focus on the latter instead.

The other sub-case we consider is $i \in E, i' \in D$. From (7), (8), (21) and (22) we have

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j} + S_1^{i'} + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j (L_j'^{i:j} + L_j'^{i':j} + R_j^{i'} + R_j^{i':j}), \tag{25}$$

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j} + S_1^{i'} + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}). \tag{26}$$

By choice of j_0 and j_1, $i : j_0 \neq i'$ and $i : j_1 \neq i'$. Suppose $i : j_0 < i'$. Then $S_1^{i'}$ and $R_{j_0}^{i'}$ remain uncanceled in (25), and no adversary query block queried after $R_{j_0}^{i'}$ remains uncanceled; in (26), $S_1^{i'}$ remains uncanceled again, but there is no $R_{j_0}^{i'}$ and no adversary query block queried after it. Thus these two can occur jointly with a probability at most $1/N^2$.

A symmetric argument can be used when $i : j_0 > i'$ and $i : j_0 \in D$: we replace $Z_{j_0}^{i:j_0}$ by $R_{j_0}^{i:j_0^2} + R_{j_0}^{i:j_0} + Z_{j_0}^{i:j_0^2} + S_1^{i:j_0^2} + S_1^{i:j_0}$ using (8), and observe that $S_1^{i:j_0}$ remains uncanceled in either equation, while $R_{j_0}^{i:j_0}$ remains uncanceled in (25), but gets canceled out in (26), and no adversary query block queried after it remains in either equation.

When $i : j_0 > i'$ and $i : j_0 \in E$, but $i : j_1$ satisfied one of the above two conditions, we can argue as above using $i : j_1$ instead. If we also have $i : j_1 > i'$ and $i : j_1 \in E$, we observe that $Z_{j_0}^{i:j_0}$ and $Z_{j_1}^{i:j_1}$ are basis elements that do not get canceled out in either equation. Their combined contribution to the left-hand side of (25) is $\alpha^{\ell-1-j_0} \cdot Z_{j_0}^{i:j_0} + \alpha^{\ell-1-j_1} \cdot Z_{j_1}^{i:j_1}$ and to the left-hand side of (26) is $(\alpha^2)^{\ell-1-j_0} \cdot Z_{j_0}^{i:j_0} + (\alpha^2)^{\ell-1-j_1} \cdot Z_{j_1}^{i:j_1}$. These two collisions are independent since $\alpha^{\ell-1-j_0} \cdot (\alpha^2)^{\ell-1-j_1} \neq \alpha^{\ell-1-j_1} \cdot (\alpha^2)^{\ell-1-j_0}$, and thus can occur with a probability at most $1/N^2$. The rest of the subcases can be analysed similarly. This completes the proof of Lemma 2. □

Proof of Lemma 3. Let τ be a good transcript, i. e., none of the events badA, badB, badC, badD, or badE occurred. Then, in the ideal world, there are 2σ samplings for generating the query responses and $\sigma_F + 2q_\nu + 5q$ for generating the basis elements. In the ideal world, the basis elements are sampled uniformly at random and independently from each other. Hence, the probability for those is given by

$$\frac{1}{N^{\sigma_F + 2q_\nu + 5q}}.$$

The situation differs for the outputs of the scheme. The ideal world is an ideal SPRP; hence, the outputs are sampled without replacement. Since all queries are from the domain $\mathcal{B}^{\leq n}$, we can group encryption and decryption queries into disjoint sets $\mathcal{L}^1, \ldots, \mathcal{L}^n$ s. t. their union contains all queries, and Set $\mathcal{L}^i$ contains exactly the queries of length i di-blocks. We define by $\textsc{Load}\left(\mathcal{L}^i\right)$ the number of queries in Set $\mathcal{L}^i$, for all $1 \leq i \leq n$. The probability for ciphertext outputs from encryption queries and plaintext outputs from decryption queries is

$$\prod_{i=1}^{n} \frac{1}{(N^{2i})_{\textsc{Load}(\mathcal{L}^i)}}.$$

Since each query has at least $2n$ bits, we can lower bound the probability by

$$\prod_{i=1}^{n} \frac{1}{(N^{2i})_{\textsc{Load}(\mathcal{L}^i)}} \leq \frac{1}{(N^2)_{2q}} \cdot \frac{1}{N^{2\sigma - 2q}}.$$

We obtain that

$$\Pr\left[\Theta_{\text{ideal}} = \tau\right] \leq \frac{1}{N^{\sigma_F + 2q_\nu + 3q + 2\sigma}} \cdot \frac{1}{\left(N^2\right)_q}. \tag{27}$$

In the real world, the construction employs a permutation $\widetilde{\pi}^{\mathsf{T}}(\cdot)$ for each tweak $\mathsf{T} \in \mathcal{T}_{\mathcal{D}\times\mathcal{I}}$ that was used in the transcript, . We write the set of all occurred tweaks of all di-blocks of all queries in the transcript and write it as $\left\{\mathsf{T}^1, \ldots, \mathsf{T}^\theta\right\}$. We further define by $\textsc{Load}\left(\mathsf{T}\right)$ the load of a tweak T, i.e., the number of distinct inputs used for it over all queries and di-blocks of the transcript. It holds that

$$\sum_{i=1}^{\theta} \textsc{Load}\left(\mathsf{T}^i\right) = \sigma_F + 2\sigma + 2q_\nu + 5q.$$

We adopt the notion of transcript-compatible permutations from [14]. We call $\widetilde{\pi}$ *compatible* with τ if for all queries, $\widetilde{\pi}$ produced all intermediate variables as well as all outputs in τ. Let $\mathsf{Comp}(\tau)$ denote the set of tweakable permutations $\widetilde{\pi}$ that are compatible with τ. Thus

$$\Pr\left[\Theta_{\text{real}} = \tau\right] = \Pr\left[\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}\left(\mathcal{T}_{D,I}, \{0,1\}^n\right) : \widetilde{\pi} \in \mathsf{Comp}(\tau)\right].$$

For a fixed tweak T, the fraction of compatible permutations is

$$\prod_{i=0}^{\textsc{Load}(\mathsf{T})-1} \frac{1}{N-i} = \frac{1}{(N)_{\textsc{Load}(\mathsf{T})}}.$$

Over all tweaks T^i, for $1 \leq i \leq \theta$, the fraction of compatible permutations is given by

$$\prod_{i=1}^{\theta} \frac{1}{(N)_{\textsc{Load}(\mathsf{T}^i)}}$$

It is hard to work with this probability directly. Instead, since we are interested in a bound for the real-world probability of transcripts, we can lower bound the probability of all $\sigma_F + 2q_\nu + 5q$ basis variables by the naive probability that they are all computed from fresh tweaks:

$$\frac{1}{N^{\sigma_F + 2q_\nu + 5q}}.$$

For the ciphertext and plaintext outputs, we can employ similar sets $\mathcal{L}^i$, for $1 \leq i \leq n$, as we had for the ideal world, where Set $\mathcal{L}^i$ again consists of all queries of length i di-blocks. The probability of outputs in the real world can then be lower bounded by

$$\prod_{i=1}^{n} \frac{1}{\left(N^{2i}\right)_{\textsc{Load}(\mathcal{L}^i)}}.$$

Now, we can upper bound the ratio of the probability of our transcripts by

$$\begin{aligned}\frac{\Pr\left[\Theta_{\text{real}} = \tau\right]}{\Pr\left[\Theta_{\text{ideal}} = \tau\right]} &\geq \frac{\frac{1}{N^{\sigma_F + 2q_\nu + 5q}} \cdot \prod_{i=1}^{n} \frac{1}{(N^{2i})_{\textsc{Load}(\mathcal{L}^i)}}}{\frac{1}{N^{\sigma_F + 2q_\nu + 5q}} \cdot \frac{1}{N^{2\sigma - 2q}} \cdot \frac{1}{(N^2)_q}} \\ &\geq \frac{\prod_{i=1}^{n} \frac{1}{(N^{2i})_{\textsc{Load}(\mathcal{L}^i)}}}{\frac{1}{(N^2)_q} \cdot \frac{1}{N^{2\sigma - 2q}}} \geq \frac{\left(N^2\right)_q \cdot N^{2\sigma - 2q}}{N^{2\sigma}} = \frac{\left(N^2\right)_q}{(N^2)^q} \\ &= \frac{(N^2)(N^2-1) \cdot \cdots \cdot (N^2 - q + 1)}{(N^2)^q} \geq \left(\frac{N^2 - q + 1}{N^2}\right)^q \\ &\geq \left(\frac{N^2 - q}{N^2}\right)^q = \left(1 - \frac{q}{N^2}\right)^q \geq 1 - \frac{q^2}{N^2},\end{aligned}$$

where the last inequality is Bernoulli's. So, we obtain our claim in Lemma 3. □

6.2 Proof Sketch for Messages with Arbitrary Number of Complete Di-blocks

Theorem 2. Let $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}_{D,I}, \{0,1\}^n)$. Let $\mathbf{A}$ be an SPRP adversary on $\mathrm{ZCZ}[\widetilde{\pi}]$ that asks at most q queries queries of domain $\mathcal{B}^+$, whose lengths sum up to at most σ di-blocks in total, and $\mathbf{A}$ runs in time at most $\textsc{time}$. Then

