ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication

Tetsu Iwata[1], Kazuhiko Minematsu[2], Thomas Peyrin[3,4,5], and Yannick Seurin[6]

[1] Nagoya University, Japan
tetsu.iwata@nagoya-u.jp
[2] NEC Corporation, Japan
k-minematsu@ah.jp.nec.com
[3] School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
[4] School of Computer Science and Engineering
Nanyang Technological University, Singapore
[5] Temasek Laboratories, Nanyang Technological University, Singapore
thomas.peyrin@ntu.edu.sg
[6] ANSSI, Paris, France
yannick.seurin@m4x.org

Abstract. We propose a new mode of operation called ZMAC allowing to construct a (stateless and deterministic) message authentication code (MAC) from a tweakable block cipher (TBC). When using a TBC with n-bit blocks and t-bit tweaks, our construction provides security (as a variable-input-length PRF) beyond the birthday bound with respect to the block-length n and allows to process $n + t$ bits of inputs per TBC call. In comparison, previous TBC-based modes such as PMAC1, the TBC-based generalization of the seminal PMAC mode (Black and Rogaway, EUROCRYPT 2002) or PMAC_TBC1k (Naito, ProvSec 2015) only process n bits of input per TBC call. Since an n-bit block, t-bit tweak TBC can process at most $n + t$ bits of input per call, the efficiency of our construction is essentially optimal, while achieving beyond-birthday-bound security. The ZMAC mode is fully parallelizable and can be directly instantiated with several concrete TBC proposals, such as Deoxys and SKINNY. We also use ZMAC to construct a stateless and deterministic Authenticated Encryption scheme called ZAE which is very efficient and secure beyond the birthday bound.

Keywords: MAC, tweakable block cipher, authenticated encryption

1 Introduction

Block Cipher-Based MACs. A Message Authentication Code (MAC) is a symmetric-key cryptographic function that ensures the authenticity of messages. A large family of MACs (such as CBC-MAC [BKR00] or OMAC [IK03]) are constructed as modes of operation of some underlying block cipher. They are

often provably secure and reasonably efficient, however, they also have inherent limitations with respect to speed and security. First, such modes cannot process more than n bits of input per block cipher call, where n is the block-length (in bits) of the underlying block cipher. Second, most block cipher-based modes are secure only up to the so-called birthday bound (i.e., up to $2^{n/2}$ message blocks), and very few proposals, such as PMAC_Plus [Yas11], achieve security *beyond the birthday bound* (BBB), often at the cost of efficiency. For block ciphers with block-length 128, birthday-bound security can be deemed to low in many situations.

For these reasons, a recent popular trend has been to design modes of operation for a stronger primitive, namely *tweakable* block ciphers (TBCs). In comparison to traditional block ciphers, TBCs take an extra t-bit input called the *tweak*, and should behave as a family of 2^t independent block ciphers indexed by the tweak. This primitive was formalized by Liskov *et al.* [LRW02] (even though the informal idea surfaced in several papers before), and turns out to be surprisingly flexible for building various cryptographic functionalities. A TBC can be either constructed in a generic way from a block cipher through a mode of operation such as XEX [Rog04], or as a dedicated design such as Threefish [FLS+10], SCREAM [GLS+14], Deoxys-BC [JNP14a], Joltik-BC [JNP14b], KIASU-BC [JNP14c], and SKINNY [BJK+16], these last four examples following the so-called TWEAKEY framework [JNP14d].

The first construction of a parallelizable[7] MAC from a TBC is PMAC1 [Rog04], derived from the block cipher-based construction PMAC [BR02] by abstracting the block cipher-based TBC implicitly used in PMAC. Assuming that the underlying TBC has n-bit blocks and t-bit tweaks, PMAC1 processes n bits of inputs per TBC call, handles messages of length up to (roughly) 2^t n-bit blocks, and is secure up to the birthday bound (i.e., up to roughly $2^{n/2}$ message blocks). This scheme is simple, efficient and fully parallelizable (all calls to the TBC except the final one can be made in parallel). For these reasons, it has been adopted for example by multiple TBC-based submissions to the CAESAR competition for Authenticated Encryption (AE), e.g. SCREAM [GLS+14], Deoxys [JNP14a], Joltik [JNP14b], or KIASU [JNP14c].

Several authors have proposed schemes that push security beyond the birthday bound. Naito [Nai15] proposed two constructions called PMAC_TBC1k and PMAC_TBC3k which are reminiscent from PMAC_Plus [Yas11]. As PMAC1, they allow to process only n bits of inputs per TBC call, but their security is significantly higher than for PMAC1: they are secure up to roughly 2^n message blocks. Recently, List and Nandi [LN17] proposed PMAC2x which extends the output size of Naito's PMAC_TBC1k scheme from n to $2n$ bits without harming efficiency nor security. (They also proposed a minor modification of PMAC_TBC1k with n-bit outputs called PMACx.) We remark that Minematsu and Iwata [MI17] recently reported severe flaws in [LN17] (the ePrint version of [LN17] was subsequently updated in order to fix these flaws).