$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathrm{ZCZ}[\widetilde{\pi}]}(\mathbf{A}) \leq \frac{4\sigma^2 + 8q^2}{N^2}.$$

Proof Sketch. The proof follows a similar strategy as that of Theorem 1. So, we only consider the equations in the analysis of bad events that differ. We add each S^i_k, $i \in [1..q]$, $k \in \left[1..\left\lceil \ell^i/n \right\rceil\right]$ to the basis. The ideal oracle samples the additional basis elements along with the original basis elements in the second step, and the definitions of the bad cases do not change. From the equations (1)–(6) that we began with, only (2) is now replaced by

$$Y^i_j = R^i_j + Z^i_j + S^i_{\lceil j/n \rceil}. \tag{2'}$$

In the extension equations, this changes only (8), which is replaced by

$$Z^i_j = R^{i:j}_j + R^i_j + Z^{i:j}_j + S^{i:j}_{\lceil j/n \rceil} + S^i_{\lceil j/n \rceil}. \tag{8'}$$

The definitions of the bad cases remain the same except badC, which now occurs when:

- For some $i \in [1..q], k \in \left[1..\left\lceil \ell^i/n \right\rceil\right]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq n(k-1)$ s.t. $(S^i_k, T^i) = (S^{i'}_k, T^{i'})$;
- For some $i \in [1..q], j \in [1..\ell^i - 1]$, there exists $i' \in [1..i-1]$ with $\ell^{i'} \geq j+1$ s.t. $(Z^i_{k,c}, T^i) = (Z^{i'}_{k,c}, T^{i'})$, where $k = \lceil j/n \rceil, c = j - n(k-1)$.

Of these, the counting does not change for the latter; for the former, there are now at most $c_{\max}q^2/2$ possible collision pairs now, where $c_{\max}$ is the maximum number of chunks in one query; we generously bound this by $\sigma^2/2$. This adds $(\sigma^2 - q^2)/2N$ to our earlier bound, to obtain the new bound for the extended version. To ensure that the counting argument for badE still goes through, we only note that for $k \in [1..\lceil \ell/n \rceil]$, S_k^i can only occur in any of the collision equations from badE with coefficients $\beta^{\ell-1-j}$ for $j \in [n(k-1)+1..nk]$, where β is either α or α^2, and for any choice of k, a non-empty subset of these coefficients cannot add to 0.

6.3 Proof Sketch for the Security of ZCZ*

Theorem 3. Let $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}_{D,I}, \{0,1\}^n)$. Let $\mathbf{A}$ be an SPRP adversary on $\mathrm{ZCZ}^*[\widetilde{\pi}]$ that asks at most q queries of domain $\{0,1\}^{\geq 2n}$, whose lengths sum up to at most σ di-blocks in total, q' of which contains an incomplete di-block at the end. Then

$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathrm{ZCZ}^*[\widetilde{\pi}]}(\mathbf{A}) \leq \frac{4\sigma^2 + 8q^2 + 9q'^2}{N^2}.$$

Proof Sketch. The ideal oracle's sampling mechanism for the tweakable blockcipher outputs for the partial di-block messages is slightly trickier. Let $\mathcal{I}$ denote the indices of the queries with incomplete di-blocks. Instead of simulating an ideal permutation, the ideal oracle simulates what [22] calls an $\pm\widetilde{\mathbf{rnd}}$ oracle, which always returns random bits, as long as no pointless queries are asked. (It is easy to argue for our construction why not permitting pointless queries does not diminish the adversary's power, so we can confine our attention to the no-pointless-query scenario.)

We use the notation $(U,V), (U_m,V_m), (U',V')$ for outputs of the blockcipher calls in the top, middle, and bottom layers respectively. M_j denotes (L_j, R_j), and $*$ denotes the index of the incomplete di-block.

- For the smallest $i \in \mathcal{I}$, $U_*^i, V_*^i, U_*'^i, V_*^i$ are sampled uniformly from $\{0,1\}^n$;
- For each i in $\mathcal{I}$ such that for no i' in $\mathcal{I}$ with $i' < i$ we have $(L_*^i, R_*^i) \neq (L_*^{i'}, R_*^{i'})$:
  - U_*^i is sampled uniformly from $\{0,1\}^n \setminus \left\{U_*^{i'} \mid i' \in \mathcal{I}, i' < i\right\}$;
  - V_*^i is sampled uniformly from $\{0,1\}^n \setminus \left\{V_*^{i'} \mid i' \in \mathcal{I}, i' < i\right\}$;
- For each i in $\mathcal{I}$ such that for no i' in $\mathcal{I}$ with $i' < i$ we have $(L_*'^i, R_*'^i) \neq (L_*'^{i'}, R_*'^{i'})$:
  - $U_*'^i$ is sampled uniformly from $\{0,1\}^n \setminus \left\{U_*'^{i'} \mid i' \in \mathcal{I}, i' < i\right\}$;
  - $V_*'^i$ is sampled uniformly from $\{0,1\}^n \setminus \left\{V_*'^{i'} \mid i' \in \mathcal{I}, i' < i\right\}$;
- For each $i \in \mathcal{I}$ the $(2n-s)$-bit suffix R^i of (U_{m*}^i, V_{m*}^i) is sampled uniformly from $\{0,1\}^{2n-s}$, and (U_{m*}^i, V_{m*}^i) is set to $(M_*^i + M_*'^i)||R^i$.

The new bad cases are:

- For some distinct i, i' in $\mathcal{I}$ with $\ell^i = \ell^{i'} = \ell$ we have
$$(M^i_{1..\ell-1}, M^i_\ell + (U^i_*, V^i_*)) = (M^{i'}_{1..\ell-1}, M^{i'}_\ell + (U^{i'}_*, V^{i'}_*));$$
- For some distinct i, i' in $\mathcal{I}$ with $\ell^i = \ell^{i'} = \ell$ we have
$$(M'^i_{1..\ell-1}, M'^i_\ell + (U'^i_*, V'^i_*)) = (M'^{i'}_{1..\ell-1}, M'^{i'}_\ell + (U'^{i'}_*, V'^{i'}_*));$$
- For some distinct i, i' in $\mathcal{I}$ with $\ell^i = \ell^{i'} = \ell$ we have
$$\begin{aligned}&(L^i_\ell + L'^i_\ell + U^i_* + U'^i_*, R^i_\ell + R'^i_\ell + V^i_* + V'^i_*)\\ &= (L^{i'}_\ell + L'^{i'}_\ell + U^{i'}_* + U'^{i'}_*, R^{i'}_\ell + R'^{i'}_\ell + V^{i'}_* + V'^{i'}_*);\end{aligned}$$
- For some distinct i, i' in $\mathcal{I}$ with $\ell^i = \ell^{i'} = \ell$ we have
$$(R^i_\ell + R'^i_\ell + V^i_* + V'^i_*, U^i_{m*}) = (R^{i'}_\ell + R'^{i'}_\ell + V^{i'}_* + V'^{i'}_*, U^{i'}_{m*});$$
- For some distinct i, i' in $\mathcal{I}$ with $\ell^i = \ell^{i'} = \ell$ we have
$$(R^i_\ell + R'^i_\ell + V^i_* + V'^i_*, V^i_{m*}) = (R^{i'}_\ell + R'^{i'}_\ell + V^{i'}_* + V'^{i'}_*, V^{i'}_{m*}).$$

The probabilities of these bad cases can be bounded by $q'^2/2N'^2$, $q'^2/2N'^2$, $q'^2/2N'^2$, $q'^2/2NN'$, $q'^2/2NN'$ in that order, where $N' = N - q'$. With the reasonable assumption that $q' \le N/2$, we can replace N' with $N/2$ in these bounds and have them sum to $8q'^2/N^2$, which is our bound for the combined probability of the new bad cases. The theorem follows from Theorem 2 and Lemma 6 of [22].

Our results in Theorems 1 and 3 had considered the instantiation with an ideal random tweaked permutation $\widetilde{\pi} \twoheadleftarrow \widetilde{\mathsf{Perm}}(\mathcal{T}_{I,D}, \{0,1\}^n)$. Corollaries 1 and 3 yield the resulting security bounds when ZCZ and ZCZ* are instantiated with a given tweakable block cipher $\widetilde{E}_K : \mathcal{K} \times \mathcal{T}_{I,D} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher with $K \twoheadleftarrow \mathcal{K}$.

Corollary 1. Let $\mathbf{A}$ be an SPRP adversary on ZCZ$[\widetilde{E}_K]$, s.t. $\mathbf{A}$ asks at most q queries of domain $\mathcal{B}^{\le n}$, that sum up to at most σ di-blocks in total, and $\mathbf{A}$ runs in time at most TIME. Then
$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathrm{ZCZ}[\widetilde{E}_K]}(\mathbf{A}) \le \frac{3\sigma^2 + 10q^2}{2N^2} + \mathbf{Adv}^{\mathrm{STPRP}}_{\widetilde{E}_K, \widetilde{E}^{-1}_K}(\mathbf{A}'),$$
where $\mathbf{A}'$ is an STPRP adversary against $\widetilde{E}_K$ that asks at most $a' = 3\sigma + \lceil \sigma/n \rceil + 6q$ queries and runs in time at most $\mathrm{TIME} + O(a')$.

Corollary 2. Let $\mathbf{A}$ be an SPRP adversary on ZCZ$^*[\widetilde{E}_K]$ that asks at most q queries of domain $\{0,1\}^{\ge 2n}$, whose lengths sum up to at most σ di-blocks in total, q' of which contains an incomplete di-block at the end, and $\mathbf{A}$ runs in time at most TIME. Then
$$\mathbf{Adv}^{\mathrm{SPRP}}_{\mathrm{ZCZ}^*[\widetilde{E}_K]}(\mathbf{A}) \le \frac{4\sigma^2 + 8q^2 + 9q'^2}{N^2} + \mathbf{Adv}^{\mathrm{STPRP}}_{\widetilde{E}_K, \widetilde{E}^{-1}_K}(\mathbf{A}'),$$
where $\mathbf{A}'$ is an STPRP adversary against $\widetilde{E}_K$ that asks at most $a' = 3\sigma + \lceil \sigma/n \rceil + 6q + 6q'$ queries and runs in time at most $\mathrm{TIME} + O(a')$.

7 Instantiation

Choice of A Primitive. ZCZ requires a tweakable block cipher with support for usable tweak sizes that match at least the block size. The additional domain and block counter information require that the tweak is in fact somewhat longer. There exist a number of performant tweakable block ciphers that appear senseful for instantiations: SKINNY-64-192, SKINNY-128-384 [3], Deoxys-BC-128-384 [31], Joltik-64-192 [30], MANTIS [3], and QARMA [2]. For performance reasons, Deoxys-BC-128-384 [31] is particularly well-suited since it it can exploit the AES-round instructions. Note that we employ the updated variant from the Deoxys submission v1.41 from the CAESAR competition, that differs slightly from the variant in the original proposal of the TWEAKEY framework [30]. For the sake of completeness, we provide a very brief overview of it.

Brief overview of Deoxys-BC. Deoxys-BC is an AES-based block cipher that follows the TWEAKEY concept [31], where key and tweak are merged into a tweakey. Its round function is exactly that of the AES, where Deoxys-384 uses 16 rounds. All rounds contain the same operations SubBytes, ShiftRows, MixColumns, and AddRoundTweakey, i.e, the final round also employs MixColumns.

For Deoxys-BC-384, the tweakey consists of three 128-bit words TK^1, TK^2, and TK^3, which are processed in a linear schedule to derive the subtweakeys STK_0 through STK_{16}. First, the words are simply copied from the $3 \cdot 128$-bit tweakey: $(\mathrm{TK}_0^1 \,\|\, \mathrm{TK}_0^2 \,\|\, \mathrm{TK}_0^3) \leftarrow \mathrm{TK}$. In each round, each byte of TK^2 is transformed by a linear update function LFSR_2; similarly, each byte of TK^3 is transformed by a linear update function LFSR_3. Thereupon, each word is updated independently from the others by the same byte permutation $h : (\mathbb{F}_2^8)^{16} \to (\mathbb{F}_2^8)^{16}$, which permutes the bytes.

$$\begin{aligned}
\mathrm{TK}_r^1 &\leftarrow h(\mathrm{TK}_{r-1}^1),\\
\mathrm{TK}_r^2 &\leftarrow h(\mathrm{LFSR}_2(\mathrm{TK}_{r-1}^2)),\\
\mathrm{TK}_r^3 &\leftarrow h(\mathrm{LFSR}_3(\mathrm{TK}_{r-1}^3)).
\end{aligned}$$

The XOR of the results and a round constant RC_i yields the round tweakey: $\mathrm{STK}_i \leftarrow \mathrm{TK}_i^1 \oplus \mathrm{TK}_i^2 \oplus \mathrm{TK}_i^3 \oplus RC_i$, for $0 \leq i \leq 16$.

Version 1.41 of Deoxys defines the permutation as

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix} \xrightarrow{h} \begin{pmatrix} 1 & 5 & 9 & 13 \\ 6 & 10 & 14 & 2 \\ 11 & 15 & 3 & 7 \\ 12 & 0 & 4 & 8 \end{pmatrix},$$

and the linear feedback-shift registers $\mathrm{LFSR}_2 : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ and $\mathrm{LFSR}_3 : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ as

$$\begin{aligned}
(x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) &\xrightarrow{\mathrm{LFSR}_2} (x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0 \| (x_7 \oplus x_5)),\\
(x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) &\xrightarrow{\mathrm{LFSR}_3} ((x_0 \oplus x_6) \| x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1).
\end{aligned}$$

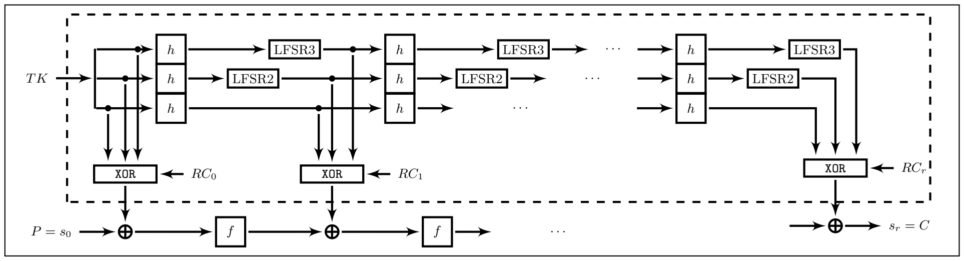


Fig. 5: Schedule for three tweakey words [29]. f denotes the AES round function.

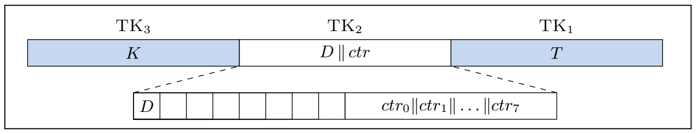


Fig. 6: Tweakey format of our instantiation, K is the secret key, T the tweak block, D the domain, and ctr an eight-byte counter, where ctr_0 is its least-significant and ctr_7 its most significant byte.

The round constants are given as

$$RC_i \stackrel{\text{def}}{=} \begin{pmatrix} 1 \ \mathtt{RCON}[i] \ 0 \ 0 \\ 2 \ \mathtt{RCON}[i] \ 0 \ 0 \\ 4 \ \mathtt{RCON}[i] \ 0 \ 0 \\ 8 \ \mathtt{RCON}[i] \ 0 \ 0 \end{pmatrix},$$

where $\mathtt{RCON}[i]$ denotes the i-th key-schedule constant of the AES:

$$\mathtt{RCON}[0..16] \stackrel{\text{def}}{=} \{\mathtt{2f}, \mathtt{5e}, \mathtt{bc}, \mathtt{63}, \mathtt{c6}, \mathtt{97}, \mathtt{35}, \mathtt{6a}, \mathtt{d4}, \mathtt{b3}, \mathtt{7d}, \mathtt{fa}, \mathtt{ef}, \mathtt{c5}, \mathtt{91}, \mathtt{39}, \mathtt{72}\}.$$

A high-level view of the schedule is given in Figure 5.

Encoding of The Tweak Components. Our instantiation employed the following characteristics:

– The domain space is given by $\mathcal{D} =^{\text{def}} \mathbb{F}_2^8$ with the chosen constants (t, b, c, s, , t\$, c\$, b\$, s\$, xl, xr, yl, yr, p, kd) $= (0, 1, \ldots, 12)$.
– The tweak-counter space is defined by $\mathcal{I} =^{\text{def}} \mathbb{F}_2^{64}$. As a consequence, the instantiation can encrypt up to 2^{64} di-blocks of $2 \cdot 128$ bits each, or 2^{69} bytes.
– Tweak, key, and block spaces are defined as $\mathcal{T} = \mathcal{K} = \mathbb{F}_{2^{128}}$.

The encoding is illustrated in Figure 6. The format is optimized for little-endian platforms. Moreover, it is optimized towards the tweakey schedule, in particular, towards the fact that TK_1 requires the least efforts of updating. Therefore, the tweak part T that is updated in each call in ZHash is used as TK_1, whereas the more expensive parts are used for the fixed key K as well as domain and counter.

Table 2: Performance of our implementation of ZCZ with Deoxys-BC-128-384 on Skylake with AVX2 and AES-NI support in the single-message setting.