[7] Liskov *et al.* [LRW02] suggested a MAC construction from a TBC called TBC-MAC, but the construction is serial.

Our Contribution. We propose a new TBC-based MAC called ZMAC. As PMAC_TBC1k [Nai15] or PMAC2x/PMACx [LN17], it achieves BBB-security (as a variable-input-length PRF) and it is fully parallelizable. However, our proposal is more efficient than any of the previous schemes. Specifically, ZMAC processes $n + t$ bits of inputs per TBC call when using an n-bit block and t-bit tweak TBC, whereas previous schemes are limited to n bits of inputs per TBC call, independently of the tweak size (see Table 1 for a comparison with existing schemes). To the best of our knowledge, this is the first TBC-based MAC that exploits the full power of the tweak input of the underlying TBC. Note that an n-bit block, t-bit tweak TBC cannot handle more than $n + t$ bits of public input per call, hence the efficiency of our construction is essentially optimal (a few tweak bits are reserved for domain separation but the impact is very limited). The tweak-length t of the TBC used in ZMAC can be arbitrary, which is important since existing dedicated TBCs have various tweak-length, smaller (e.g. Threefish or KIASU-BC) or larger (e.g. Deoxys-BC or SKINNY) than the block-length n.

Main Ideas of Our Design. Our construction follows the traditional "UHF-then-PRF" paradigm: first, the message is hashed with a universal hash function (UHF), and the resulting output is given to a fixed-input-length PRF. Building a BBB-secure fixed-input-length PRF from a TBC is more or less straightforward (one can simply use the "XOR of permutations" construction, which has been extensively analyzed [Luc00, Pat08, Pat13, CLP14]). The most innovative part of our work lies in the design of our TBC-based UHF, which we call ZHASH. The structure of our proposal is reminiscent of Naito's PMAC_TBC1k (and thus of PMAC_Plus) combined with the XTX tweak extension construction by Minematsu and Iwata [MI15]. We note that a TBC is often used to abstract a block cipher-based construction to simplify the security proof, for example in the case of PMAC and OCB [Rog04], where one can prove the security of TBC-based abstraction and the construction of TBC itself separately. The TBC-based abstraction eliminates the handling of masks, which simplifies the security proof. That is, it is often the case that TBC-based constructions do not have masks, where the masks are treated as tweaks. With ZMAC, we take the opposite direction to the common approach. We restore the masks in the construction, and our scheme explicitly relies on the use of masks together with a TBC.

Application to Deterministic Authenticated Encryption. Following List and Nandi [LN17], we use ZMAC to construct a (stateless) Deterministic Authenticated Encryption (DAE) scheme (i.e., a scheme whose security does not rely on the use of random IVs or nonces[8] [RS06]). The resulting scheme, called ZAE, is BBB-secure and very efficient: it processes on average $n(n + t)/(2n + t)$ input bits per TBC call (this complex form comes from the fact that the MAC, resp. encryption part processes $n + t$, resp. n input bits per TBC call). Note that when $t = 0$, this is (unsurprisingly) similar to standard double-pass block cipher-based DAE schemes ($n/2$ bits per block cipher call), but as t grows, efficiency

[8] DAE implies resistance against nonce-misuse by incorporating the nonce into the associated data, and thus is also called Misuse-Resistant AE (MRAE).

Table 1. Comparison of our designs ZMAC and ZAE with other MAC and DAE (a.k.a MRAE) schemes. Column "# bits per call" refers to the number of bits of input processed per primitive call. Notation: n is the block-length of the underlying BC/TBC, t is the tweak-length of the underlying TBC. NR denotes the nonce-respecting scenario.

Scheme	Prim.	# bits per call	Parallel	Security	Ref.
Message Authentication Code					
CMAC	BC	n	N	$n/2$	[IK03]
PMAC	BC	n	Y	$n/2$	[BR02]
SUM-ECBC	BC	$n/2$	N	$2n/3$	[Yas10]
PMAC_Plus	BC	n	Y	$2n/3$	[Yas11]
PMAC1	TBC	n	Y	$n/2$	[Rog04]
PMAC_TBC1k	TBC	n	Y	n	[Nai15]
PMACx/PMAC2x	TBC	n	Y	n	[LN17]
ZMAC	TBC	$n+t$	Y	$\min\{n,(n+t)/2\}$	Sec. 3
Deterministic Authenticated Encryption					
SIV	BC	$n/2$	Y	$n/2$	[RS06]
SCT	TBC	$n/2$	Y	$n/2$ (n for NR)	[PS16]
SIVx	TBC	$n/2$	Y	n	[LN17]
ZAE	TBC	$n(n+t)/(2n+t)$	Y	$\min\{n,(n+t)/2\}$	Sec. 5

approaches n bits per TBC calls, i.e., the efficiency of an *online* block cipher-based scheme (which cannot be secure in the DAE sense). We provide a comparison with other DAE schemes in Table 1. We emphasize that ZAE is a mere combination of ZMAC with a TBC-based encryption mode called IVCTRT previously proposed in [PS16] through the SIV composition method [RS06]. Nevertheless, we think the proposal of a concrete DAE scheme based on ZMAC is quite relevant here, and helps further illustrate the performance gains allowed by ZMAC (see Table 3 in Section 6).