Message length (bytes)									
128	256	512	1 kB	2 kB	4 kB	8 kB	16 kB	32 kB	64 kB
23.30	18.11	11.88	8.64	7.06	6.21	5.91	5.65	5.48	5.42

Performance Results. We implemented two C versions of an instantiation of ZCZ with Deoxys-BC-128-384, as defined in Section 4 and above: one reference version and a version that employs AVX2 and AES native instructions for Intel processors.[4] More comments on certain implementation optimizations can be found in Appendix E. We benchmarked our optimized implementation on an Intel Core i5-6200U Skylake CPU at 2.3 GHz with 8 GB RAM with Intel TurboBoost, HyperThreading, and Intel SpeedStep disabled. We used the median from $10,000$ measurements each. Table 2 illustrates our results. We excluded the costs for key setup of K; the remaining costs are integrated.

Acknowledgments.

The authors thank all anonymous reviewers for their fruitful comments that greatly helped to improve this work.

References

1. 1619.2-2010, I.S.: IEEE Standard for Wide-Block Encryption for Shared Storage Media. IEEE Std 1619.2-2010 pp. 1–91 (March 2011). https://doi.org/10.1109/IEEESTD.2011.5729263
2. Avanzi, R.: The QARMA Block Cipher Family. Almost MDS Matrices Over Rings With Zero Divisors, Nearly Symmetric Even-Mansour Constructions With Non-Involutory Central Rounds, and Search Heuristics for Low-Latency S-Boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017). https://doi.org/10.13154/tosc.v2017.i1.4-44
3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO II. LNCS, vol. 9815, pp. 123–153. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_5
4. Bellare, M., Micciancio, D.: A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. In: Fumy, W. (ed.) EUROCRYPT. LNCS, vol. 1233, pp. 163–192. Springer (1997). https://doi.org/10.1007/3-540-69053-0_13
5. Bernstein, D.J.: Some Challenges in Heavyweight Cipher Design. Tech. rep. (January 11 2016), `https://cr.yp.to/talks/2016.01.15/slides-djb-20160115-a4.pdf`

[4] The source code is available freely in the public domain at `https://github.com/medsec/zcz`.

6. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Assche, G.V., Keer, R.V.: Farfalle: parallel permutation-based cryptography. IACR Transactions on Symmetric Cryptology **2017**(4), 1–38 (2017), https://tosc.iacr.org/index.php/ToSC/article/view/801
7. Bhaumik, R., Nandi, M.: An Inverse-free Single Keyed Tweakable Enciphering Scheme. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT, Part II. LNCS, vol. 9453, pp. 159–180. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_7
8. Biryukov, A., Daemen, J., Lucks, S., Vaudenay, S.: Topics and Research Directions for Symmetric Cryptography. In: Early Symmetric Crypto Workshop. vol. 2017 (2017), https://www.cryptolux.org/mediawiki-esc2017/images/9/9a/ASJS-Topics_SymCrypto-ESC17.pdf
9. Chakraborty, D., Ghosh, S., Lopez, C.M., Sarkar, P.: FAST: A New Family of Secure and Efficient Tweakable Enciphering Schemes. Cryptology ePrint Archive, Report 2017/849 (2017), http://eprint.iacr.org/2017/849
10. Chakraborty, D., Mancillas-López, C., Rodríguez-Henríquez, F., Sarkar, P.: Efficient Hardware Implementations of BRW Polynomials and Tweakable Enciphering Schemes. IEEE Trans. Computers **62**(2), 279–294 (2013). https://doi.org/10.1109/TC.2011.227
11. Chakraborty, D., Mancillas-López, C., Sarkar, P.: STES: A Stream Cipher Based Low Cost Scheme for Securing Stored Data. IACR Cryptology ePrint Archive **2013**, 347 (2013), http://eprint.iacr.org/2013/347
12. Chakraborty, D., Sarkar, P.: A New Mode of Encryption Providing a Tweakable Strong Pseudo-Random Permutation. In: Robshaw, M.J.B. (ed.) FSE. LNCS, vol. 4047, pp. 293–309. Springer (2006). https://doi.org/10.1007/11799313_19
13. Chakraborty, D., Sarkar, P.: HCH: A New Tweakable Enciphering Scheme Using the Hash-Counter-Hash Approach. IEEE Transactions on Information Theory **54**(4), 1683–1699 (2008). https://doi.org/10.1109/TIT.2008.917623
14. Chen, S., Steinberger, J.P.: Tight Security Bounds for Key-Alternating Ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT. LNCS, vol. 8441, pp. 327–350. Springer (2014). https://doi.org/10.1007/978-3-642-55220-5_19
15. Chen, Y.L., Luykx, A., Mennink, B., Preneel, B.: Efficient Length Doubling From Tweakable Block Ciphers. IACR Trans. Symmetric Cryptol. **2017**(3), 253–270 (2017). https://doi.org/10.13154/tosc.v2017.i3.253-270
16. Cogliati, B., Lee, J., Seurin, Y.: New Constructions of MACs from (Tweakable) Block Ciphers. In: IACR Transactions on Symmetric Cryptology. vol. 2017, pp. 27–58 (2017). https://doi.org/10.13154/tosc.v2017.i2.27-58
17. Coron, J., Dodis, Y., Mandal, A., Seurin, Y.: A Domain Extender for the Ideal Cipher. In: Micciancio, D. (ed.) TCC. LNCS, vol. 5978, pp. 273–289. Springer (2010). https://doi.org/10.1007/978-3-642-11799-2_36
18. García, N.I.G.T., Chakraborty, D.: Efficient Software Implementations of Disk Encryption Schemes Using AES-NI Support. Master's thesis, Unidad Zacatenco, Departamento de Computacion (Feb 2012)
19. Gueron, S., Mouha, N.: Simpira v2: A Family of Efficient Permutations Using the AES Round Function. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT, Part I. LNCS, vol. 10031, pp. 95–125 (2016). https://doi.org/10.1007/978-3-662-53887-6_4
20. Halevi, S.: EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT. LNCS, vol. 3348, pp. 315–327. Springer (2004). https://doi.org/10.1007/978-3-540-30556-9_25

21. Halevi, S.: Invertible Universal Hashing and the TET Encryption Mode. In: Menezes, A. (ed.) CRYPTO. LNCS, vol. 4622, pp. 412–429. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_23
22. Halevi, S., Rogaway, P.: A Tweakable Enciphering Mode. In: Boneh, D. (ed.) CRYPTO. LNCS, vol. 2729, pp. 482–499. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_28
23. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. In: Okamoto, T. (ed.) CT-RSA. LNCS, vol. 2964, pp. 292–304. Springer (2004)
24. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v5. http://competitions.cr.yp.to/caesar-submissions.html (2014)
25. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust Authenticated-Encryption AEZ and the Problem That It Solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT (1). LNCS, vol. 9056, pp. 15–44. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_2
26. Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: Robshaw, M.J.B. (ed.) FSE. LNCS, vol. 4047, pp. 310–327. Springer (2006). https://doi.org/10.1007/11799313_20
27. Iwata, T., Minematsu, K.: Stronger Security Variants of GCM-SIV. IACR Trans. Symmetric Cryptol. **2016**(1), 134–157 (2016). https://doi.org/10.13154/tosc.v2016.i1.134-157
28. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication. In: Katz, J., Shacham, H. (eds.) CRYPTO, Part III. LNCS, vol. 10403, pp. 34–65. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_2
29. Jean, J.: TikZ for Cryptographers. https://www.iacr.org/authors/tikz/ (2016)
30. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT (2). LNCS, vol. 8874, pp. 274–288 (2014). https://doi.org/10.1007/978-3-662-45608-8_15
31. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1.41 (2016), third-round submission to the CAESAR competition. https://competitions.cr.yp.to/round3/deoxysv141.pdf
32. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO. LNCS, vol. 2442, pp. 31–46. Springer (2002). https://doi.org/10.1007/s00145-010-9073-y
33. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC Mode for Lightweight Block Ciphers. In: Peyrin, T. (ed.) FSE. LNCS, vol. 9783, pp. 43–59. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_3
34. McGrew, D.A., Fluhrer, S.R.: The Extended Codebook (XCB) Mode of Operation. IACR Cryptology ePrint Archive **2004**, 278 (2004), https://eprint.iacr.org/2004/278
35. McGrew, D.A., Fluhrer, S.R.: The Security of the Extended Codebook (XCB) Mode of Operation. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) SAC. LNCS, vol. 4876, pp. 311–327. Springer (2007). https://doi.org/10.1007/978-3-540-77360-3_20
36. Minematsu, K.: Beyond-Birthday-Bound Security Based on Tweakable Block Cipher. In: Dunkelman, O. (ed.) FSE. LNCS, vol. 5665, pp. 308–326. Springer (2009). https://doi.org/10.1007/978-3-642-03317-9_19
37. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT. LNCS, vol. 8441, pp. 275–292. Springer (2014), https://doi.org/10.1007/978-3-642-55220-5_16

38. Minematsu, K.: Building blockcipher from small-block tweakable blockcipher. Designs, Code and Cryptography **74**(3), 645–663 (2015). https://doi.org/10.1007/s10623-013-9882-8
39. Minematsu, K., Iwata, T.: Building Blockcipher from Tweakable Blockcipher: Extending FSE 2009 Proposal. In: Chen, L. (ed.) IMACC. pp. 391–412 (2011). https://doi.org/10.1007/978-3-642-25516-8_24
40. Minematsu, K., Matsushima, T.: Tweakable Enciphering Schemes from Hash-Sum-Expansion. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT. LNCS, vol. 4859, pp. 252–267. Springer (2007). https://doi.org/10.1007/978-3-540-77026-8_19
41. Naito, Y.: Full PRF-Secure Message Authentication Code Based on Tweakable Block Cipher. In: Au, M.H., Miyaji, A. (eds.) ProvSec. LNCS, vol. 9451, pp. 167–182. Springer (2015), https://link.springer.com/chapter/10.1007/978-3-319-26059-4_9
42. Nandi, M.: Improving upon HCTR and Matching Attacks for Hash-Counter-Hash Approach. IACR Cryptology ePrint Archive **2008**, 90 (2008), http://eprint.iacr.org/2008/090
43. Nandi, M.: A Generic Method to Extend Message Space of a Strong Pseudorandom Permutation. Computación y Sistemas **12**(3) (2009), http://cys.cic.ipn.mx/ojs/index.php/CyS/article/view/1204
44. Nandi, M.: On the Optimality of Non-Linear Computations of Length-Preserving Encryption Schemes. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT (II). LNCS, vol. 9453, pp. 113–133. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_5
45. Naor, M., Reingold, O.: A Pseudo-Random Encryption Mode (1997), manuscript available from www.wisdom.weizmann.ac.il/~naor
46. Naor, M., Reingold, O.: On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. J. Cryptology **12**(1), 29–66 (1999). https://doi.org/10.1007/PL00003817
47. Patarin, J.: The "Coefficients H" Technique. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC. LNCS, vol. 5381, pp. 328–345. Springer (2008). https://doi.org/10.1007/978-3-642-04159-4_21
48. Peyrin, T., Seurin, Y.: Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In: Robshaw, M., Katz, J. (eds.) CRYPTO I. LNCS, vol. 9814, pp. 33–63. Springer (2016). https://doi.org/10.1007/978-3-662-53018-4_2
49. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: ASIACRYPT. LNCS, vol. 3329, pp. 16–31. Springer (2004). https://doi.org/10.1007/978-3-540-30539-2_2
50. Rogaway, P., Zhang, Y.: Onion-AE: Foundations of Nested Encryption. PoPETs **2018**(2), 85–104 (2018). https://doi.org/10.1515/popets-2018-0014
51. Sarkar, P.: Improving Upon the TET Mode of Operation. In: Nam, K., Rhee, G. (eds.) ICISC. LNCS, vol. 4817, pp. 180–192. Springer (2007)
52. Sarkar, P.: Efficient Tweakable Enciphering Schemes from (Block-Wise) Universal Hash Functions. IEEE Trans. on Inf. Theory **55**(10), 4749–4760 (2009). https://doi.org/10.1109/TIT.2009.2027487
53. Sarkar, P.: Tweakable Enciphering Schemes Using Only the Encryption Function of a Block Cipher. Inf. Process. Lett. **111**(19), 945–955 (2011). https://doi.org/10.1016/j.ipl.2011.06.014

54. Shrimpton, T., Terashima, R.S.: A Modular Framework for Building Variable-Input-Length Tweakable Ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT (1). LNCS, vol. 8269, pp. 405–423. Springer (2013), https://doi.org/10.1007/978-3-642-42033-7_21
55. Wang, P., Feng, D., Wu, W.: HCTR: A Variable-Input-Length Enciphering Mode. In: Feng, D., Lin, D., Yung, M. (eds.) CISC. LNCS, vol. 3822, pp. 175–188. Springer (2005). https://doi.org/10.1007/11599548_15
56. Yasuda, K.: A New Variant of PMAC: Beyond the Birthday Bound. In: Rogaway, P. (ed.) CRYPTO. LNCS, vol. 6841, pp. 596–609. Springer (2011). https://doi.org/10.1007/978-3-642-22792-9_34
57. Zhang, L., Wu, W., Sui, H., Wang, P.: 3kf9: Enhancing 3GPP-MAC beyond the Birthday Bound. In: Wang, X., Sako, K. (eds.) ASIACRYPT. LNCS, vol. 7658, pp. 296–312. Springer (2012). https://doi.org/10.1007/978-3-642-34961-4_19

A Related Work

This section briefly revisits existing SPRPs that we categorized into (1) Generalized Feistel networks, (2) Encrypt-Mix-Encrypt, (3) Hash-ECB-Hash, (4) Hash-Counter-Hash, and (5) miscellaneous designs.

The original Naor-Reingold construction [46] is a four-round Feistel network with two layers of an ϵ-AXU family of hash functions that wrap two layers of encryption, which could be called the starting point of wide-block ciphers. Schemes based on the Encrypt-Mix-Encrypt paradigm (also Encrypt-Mask-Encrypt) combine two wrapping layers of encryption with an intermediate layer of linear mixing. There are at least six such constructions: CMC [22], EME and EME$^+$ [23], EME* [20], AEZ-Core [24, 25], and FMix [7]. Among them, EME* has become part of the P1619.2 standard for Wide-Block Encryption for Shared Storage Media [1]. The length doubler LDT by Chen et al. [15] could be also classified into this category, and could potentially provide even security beyond the birthday bound; however, its domain and range are limited to n up to $2n - 1$ bits.

Hash-ECB-Hash constructions consist of an encryption layer sandwiched by two hashing layers at top and bottom, a design principle that originated by Naor and Reingold [45]. Further examples include PEP [12], TET [21], or HEH [51].

Hash-Counter-Hash constructions resemble an unbalanced three-round Feistel network of two universal hash functions that wrap an encryption layer of counter mode. The first such construction was XCBv1 by McGrew and Viega that was later adapted, proven, and, as XCBv3, has become patented and part of the IEEE P1619.2 standard [1, 34, 35]. The name of the approach stems from HCTR by Wang et al. [55]. More constructions in this category include, e.g., HCH [13], HMC [42] (Hash Modified Counter), the LargeBlock1/2 constructions [39] and HSE [40]. In an ongoing series of works, Sarkar et al. [10, 18, 52] have been proposing various further versions, such as an improved HEH, and similar schemes with OFB (HOH), and counter mode (HMCH), respectively. In TES and its extensions STES and FAST [9, 11, 53], the authors later eliminated the need for the inverse operation of the block cipher.

A recent design that does not fully fit in the former categories is e.g., the two instantiations of Protected IV [54]. Protected IV is a modular framework that resembles the Ψ_3 construction by Coron et al. [17], i.e., a three-round unbalanced Feistel-like network based on tweakable ciphers. Its authors propsosed two instances of PIV, coined TCT_1 and TCT_2; the former with birthday-bound and the latter with $2n/3$-bit security, which stems from the use of a two-round Even-Mansour primitive.

The recent proposals HHFHFH [5] and Simpira (v2) [19] can be classified again as Feistel networks. Mr. Monster Burrito and HHFHFH are four-round unbalanced Feistel networks that were coined them heavyweight ciphers with high security guarantees due to the use of large-block permutations as primitives. Simpira is a family of variable-length ciphers based on the round function of the AES, where, family means that the Feistel design differs for small input sizes, and becomes general for inputs of eight blocks and above. In contrast to most designs, Simpira is based on the wide-trail heuristic where the authors bounded the number of rounds necessary to guarantee at least 25 active S-boxes and used three times this number of rounds.

B Analyses of badA, badB, badC

Analysis of badA. For $i \in E, j \in [1..\ell^i]$, $L_j'^i$ and $R_j'^i$ are sampled uniformly and independently from $\{0,1\}^n$. Thus, any collision over them occurs with a probability of $1/N$, and any disjoint pair of such collisions, being independent, jointly occurs with a probability $1/N^2$. The same reasoning holds for L_j^i and R_j^i, $i \in D, j \in [1..\ell^i]$.

Analysis of badB. Using (3)–(6), we can rewrite these collisions as:

- $(L_\ell^i + X_L^i, R_\ell^i + X_R^i) = (L_\ell^{i'} + X_L^{i'}, R_\ell^{i'} + X_R^{i'})$;
- $(S^i, R_\ell^i + X_R^i) = (S^{i'}, R_\ell^{i'} + X_R^{i'})$;
- $(S^i, T^i) = (S^{i'}, T^{i'})$;
- $(L_\ell'^i + Y_L^i, T^i) = (L_\ell'^{i'} + Y_L^{i'}, T^{i'})$;
- $(L_\ell'^i + Y_L^i, R_\ell'^i + Y_R^i) = (L_\ell'^{i'} + Y_L^{i'}, R_\ell'^{i'} + Y_R^{i'})$.