Future Works. ZMAC achieves optimal efficiency while providing full n-bit security (assuming $t \geq n$). For this reason, it seems that this mode cannot be substantially improved. However, it would be very interesting to study how ZMAC's design can influence ad-hoc TBC constructions: if one could construct an efficient, BBB-secure n-bit block TBC with a very large tweak (something which has not been studied much yet), this would lead to extremely efficient MAC algorithms.

Organization. We give useful definitions in Section 2. Our new mode ZMAC is defined in Section 3, and its security is analyzed in Section 4. Applications to Authenticated Encryption are presented in Section 5. Finally, a performance estimation for ZMAC and ZAE when Deoxys-BC or SKINNY are used to instantiate the TBC is provided in Section 6.

2 Preliminaries

Basic Notation. Let $\{0,1\}^*$ be the set of all finite bit strings. For an integer $n \geq 0$, let $\{0,1\}^n$ be the set of all bit strings of length n, and $(\{0,1\}^n)^+$ be the set of all bit strings of length a (non-zero) positive multiple of n. For $X \in \{0,1\}^*$, $|X|$ is its length in bits, and for $n \geq 1$, $|X|_n = \lceil |X|/n \rceil$ is its length in n-bit blocks. The string of n zeros is denoted 0^n. The concatenation of two bit strings X and Y is written $X \,\|\, Y$, or XY when no confusion is possible. For any $X \in \{0,1\}^n$ and $i \leq n$, let $\mathtt{msb}_i(X)$, resp. $\mathtt{lsb}_i(X)$ be the first, resp. last i bits of X. For non-negative integers a and d with $a \leq 2^d - 1$, let $\mathtt{str}_d(a)$ be the d-bit binary representation of a.

Given a bit string $X \in \{0,1\}^{i+j}$, we write

$$(X[1], X[2]) \xleftarrow{i,j} X$$

where $X[1] = \mathtt{msb}_i(X)$ and $X[2] = \mathtt{lsb}_j(X)$. For $X \in \{0,1\}^*$, we also define the parsing into fixed-length subsequences of length n, denoted

$$(X[1], X[2], \ldots, X[m]) \xleftarrow{n} X,$$

where $m = |X|_n$, $X[1] \,\|\, X[2] \,\|\, \ldots \,\|\, X[m] = X$, $|X[i]| = n$ for $1 \leq i < m$ and $0 \leq |X[m]| \leq n$ when $|X| > 0$. When $|X| = 0$, we let $X[1] \xleftarrow{n} X$, where $X[1]$ is the empty string.

Let n and t be positive integers. For any $X \in \{0,1\}^*$, we define the "one-zero padding" $\mathtt{ozp}(X)$ to be X if $|X|$ is a positive multiple of $(n+t)$ and $X \,\|\, 10^c$ for $c = |X| \bmod (n+t) - 1$ otherwise. We stress that $\mathtt{ozp}(\cdot)$ is defined with respect to $(n+t)$-bit blocks rather than n-bit blocks, and that the empty string is padded to 10^{n+t-1}.

For any $X \in \{0,1\}^n$ and $Y \in \{0,1\}^t$, we define

$$X \oplus_t Y \stackrel{\text{def}}{=} \begin{cases} \mathtt{msb}_t(X) \oplus Y & \text{if } t \leq n, \\ (X \,\|\, 0^{t-n}) \oplus Y & \text{if } t > n. \end{cases}$$

Hence, $|X \oplus_t Y| = t$ in both cases and if $t = n$ then $X \oplus_t Y = X \oplus Y$.

Given a non-empty set $\mathcal{X}$, we let $X \xleftarrow{\$} \mathcal{X}$ denote the draw of an element X uniformly at random in $\mathcal{X}$.

Galois Field. An element a in the Galois field $\mathrm{GF}(2^n)$ will be interchangeably represented as an n-bit string $a_{n-1} \ldots a_1 a_0$, a formal polynomial $a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$, or an integer $\sum_{i=0}^{n-1} a_i 2^i$. Hence, by writing $2 \cdot a$ or $2a$ when no confusion is possible, we mean the multiplication of a by $2 = \mathtt{x}$. This operation is called *doubling*. For $n = 128$, we define the field $\mathrm{GF}(2^n)$ (as is standard) by the primitive polynomial $\mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. The doubling $2a$ over this field is $(a \ll 1)$ if $\mathtt{msb}_1(a) = 0$ and $(a \ll 1) \oplus (0^{120}10000111)$ if $\mathtt{msb}_1(a) = 1$, where $(a \ll 1)$ denotes the left-shift of a by one bit.

KEYED FUNCTIONS AND MODES. A keyed function with key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$. We write $F_K(X)$ for $F(K, X)$. If Mode is a mode of operation for F using a single key $K \in \mathcal{K}$ for F, we write $\mathsf{Mode}[F_K]$ instead of $\mathsf{Mode}[F]_K$.

For any keyed function $F : \mathcal{K} \times (\{0,1\}^n)^+ \to \{0,1\}^a$ for some a, we define the collision probability of F as

$$\mathsf{Coll}_F(n, m, m') \stackrel{\text{def}}{=} \max_{\substack{M \in (\{0,1\}^n)^m \\ M' \in (\{0,1\}^n)^{m'} \\ M \neq M'}} \Pr[K \xleftarrow{\$} \mathcal{K} : F_K(M) = F_K(M')].$$

TWEAKABLE BLOCKCIPHERS. A tweakable blockcipher (TBC) is a keyed function $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ such that for each $(K, T) \in \mathcal{K} \times \mathcal{T}$, $\widetilde{E}(K, T, \cdot)$ is a permutation over $\mathcal{M}$. Here, K is the key and T is a public value called tweak. Note that a conventional block cipher is a TBC such that the tweak space $\mathcal{T}$ is a singleton. The output $\widetilde{E}(K, T, X)$ of the encryption of $X \in \mathcal{M}$ under key $K \in \mathcal{K}$ and tweak $T \in \mathcal{T}$ may also be written $\widetilde{E}_K(T, X)$ or $\widetilde{E}_K^T(X)$. Following [PS16], when the tweak space of $\widetilde{E}$ is $\mathcal{T}_I = \mathcal{T} \times \mathcal{I}$ for some $\mathcal{I} \subset \mathbb{N}$ and for some set $\mathcal{T}$, we call $\mathcal{T}$ the *effective* tweak space of $\widetilde{E}$, and we write $\widetilde{E}^i(K, T, X)$ to mean $\widetilde{E}(K, (T, i), X)$. By convention we also write $\widetilde{E}_K^i(T, X)$ or $\widetilde{E}_K^{i,T}(X)$. The set $\mathcal{I}$ is typically a small set used to generate a small number of distinct TBC instances in the scheme, something we call domain separation. For $T' = (T, i) \in \mathcal{T}_I$, we call $i \in \mathcal{I}$ the domain separation integer of tweak T'.

RANDOM PRIMITIVES. Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{T}$ be non-empty finite sets. Let $\mathrm{Func}(\mathcal{X}, \mathcal{Y})$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, and let $\mathrm{Perm}(\mathcal{X})$ be the set of all permutations over $\mathcal{X}$. Moreover, let $\mathrm{Perm}^{\mathcal{T}}(\mathcal{X})$ be the set of all functions $f : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ such that for any $T \in \mathcal{T}$, $f(T, \cdot)$ is a permutation over $\mathcal{X}$.

A uniform random function (URF) with domain $\mathcal{X}$ and range $\mathcal{Y}$, denoted $\mathsf{R} : \mathcal{X} \to \mathcal{Y}$, is a random function with uniform distribution over $\mathrm{Func}(\mathcal{X}, \mathcal{Y})$. Similarly, a uniform random permutation (URP) over $\mathcal{X}$, denoted $\mathsf{P} : \mathcal{X} \to \mathcal{X}$, is a random permutation with uniform distribution over $\mathrm{Perm}(\mathcal{X})$. An n-bit URP is a URP over $\{0,1\}^n$. Finally, a tweakable URP (TURP) with tweak space $\mathcal{T}$ and message space $\mathcal{X}$, denoted $\widetilde{\mathsf{P}} : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$, is a random tweakable permutation with uniform distribution over $\mathrm{Perm}^{\mathcal{T}}(\mathcal{X})$.

SECURITY NOTIONS. We recall standard security notions for (tweakable) block ciphers and keyed functions.

Definition 1. *Let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ be a TBC, and let $\mathcal{A}$ be an adversary with oracle access to a tweakable permutation whose goal is to distinguish $\widetilde{E}$ and a TURP $\widetilde{\mathsf{P}} : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ by oracle access. The advantage of $\mathcal{A}$ against the Tweakable Pseudorandom Permutation-security (or TPRP-security) of $\widetilde{E}$ is defined as*

$$\mathtt{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{A}) \stackrel{\text{def}}{=} \left| \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\widetilde{E}_K} \Rightarrow 1] - \Pr[\widetilde{\mathsf{P}} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{X}) : \mathcal{A}^{\widetilde{\mathsf{P}}} \Rightarrow 1] \right|,$$

where $\mathcal{A}^{\widetilde{E}_K} \Rightarrow 1$ *denotes the event that the final binary decision by* $\mathcal{A}$ *is 1.*

We remark that the above definition only allows $\mathcal{A}$ to make encryption queries. If decryption queries are allowed, the corresponding notion is called Strong TPRP (or STPRP) security. In this paper, we only use TPRP-security for the TBC underlying our constructions. The standard PRP-security notion for conventional block ciphers is recovered by letting the tweak space $\mathcal{T}$ be a singleton.