The main observation here is that when i is akin to i', $(L_\ell^i, R_\ell^i) \neq (L_\ell^{i'}, R_\ell^{i'})$, and $(L_\ell'^i, R_\ell'^i) \neq (L_\ell'^{i'}, R_\ell^{i'})$; furthermore, for $i \in E$, $(S^i, R_\ell^i) \neq (S^{i'}, R_\ell^{i'})$ and for $i \in D$, $(T^i, L_\ell'^i) \neq (T^{i'}, L_\ell'^{i'})$ (by construction of ideal oracle). This ensures that each of the collision pairs is either impossible or consists of four uniformly sampled random variables; thus, the reasoning for badA above holds for badB as well. The same reasoning carries over to badC.

C Full Analysis of badE

A part of this proof was described in Section 6. We present the rest of the proof here, skipping the sub-cases already examined before. Recall that $i' < i$ and $\ell \overset{\text{def}}{=}$

$\ell^{i'} = \ell^i$, and $\alpha_j(\cdot)$ and $\alpha_j^2(\cdot)$ are linear functions defined as $\alpha_j(x) =^{\text{def}} \alpha^{\ell-1-j} \cdot x$ and $\alpha_j^2(x) =^{\text{def}} (\alpha^2)^{\ell-1-j} \cdot x$. We look at the two main cases separately and branch into four subcases each:

Case 1: $(X_L^{*i}, X_R^{*i}) = (X_L^{*i'}, X_R^{*i'})$. We split into four subcases:

Subcase 1.1: $i \in E, i' \in E$. This was analysed in Section 6.

Subcase 1.2: $i \in E, i' \in D$. This was also analysed in Section 6.

Subcase 1.3: $i \in D, i' \in E$. From (7), (8), (21), and (22) we have

$$\sum_{j=0}^{\ell-1} \alpha_j(Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i:j}) = \sum_{j=0}^{\ell-1} \alpha_j(L_j'^{i:j} + L_j'^{i':j} + R_j^i + R_j^{i:j}), \tag{28}$$

$$\sum_{j=0}^{\ell-1} \alpha_j^2(Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i:j}) = \sum_{j=0}^{\ell-1} \alpha_j^2(L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}). \tag{29}$$

This case is straightforward: S_1^i does not get canceled out in either equation, and R_j^i remains only in (28), with no terms queried after it. Thus, the joint probability does not exceed $1/N^2$.

Subcase 1.4: $i \in D, i' \in D$. From (8), (21), and (22), we have

$$\begin{aligned} &\sum_{j=0}^{\ell-1} \alpha_j(Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i'} + S_1^{i:j} + S_1^{i':j}) \\ &= \sum_{j=0}^{\ell-1} \alpha_j(L_j'^{i:j} + L_j'^{i':j} + R_j^i + R_j^{i'} + R_j^{i:j} + R_j^{i':j}), \end{aligned} \tag{30}$$

$$\begin{aligned} &\sum_{j=0}^{\ell-1} \alpha_j^2(Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i'} + S_1^{i:j} + S_1^{i':j}) \\ &= \sum_{j=0}^{\ell-1} \alpha_j^2(L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}). \end{aligned} \tag{31}$$

The argument here is identical to that of the case above.

Case 2: $(Y_L^{*i}, Y_R^{*i}) = (Y_L^{*i'}, Y_R^{*i'})$. We can write this collision as

$$\sum_{j=0}^{\ell-1} \alpha_j^2(Y_j^i + Y_j^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j^2(L_j'^i + L_j'^{i'}) \quad \text{and} \quad \sum_{j=0}^{\ell-1} \alpha_j(Y_j^i + Y_j^{i'}) = 0.$$

Using (2) we can rewrite these as

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^i + Z_j^{i'} + S_1^i + S_1^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^i + L_j'^{i'} + R_j^i + R_j^{i'}), \tag{32}$$

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^i + Z_j^{i'} + S_1^i + S_1^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j (R_j^i + R_j^{i'}). \tag{33}$$

As before, we let j_0 and j_1 be the two largest indices where $Y_j^i + Y_j^{i'}$ does not trivially vanish, with $j_0 > j_1$. Again, we split into four subcases:

Subcase 2.1: $i \in E, i' \in E$. From (7), (32), and (33), we have

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}), \tag{34}$$

$$\begin{aligned}&\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i'})\\ =&\sum_{j=0}^{\ell-1} \alpha_j (L_j'^i + L_j'^{i'} + L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}).\end{aligned} \tag{35}$$

This is another straightforward case: S_1^i does not get canceled out in either equation, and $L_j'^i$ remains only in (35), with no terms queried after it. Thus, the joint probability does not exceed $1/N^2$.

Subcase 2.2: $i \in E, i' \in D$. From (7), (8), (32), and (33), we have

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}), \tag{36}$$

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j} + S_1^i + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j (L_j'^i + L_j'^{i:j} + R_j^{i:j} + R_j^{i':j}). \tag{37}$$

The argument here is identical to that of the case above.

Subcase 2.3: $i \in D, i' \in E$. From (7), (8), (32), and (33), we have

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j} + S_1^{i:j} + S_1^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L_j'^{i:j} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}), \tag{38}$$

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j} + S_1^{i:j} + S_1^{i'}) = \sum_{j=0}^{\ell-1} \alpha_j (L_j'^{i'} + L_j'^{i':j} + R_j^{i:j} + R_j^{i':j}). \tag{39}$$

By choice of j_0 and j_1, it holds that $i : j_0 \neq i'$ and $i : j_1 \neq i'$. Suppose $i : j_0 < i'$. Then, $S_1^{i'}$ and $R_{j_0}^{i'}$ remain uncanceled in (39), and no adversary query block queried after $L'^{i'}_{j_0}$ remains uncanceled; in (38), $S_1^{i'}$ remains uncanceled again, but there is no $L'^{i'}_{j_0}$ and no adversary query block queried after it. Thus, these two can occur jointly with a probability at most $1/N^2$.

A symmetric argument can be used when $i : j_0 > i'$ and $i : j_0 \in E$: we observe that $S_1^{i:j_0}$ remains uncanceled in either equation, while $L'^{i:j_0}_{j_0}$ remains uncanceled in (38), but gets canceled out in (39), and no adversary query block queried after it remains in either equation.

When $i : j_0 > i'$ and $i : j_0 \in D$, but $i : j_1$ satisfied one of the above two conditions, we can argue as above using $i : j_1$ instead. If we also have $i : j_1 > i'$ and $i : j_1 \in D$, $Z_{j_0}^{i:j_0}$ and $Z_{j_1}^{i:j_1}$ are basis elements that do not get canceled out in either equation. Their combined contribution to the left-hand side of (25) is $\alpha^{\ell-1-j_0} \cdot Z_{j_0}^{i:j_0} + \alpha^{\ell-1-j_1} \cdot Z_{j_1}^{i:j_1}$ and to the left-hand side of (26) is $(\alpha^2)^{\ell-1-j_0} \cdot Z_{j_0}^{i:j_0} + (\alpha^2)^{\ell-1-j_1} \cdot Z_{j_1}^{i:j_1}$. These two collisions are independent since $\alpha^{\ell-1-j_0} \cdot (\alpha^2)^{\ell-1-j_1} \neq \alpha^{\ell-1-j_1} \cdot (\alpha^2)^{\ell-1-j_0}$. Thus, they can occur with probability at most $1/N^2$.

Subcase 2.4: $i \in D, i' \in D$. From (8), (32), and (33), we have

$$\sum_{j=0}^{\ell-1} \alpha_j^2 (Z_j^{i:j} + Z_j^{i':j} + S_1^{i:j} + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j^2 (L'^{i:j}_j + L'^{i':j}_j + R_j^{i:j} + R_j^{i':j}), \quad (40)$$

$$\sum_{j=0}^{\ell-1} \alpha_j (Z_j^{i:j} + Z_j^{i':j} + S_1^{i:j} + S_1^{i':j}) = \sum_{j=0}^{\ell-1} \alpha_j (R_j^{i:j} + R_j^{i':j}). \quad (41)$$

By choice of j_0, it holds that $i : j_0 \neq i' : j_0$. Suppose that $i : j_0 > i' : j_0$. The basis element $S_1^{i:j_0}$ does not get canceled out in either equation; moreover, $L'^{i:j_0}_{j_0}$ remains only in (40), while it gets canceled out in (41); moreover, none of the adversary-queried blocks queried after it remains in either equation. Thus, the two collisions can occur jointly with a probability not exceeding $1/N^2$. The argument is symmetric when $i : j_0 < i' : j_0$.

D Insecure Preliminary Variants

It took some time to arrive at the present definition of ZCZ. This section examines four variants of this design which allow attacks, and which provide an implicit rationale of our current definition.