Definition 2. *For* $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$*, let* $\mathcal{A}$ *be an adversary whose goal is to distinguish* F_K *and a URF* $\mathsf{R} : \mathcal{X} \to \mathcal{Y}$ *by oracle access. The advantage of* $\mathcal{A}$ *against the PRF-security of* F *is defined as*

$$\mathtt{Adv}^{\mathtt{prf}}_{F}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[\mathsf{R} \xleftarrow{\$} \mathrm{Func}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{\mathsf{R}} \Rightarrow 1]\right|.$$

Moreover, for any $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *and* $G : \mathcal{K}' \times \mathcal{X} \to \mathcal{Y}$*, the advantage of* $\mathcal{A}$ *in distinguishing* F *and* G *is defined as*

$$\mathtt{Adv}^{\mathtt{dist}}_{F,G}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{G_{K'}} \Rightarrow 1]\right|.$$

When a cryptographic scheme (or a mode of operation) Mode uses a (T)BC of block-length n bits, the security bound (i.e., the best advantage for any adversary with fixed resources) is typically a function of the query complexity of the adversary (in terms of number q of queries or total number σ of queried blocks) and n. When this function reaches 1 for query complexity $2^{n/2}$, we say that Mode is secure up to the birthday bound, since this typically arises from the birthday paradox on the block input of the (T)BC. Conversely, if the advantage is negligibly small for any adversary of query complexity $2^{n/2}$, we say that Mode is secure beyond the birthday bound (BBB-secure).

3 Specification of ZMAC

3.1 Overview

Let $\widetilde{E} : \mathcal{K} \times \mathcal{T}_I \times \{0,1\}^n \to \{0,1\}^n$ be a TBC with tweak space $\mathcal{T}_I = \mathcal{T} \times \mathcal{I}$, where $\mathcal{T} = \{0,1\}^t$ for some $t > 0$ and $\mathcal{I} \supseteq \{0, 1, \ldots, 9\}$. We present a construction of a PRF $\mathsf{ZMAC}[\widetilde{E}] : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^{2n}$ with variable-input-length and $2n$-bit outputs based on $\widetilde{E}$.

The ZMAC mode has the following properties, holding for any effective tweak size $t > 0$:

1. it uses a single key for calls to $\widetilde{E}$;
2. the calls to $\widetilde{E}$ are parallelizable;
3. it processes on average $n + t$ input bits per TBC call;
4. it is provably secure as long as the total length σ of queries in $(n + t)$-bit blocks is small compared with $2^{\min\{n,(n+t)/2\}}$.

ZMAC is more efficient that any previous TBC-based MAC, which process at most n bits per TBC call (e.g., when $t = n$, ZMAC is twice faster than PMAC1). We emphasize that any mode based on an n-bit block, t-bit tweak TBC can process at most $n+t$ input bits per TBC call, thus ZMAC's efficiency is essentially optimal if one wants to achieve any meaningful provable security, since otherwise there must be some part of the input which is not processed by the TBC.[9]

Property 4 shows that the security of ZMAC is beyond the birthday bound with respect to n. In particular, it is n-bit secure when $t \geq n$. These properties demonstrate that ZMAC is the first TBC-based MAC to fully use the power of the underlying TBC.

We specify ZMAC with $2n$-bit outputs, which will be useful for defining our BBB-secure DAE scheme in Section 5. However, if one simply wants an n-bit-secure MAC, one can truncate the output of ZMAC to n bits (which saves two TBC calls in the finalization).

Design Rationale. The structure of ZMAC has some similarities with previous BBB-secure TBC-based PRF constructions [Nai15, LN17]. However, there are several innovative features that make ZMAC faster and n-bit secure.

The core idea of [Nai15, LN17] is to start from a TBC-based instantiation of PHASH, the UHF underlying PMAC [Rog04]. PHASH is quite simple: it simply XORs together the encryptions $\widetilde{E}_K(i, M_i)$ of message blocks with the index i of the block as tweak. In order to obtain a $2n$-bit output, some linear layer is applied to all encrypted blocks, as originally introduced by Yasuda [Yas11] in his PMAC_Plus block cipher-based PRF. This yields a $2n$-bit message hash, to which some finalization function (a fixed-input-length PRF) is applied to obtain the final output.

Whereas the t-bit tweak in the previous schemes takes as input the index of each message block, we crucially use both the message space and the tweak space of the TBC to process $n + t$ input bits in order to improve efficiency. The block index is incorporated via (a variant of) a tweak extension scheme called XTX [MI15], which allows to efficiently update the block index with only two field doublings, somehow similarly to XEX [Rog04].

The above trick, however, is not enough to achieve BBB-security. Since we process each $(n+t)$-bit input block by one call to an n-bit output TBC, the input block and the output block are no longer in one-to-one correspondence. Yet the BBB-security of previous schemes (where each input block is n-bit) crucially relies on this fact (otherwise, one can find a collision with complexity $2^{n/2}$, resulting in $n/2$-bit security). Fortunately, this problem can be solved by processing each $(n + t)$-bit input block with a Feistel-like permutation involving one TBC call, and applying the linear layer to the output of this $(n + t)$-bit permutation.

High-Level Structure of ZMAC. ZMAC consists of a hashing part

$$\mathsf{ZHASH}[\widetilde{E}] : \mathcal{K} \times (\{0,1\}^{n+t})^+ \to \{0,1\}^{n+t}$$

[9] Alternatively, one can combine another large non-linear component such as a field multiplication with an extra key, however this increases the implementation size.

Algorithm $\mathsf{ZHASH}[\widetilde{E}_K](X)$

1. $U \leftarrow 0^n$, $V \leftarrow 0^t$
2. $L_\ell \leftarrow \widetilde{E}^9_K(0^t, 0^n)$
3. $L_r \leftarrow \widetilde{E}^9_K(0^{t-1}1, 0^n)$
4. $(X[1], \ldots, X[m]) \xleftarrow{n+t} X$
5. **for** $i = 1$ **to** m **do**
6. $\quad (X_\ell, X_r) \xleftarrow{n,t} X[i]$
7. $\quad S_\ell \leftarrow L_\ell \oplus X_\ell$
8. $\quad S_r \leftarrow L_r \oplus_t X_r$
9. $\quad C_\ell \leftarrow \widetilde{E}^8_K(S_r, S_\ell)$
10. $\quad C_r \leftarrow C_\ell \oplus_t X_r$
11. $\quad U \leftarrow 2(U \oplus C_\ell)$
12. $\quad V \leftarrow V \oplus C_r$
13. $\quad (L_\ell, L_r) \leftarrow (2L_\ell, 2L_r)$
14. **return** (U, V)

Algorithm $\mathsf{ZFIN}[\widetilde{E}_K](i, U, V)$

1. $Y[1] \leftarrow \widetilde{E}^i_K(V, U) \oplus \widetilde{E}^{i+1}_K(V, U)$
2. $Y[2] \leftarrow \widetilde{E}^{i+2}_K(V, U) \oplus \widetilde{E}^{i+3}_K(V, U)$
3. $Y \leftarrow Y[1] \,\|\, Y[2]$
4. **return** Y

Algorithm $\mathsf{ZMAC}[\widetilde{E}_K](M)$

1. $X \leftarrow \mathtt{ozp}(M)$
2. $(U, V) \leftarrow \mathsf{ZHASH}[\widetilde{E}_K](X)$
3. **if** $M \in (\{0,1\}^{n+t})^+$
4. $\quad Y \leftarrow \mathsf{ZFIN}[\widetilde{E}_K](0, U, V)$
5. **else**
6. $\quad Y \leftarrow \mathsf{ZFIN}[\widetilde{E}_K](4, U, V)$
7. **return** Y

Fig. 1. Specification of ZMAC.

and a finalization part

$$\mathsf{ZFIN}[\widetilde{E}] : \mathcal{K} \times \{0,1\}^{n+t} \to \{0,1\}^{2n}.$$

Then, ZMAC is defined as the composition of ZHASH and ZFIN. When the input-length is not a positive multiple of $(n+t)$ bits, one-zero padding (into $(n+t)$-bit blocks) is applied first. To separate inputs whose length is a positive multiple of $(n+t)$ bits or not, we use distinct domain separation integers in ZFIN.

The pseudocode for ZHASH, ZFIN, and ZMAC is shown in Figure 1. It gives a unified specification that covers both cases $t \leq n$ and $t > n$ (note that the only operation which differs in the two cases is the $\oplus_t$ operation). We describe more informally ZHASH separately for $t \leq n$ and $t > n$, as well as ZFIN in the following sections.

3.2 Specification of ZHASH for the Case $t \leq n$

We first define $\mathsf{ZHASH}[\widetilde{E}]$ when $t \leq n$. For simplicity, we assume $n + t$ is even. Before processing the input, $\mathsf{ZHASH}[\widetilde{E}]$ computes two n-bit initial mask values $L_\ell = \widetilde{E}^9_K(0^t, 0^n)$ and $L_r = \widetilde{E}^9_K(0^{t-1}1, 0^n)$.

Given input $X \in (\{0,1\}^{n+t})^+$, $\mathsf{ZHASH}[\widetilde{E}]$ parses X into $(n+t)$-bit blocks $(X[1], \ldots, X[m])$, parses each block $X[i]$ as $X_\ell[i] = \mathtt{msb}_n(X[i])$ and $X_r[i] = \mathtt{lsb}_t(X[i])$, and computes, for $i = 1$ to m,

$$C_\ell[i] = \widetilde{E}^8_K(2^{i-1}L_r \oplus_t X_r[i], 2^{i-1}L_\ell \oplus X_\ell[i]), \tag{1}$$
$$C_r[i] = C_\ell[i] \oplus_t X_r[i]. \tag{2}$$