D.1 Basic Design

Our first attempts started with an EME- and HCTR-like structure with two wrapping layers of ZHash. This already implies to cluster the message into $2n$-bit di-blocks (or dual blocks). A chaining was necessary to achieve PRP security, i.e., every ciphertext bit must depend on every plaintext bit. For SPRP security, the opposite is also necessary, i.e., a chaining must also make every plaintext bit depend on every ciphertext bit. EME and AEZ provide an elegant solution where an X accumulator sums up the results X_i from the top encryption layer of the first $\ell - 1$ di-blocks (i.e., all but the final one):

$$X_i \leftarrow \widetilde{E}_K^{\mathsf{t},i,R_i}(L_i).$$

The accumulated value is then randomized and added it into the computation of the final di-block.

$$X \leftarrow \sum_{i=1}^{\ell-1-i} X_i, \qquad X_L \leftarrow \widetilde{E}_K^{\mathsf{x},0}(X), \qquad X_R \leftarrow \widetilde{E}_K^{\mathsf{x},1}(X).$$

The final di-block is processed then through the first layer of encryption and mixing in the middle. The outputs are then used to derive two n-bit values S and T:

$$S \leftarrow \widetilde{E}_K^{t\$,\ell,R_\ell+X_R}(L_\ell + X_L), \qquad T \leftarrow \widetilde{E}_K^{s\$,\ell,S}(R_\ell + X_R).$$

From S and T, a counter-mode layer derives chaining values Z_i, that are added to all previous di-blocks $1, \ldots, \ell - 1$:

$$Z_i \leftarrow \widetilde{E}_K \mathsf{c}, i, T(S) \qquad L'_i \leftarrow X_i + Z_i \qquad Y_i \leftarrow R_i + Z_i.$$

The result yields then the inputs Y_i, for $1 \leq i \leq \ell - 1$, to the bottom layer of encryption for the first $\ell - 1$ di-blocks:

$$R'_i \leftarrow \widetilde{E}_K^{\mathsf{b},i,L'_i}(Y_i).$$

The final di-block continues encryption of the bottom layer:

$$P_L \leftarrow \widetilde{E}_K^{\mathsf{c}\$,\ell,T}(S) \qquad P_R \leftarrow \widetilde{E}_K^{\mathsf{b}\$,\ell,P_L}(T).$$

The values Y_i are again accumulated to a chaining value Y

$$Y \leftarrow \sum_{i=1}^{\ell-1-i} Y_i, \qquad Y_L \leftarrow \widetilde{E}_K^{\mathsf{y},0}(Y), \qquad Y_R \leftarrow \widetilde{E}_K^{\mathsf{y},1}(Y),$$

that is used to mask the output of the final di-block, mirroring the top:

$$L'_\ell \leftarrow Y_L + P_L, \qquad R'_\ell \leftarrow Y_R + P_R.$$

This design yielded a basis for the later structure. However, the naive usage of ZHash in this approach opens several possible attack vectors:

– We had to compute additional values Z_i to prevent that an adversary could observe differences ΔY_i in the left branches of the ciphertext di-blocks $\Delta L'_i$.

– We introduced multiplications with powers of a primitive element in the computations of X_L and X_R to prevent that a collision in either would also lead to a collision in the other value (and similarly for Y_L and Y_R).
– Thereupon, we noticed that pure sums for X_L and X_R would be insufficient as the differences in ΔX_L and ΔX_R could be canceled by smart choice of ΔL_ℓ and ΔR_ℓ, and a randomization of X_L and X_R was required.
– Moreover, while multiplications with powers of α are quite fast, it is well-known since the days of EME [23, 25] that one can cancel differences in certain di-blocks that yield a zero sum if the multiplications of the same chaining value cover more than n di-blocks. Therefore, we had to introduce new chaining values similar to EME* [20].

To illustrate the relevance of each design aspect on its own, we describe in the following attacks on preliminary versions with respect to the final ZCZ construction, where only the particular element that we added as countermeasure would be missing.

D.2 Variant I: Omitting the Additions of S^i to The Right Branches

In this construction, the values Y_j^i are computed as $Y_j^i = R_j^i + Z_j^i$. Suppose that we make q queries such that query i has at least i di-blocks, and for each $i \in [1..q-1]$,

$$\left(L_i^i, R_i^i\right) = \left(L_i^{i+1}, R_i^{i+1}\right).$$

This ensures that $X_i^i = X_i^{i+1}$, so we can calculate

$$\Delta Y_i \overset{\text{def}}{=} Y_i^i + Y_i^{i+1} = L_i'^i + L_i'^{i+1}.$$

When q is chosen in the order of n, with high probability, we can find a zero-sum subset of $\left\{\alpha^{q-1-i} \cdot \Delta Y_i \mid i \in [1..q-1]\right\}$. Assume for some $\mathcal{I} \subseteq [1..q-1]$, we have

$$\sum_{i \in \mathcal{I}} \alpha^{q-1-i} \cdot \Delta Y_i = 0.$$

For details on how to efficiently identify such a subset, see e.g., Bellare and Micciancio [4]. Moreover, from $Y_i^i + Y_i^{i+1} = L_i'^i + L_i'^{i+1}$, it follows that $Y_i^i = L_i'^i = Y_i^{i+1} + L_i'^{i+1}$ for all i. Therefore, it holds that

$$\sum_{i \in \mathcal{I}} \alpha^{2(q-1-i)} \cdot \left(\Delta Y_i + \Delta L_i'^i\right) = 0.$$

For a distinguisher, we ask two decryption queries as follows: the first query consists of q blocks, with

$$\left(L_1'^1, R_1'^1\right), \left(L_2'^2, R_2'^2\right), \ldots, \left(L_{q-1}'^{q-1}, R_{q-1}'^{q-1}\right)$$

as the first $q-1$ di-blocks; in the second query, for each $i \in \mathcal{I}$, the i-th di-block is replaced by $(L_i'^{i+1}, R_i'^{i+1})$, and everywhere else, it is identical to the first query. These two queries lead to a collision in both $({Y^*}_L^1, {Y^*}_R^1) = ({Y^*}_L^2, {Y^*}_R^2)$, and hence will result in a collision in the right half of the final plaintext di-block $(L_\ell^1, R_\ell^1) = (L_\ell^2, R_\ell^2)$.

In contrast to this variant, the adversary cannot control differences in the values Y_i over different queries with a common prefix in ZCZ. There, the values S_i cannot efficiently be replicated over different queries. The unknown difference in the values S_i masks the difference in the values Y_i, rendering the attack ineffective.

D.3 Variant II: Omitting the Multiplications by Powers of α

As a consequence of the attack above, we also derived the values S^i and added them to the right branches of all di-blocks but the final one. Still, a number of further attack vectors remained. A clear point in the basic design was that a collision in X_L would automatically lead to a collision also in X_R. Subsequently, we added multiplications by powers of a primitive element α to one of them. Their relevance is outlined in the following attack.

In this variant, the values which means X_L^* and X_R^* are computed as

$$X_L^* = \sum_{i=1}^{\ell} X_i, \quad \text{and} \quad X_R^* = \sum_{i=1}^{\ell} X_i + R_i;$$

the values Y_L^* and Y_R^* are computed analogously. Clearly, one can query $q = 2^{n/2}$ queries which share equal di-blocks $R_j^i = R_j^k$ for all $1 \leq i < k \leq q$ and all j. This variant allows then a birthday distinguisher. Given $2^{n/2}$ queries, the probability is significant that there exist two messages for which ${X^*}_L^i = {X^*}_j^k$ holds. This event implies ${X^*}_R^i = {X^*}_R^k$ naturally.

D.4 Variant III: Omitting the Encryptions from (X_L^*, X_R^*) and (Y_L^*, Y_R^*) to (X_L, X_R) and (Y_L, Y_R)

Having introduced the second element Z_i for adding and the multiplications with powers of α, we already had a version that would be almost secure up to the birthday bound. However, we observed soon that an adversary could still choose the values $(\Delta L_\ell, \Delta R_\ell)$ well such that their differences would cancel those in ΔX_L and ΔX_R after about $2^{n/2}$ queries. In the following, we sketch an attack if we would omit the encryptions from from (X_L^*, X_R^*) and (Y_L^*, Y_R^*) to (X_L, X_R) and (Y_L, Y_R).

In this construction, it holds that

$$X_L = \sum_{j=1}^{\ell-1} \alpha^{\ell-1-j} X_j \quad \text{and} \quad X_R = \sum_{j=1}^{\ell-1} (\alpha^2)^{\ell-1-j} (X_j + R_j),$$

which means $X_L = X_L^*$, $X_R = X_R^*$, $Y_L = Y_L^*$, and $Y_R = Y_R^*$. This construction would allow a birthday distinguisher. Assume, we choose $q = 2^{n/2}$ messages that possess equal length ℓ and that consist of at least two di-blocks plus the final di-block each. The messages differ from each other in exactly three blocks: the final di-block (L_ℓ^i, R_ℓ^i) and some block L_j^i for some fixed index j. The final di-blocks are chosen so that it holds for every message:

$$R_\ell^i = \alpha^{2j-j} L_\ell^i = \alpha^j L_\ell^i.$$

If there exist two messages M^i and M^k for which $\Delta X_j = X_j^i + X_j^k = \Delta L_\ell = L_\ell^i + L_\ell^k$, then it holds that

$$\begin{aligned}
&\Delta X_L = \Delta L_\ell \text{ and} \\
&\Delta X_R = \alpha^{\ell-1-j} \Delta X_j = \Delta R_\ell.
\end{aligned}$$

For this pair of messages M^i and M^k, the values $S^i = S^k$ and $T^i = T^k$ will be identical, and we obtain collisions in the ciphertexts of all di-blocks whose plaintext di-blocks were equal between both messages. When choosing $2^{n/2}$ messages, the probability to obtain such a pair becomes significant. However, encrypting the sums X_L^*, X_R^*, Y_L^*, and Y_R^* effectively prevents such distinguishers.