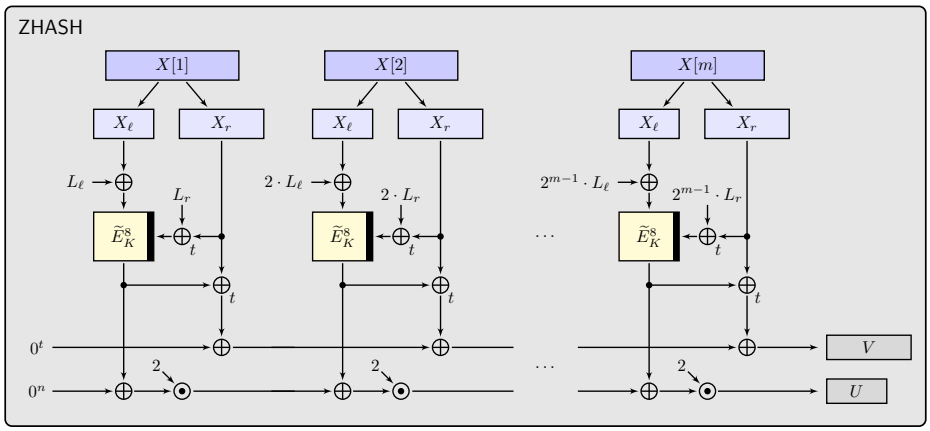

Fig. 2. The ZHASH hash function.

Then $\mathsf{ZHASH}[\widetilde{E}]$ computes two chaining values, $U \in \{0,1\}^n$ and $V \in \{0,1\}^t$ defined as

$$U = \bigoplus_{i=1}^{m} 2^{m-i+1} C_\ell[i],$$
$$V = \bigoplus_{i=1}^{m} C_r[i].$$

The final output is (U, V).

As shown in Figure 1, the field doublings are computed in an incremental manner. Specifically, $\mathsf{ZHASH}[\widetilde{E}]$ needs one call to $\widetilde{E}$ and three $\mathrm{GF}(2^n)$ doublings to process an $(n+t)$-bit block, plus two pre-processing calls to $\widetilde{E}$. Obviously, the calls to $\widetilde{E}$ are parallelizable.

3.3 Specification of ZHASH for the Case $t > n$

The hashing scheme $\mathsf{ZHASH}[\widetilde{E}]$ for the case $t > n$ is defined as follows (the two internal masks L_ℓ and L_r are derived and incremented in the same way as in the case $t \leq n$).

- The input X is parsed into $(n+t)$-bit blocks as in the case $t \leq n$, and each block is further parsed into n, n, and $t-n$ bit-blocks;
- The first and second n-bit sub-blocks are processed in the same way as in the case $t = n$. The third $(t-n)$-bit sub-block is directly fed to the tweak input of the TBC as the last $(t-n)$ bits of effective tweak;
- The output consists of two checksums, $U \in \{0,1\}^n$ and $V \in \{0,1\}^t$, where $(U, \mathtt{msb}_n(V))$ corresponds to the output for the case $t = n$, and $\mathtt{lsb}_{t-n}(V)$ corresponds to the sum of all third $(t-n)$-bit sub-blocks.

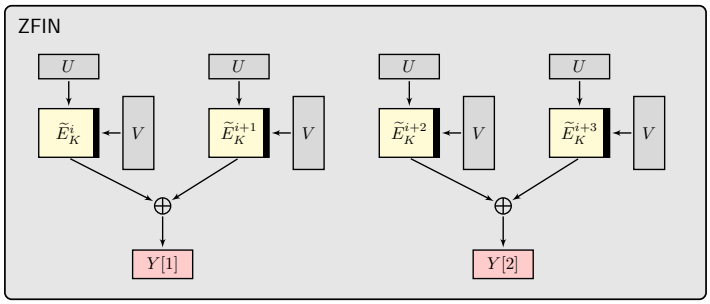

Fig. 3. The ZFIN finalization function.

Hence, the computation of V is just written as the sum of all C_r blocks in the unified specification of Figure 1, since the last $(t-n)$ bits of $C_r[i]$ only contains the last $(t-n)$ bits of the input block $X[i]$.

3.4 Finalization

The finalization function, denoted by $\mathsf{ZFIN}[\widetilde{E}]$, takes the output of $\mathsf{ZHASH}[\widetilde{E}]$, $(U, V) \in \{0,1\}^n \times \{0,1\}^t$, and generates a $2n$-bit output. It is defined as

$$\mathsf{ZFIN}[\widetilde{E}_K](i, U, V) = (\widetilde{E}_K^i(U, V) \oplus \widetilde{E}_K^{i+1}(U, V) \,\|\, \widetilde{E}_K^{i+2}(U, V) \oplus \widetilde{E}_K^{i+3}(U, V)),$$

where the first argument i is a non-negative integer used for domain separation. Note that if $|i - j| \geq 4$, domain separation integers used for TBC calls in $\mathsf{ZFIN}[\widetilde{E}_K](i, \cdot, \cdot)$ and in $\mathsf{ZFIN}[\widetilde{E}_K](j, \cdot, \cdot)$ are distinct. We use $i = 0$ when no padding is applied, i.e., when $M \in (\{0,1\}^{n+t})^+$, and $i = 4$ otherwise.