D.5 Variant IV: Using S_1 for More Than n Di-Blocks

Finally, there is another attack if one doubles a single chaining value S for more than n times, as has already been observed by Halevi and Rogaway [23] on EME. Here, we discuss its impact on the preliminary version of ZCZ that would have only S_1.

For messages that consist of more than one chunk (i.e., a sequence of n non-final di-blocks), we have to derive a new chaining value S_i for every chunk. Otherwise, a similar attack as for Variant I would be possible also here. For this variant, the values Y_j^i would be computed as

$$Y_j^i = R_j^i + Z_j^i + S_1^i,$$

for all $1 \le j \le \ell - 1$, and all queries i. For our distinguisher, we can apply the same strategy as in Variant I, and ask q queries of same length ℓ, with $\ell > 2n+1$ di- blocks, such that the first $\ell - 1$ di-blocks are always equal over all queries:

$$(L_j^i, R_j^i) = (L_j^{i'}, R_j^{i'}), \quad \text{for all } 1 \le i \ne i' \le q, \quad 1 \le j \le \ell - 1.$$

The queries are pairwise distinct in their final di-block (L_ℓ, R_ℓ). As in the attack on Variant I, the adversary can observe the differences

$$\Delta Z_j^{i,i'} = \Delta Z_j^i + \Delta Z_j^{i'}$$

from the differences in the left branches of the di-blocks

$$\Delta {L'}_j^{i,i'} = \Delta {L'}_j^i + \Delta {L'}_j^{i'}.$$

Given $\ell > n+1$, there exist subsets of indices $\mathcal{J} \subseteq \{1, \ldots, n+1\}$ s.t. it holds:

$$\sum_{j \in \mathcal{J}} \alpha^{\ell-1-j} = 0.$$

Such sets must exist for this variant, e.g., for the case $n = 128$ and $\ell = 130$, and the primitive polynomial of GCM, one option would be $\mathcal{J} = \{0, 1, 2, 7, 128\}$. The concrete set of indices $\mathcal{J}$ depends on the chosen primitive polynomial. When j is in the order of $2n$, there are about 2^n choices (since we can multiply any polynomial of degree at most n to the primitive polynomial) of $\mathcal{J}$. For a second necessary condition, define $\Delta W_j^{i,i'} = \alpha^{\ell-1-j} \cdot \Delta Z_j^{i,i'}$. Then, the second condition is

$$\sum_{j \in \mathcal{J}} \Delta W_j^{i,i'} = 0.$$

Among the 2^n choices for $\mathcal{J}$, there exists one subset of indices that will also have this second condition fulfilled. To efficiently find a solution for both conditions, the adversary can define $2n$-bit values

$$\left(\Delta W_j^{i,i'} \parallel \alpha^{\ell-1-j}\right)$$

and find a subset $\{J\}'$ for which the sum yields 0^{2n} by solving linear equations, analogously to the approach in [4]. Once the adversary has found it, it holds that

$$\sum_{j \in \mathcal{J}'} \left(\alpha^{\ell-1-j} \cdot \Delta Y_j^{i,i'}\right) = \underbrace{\sum_{j \in \mathcal{J}'} \left(\alpha^{\ell-1-j} \cdot \Delta Z_j^{i,i'}\right)}_{=0} + \underbrace{\sum_{j \in \mathcal{J}'} \left(\alpha^{\ell-1-j} \cdot \Delta S_1^{i,i'}\right)}_{=0} = 0.$$

Note that the adversary can hold the values R_j^i equal over all queries $1 \leq i \leq q$. Since

$$\sum_{j \in \mathcal{J}} \alpha^{\ell-1-j} = 0 \quad \text{implies that} \quad \sum_{j \in \mathcal{J}} (\alpha^2)^{\ell-1-j} = 0,$$

the collision would affect always both X_L^* and X_R^* (or the corresponding values Y_L^* and Y_R^* if the attack uses decryption queries). So, the same attack as for Variant I would apply.

Naturally, the strategy of deriving a new chaining value for every chunk of n di-blocks helped protecting against it, as has already been used in EME* [20] and AEZ [25].

E Micro-optimizations of the Implementation.

Micro-optimization of the LFSR. The effort for updating the counter *ctr* through the schedule can be reduced slightly by storing eight (or up to 32 on

```
#define vshift_left_64(x, r)    _mm_slli_epi64(x, r)
#define vshift_right_64(x, r)   _mm_srli_epi64(x, r)
#define vshift_bytes_left(x, r) _mm_bslli_si128(x, r)
#define clmul(x, y, mask)       _mm_clmulepi64_si128(x, y, mask)
#define REDUCTION_POLYNOMIAL    _mm_set_epi32(0, 0, 0, 135)

__m128i gf_2_128_double_eight(__m128i hash, __m128i x[8]) {
    __m128i tmp[8];
    tmp[0] = vshift_right_64(hash, 56);
    tmp[1] = vshift_right_64(x[0], 57);
    tmp[2] = vshift_right_64(x[1], 58);
    tmp[3] = vshift_right_64(x[2], 59);
    tmp[4] = vshift_right_64(x[3], 60);
    tmp[5] = vshift_right_64(x[4], 61);
    tmp[6] = vshift_right_64(x[5], 62);
    tmp[7] = vshift_right_64(x[6], 63);

    __m128i sum = vxor_eight(tmp);  // sum = tmp[0] xor ... xor tmp[7]
    __m128i mod = clmul(sum, REDUCTION_POLYNOMIAL, 0x01);
    __m128i sum_low = vshift_bytes_left(sum, 8);

    tmp[0] = vshift_left_64(hash, 8);
    tmp[1] = vshift_left_64(x[0], 7);
    tmp[2] = vshift_left_64(x[1], 6);
    tmp[3] = vshift_left_64(x[2], 5);
    tmp[4] = vshift_left_64(x[3], 4);
    tmp[5] = vshift_left_64(x[4], 3);
    tmp[6] = vshift_left_64(x[5], 2);
    tmp[7] = vshift_left_64(x[6], 1);

    sum = vxor_eight(tmp);  // sum = tmp[0] xor ... xor tmp[7]
    // sum xor sum_low xor mod xor x[7]
    return vxor4(sum, sum_low, mod, x[7]);
}
```

256-bit registers) multiple subsequent values of the least-significant byte of the counter *ctr*. Then, one can process them at once in one AVX register. Moreover, we can observe that eight iterations of the LFSR can be parallelized almost fully. Given $\mathbb{F}_2^8 \ni x^r = (x_7 \,\|\, \ldots \,\|\, x_0)$, we can precompute three values and compute thereupon eight words in parallel with a single shift per word:

$$\begin{aligned} y &= (x^r \oplus (x^r \ll 2)) \wedge \texttt{0xFF}, \\ z &= y \oplus (y \gg 6), \\ w &= (x^r \ll 8) \,\|\, z, \\ x^{r+i} = \mathrm{LFSR}_2^i(x) &= (w \gg (8-i)) \wedge \texttt{0xFF}, \text{ for } 1 \le i \le 8. \end{aligned}$$

A similar procedure would be applicable for LFSR_3, but irrelevant for our purposes since we store the fixed secret part K in TK_3.

Micro-optimization of the Doublings. A potentially very old trick allows to parallelize the doublings (and similarly, the multiplications with four) in the middle layer. Consider X_L^* as an example in the following. Instead of updating $X_L^* = (\alpha \cdot X_L^*) \oplus X_i$ sequentially for $1 \le i < \ell$, we can multiply eight subsequent values $X_i, \ldots, X_{i+8}$ to compute the summands of the following in parallel:

$$X_L^* = \alpha^8 \cdot X_L^* + \alpha^7 \cdot X_1 + \cdots + \alpha \cdot X_7 + X_8.$$

We can then parallelize those by shifting each of the eight words $X_L^*, X_i, \ldots, X_7$ twice, as illustrated in the Listing above. Since $r(\mathtt{x}) = p(\mathtt{x}) + \mathtt{x}^{128} = \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x}^1 + \mathtt{1}$, a single multiplication for eight words suffices, and never exceeds the least significant 16 bits of V^H. For the multiplications with α^2, the shift amounts are adapted from $\ll i$ and $\gg 64 - i$ to $\ll 2i$ and $\gg 64 - (2 \cdot i)$, respectively.