We remark that ZFIN is close but not identical to finalization functions used in previous works [Nai15, LN17]. For example, Naito [Nai15] employed $\widetilde{E}_K^i(U, V) \oplus \widetilde{E}_K^{i+1}(V, U)$ for building a PRF with n-bit outputs. One potential advantage of ZFIN over using two independent instances of Naito's construction is that ZFIN can be faster if the algorithm of $\widetilde{E}$ allows to leverage on the similarity of inputs for computing $\widetilde{E}_K^i(U, V)$ and $\widetilde{E}_K^{i+1}(U, V)$.

4 The PRF Security of ZMAC

4.1 XT Tweak Extension

Our first step is to recast the use of masks $2^{i-1}L_\ell$ and $2^{i-1}L_r$ as a way to extend the tweak space of $\widetilde{E}$. More specifically, we observe that the "core" construction of ZHASH in Eq. (1),

$$((T, i), X) \mapsto \widetilde{E}_K^8(2^{i-1}L_r \oplus_t T, 2^{i-1}L_\ell \oplus X), \tag{3}$$

keyed by $(K, (L_\ell, L_r))$, is an instantiation of a CPA-secure variant of a tweak extension scheme called XTX proposed in [MI15], which allows to extend the tweak space of $\widetilde{E}^8$ from $\mathcal{T} = \{0,1\}^t$ to $\mathcal{T}_J = \mathcal{T} \times \mathcal{J}$ with $\mathcal{J} = \{1, \ldots, 2^n - 1\}$. Following the naming convention for XE and XEX by Rogaway [Rog04] which defines CPA- and CCA-secure TBCs based on a block cipher, we use XT to denote the CPA-secure variant of XTX without output mask.

In order to describe the XT construction, we need the notion of partial AXU hash function introduced by [MI15].

Definition 3. *Let $H : \mathcal{L} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function with key space $\mathcal{L}$, domain $\mathcal{X}$, and range $\mathcal{Y} = \{0,1\}^n \times \{0,1\}^t$. We say that H is (n, t, ϵ)-partial almost-XOR-universal ((n, t, ϵ)-pAXU) if for any $X \neq X'$, one has*

$$\max_{\delta \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \mathcal{L} : H_L(X) \oplus H_L(X') = (\delta, 0^t)] \leq \epsilon.$$

Now define the XT tweak extension scheme. Let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a TBC with tweak space $\mathcal{T} = \{0,1\}^t$ and let $H : \mathcal{L} \times \mathcal{T}' \to \mathcal{Y}$ be a keyed function with range $\mathcal{Y} = \{0,1\}^n \times \{0,1\}^t$. Let $\mathsf{XT}[\widetilde{E}, H]$ be the TBC with key space $\mathcal{K} \times \mathcal{L}$, tweak space $\mathcal{T}'$, and message space $\{0,1\}^n$ defined as

$$\mathsf{XT}[\widetilde{E}, H]_{K,L}(T', X) = \widetilde{E}_K(Z_r, Z_\ell \oplus X) \text{ where } H_L(T') = (Z_\ell, Z_r). \tag{4}$$

The following lemma characterizes the security of $\mathsf{XT}[\widetilde{\mathsf{P}}, H]$ where $\widetilde{E}$ is replaced by a TURP $\widetilde{\mathsf{P}}$. It is similar to [MI15, Theorem 1] and its proof is deferred to the full version of the paper.

Lemma 1. *Let $\mathsf{XT}[\widetilde{\mathsf{P}}, H]$ be defined as above, where $\widetilde{\mathsf{P}} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a TURP and H is (n, t, ϵ)-pAXU. Then, for any adversary $\mathcal{A}$ making at most q queries, one has*

$$\mathtt{Adv}^{\mathtt{tprp}}_{\mathsf{XT}[\widetilde{\mathsf{P}}, H]}(\mathcal{A}) \leq \frac{q^2 \epsilon}{2}.$$

Assume for a moment that $L_\ell = \widetilde{E}^9_K(0^t, 0^n)$ and $L_r = \widetilde{E}^9_K(0^{t-1}1, 0^n)$ are uniformly random (this will hold once the TBC underlying ZMAC has been replaced by a TURP later in the security proof). Consider the function H with key space $\{0,1\}^n \times \{0,1\}^n$, domain $\mathcal{T}_J = \mathcal{T} \times \mathcal{J}$ with $\mathcal{J} = \{1, \ldots, 2^n - 1\}$, and range $\{0,1\}^n \times \{0,1\}^t$ defined as

$$H_{(L_\ell, L_r)}(T, i) = (2^{i-1} L_\ell, 2^{i-1} L_r \oplus_t T). \tag{5}$$

Then observe that the construction of Eq. (3) is exactly $\mathsf{XT}[\widetilde{E}^8, H]$ with H defined as above. We prove that H is pAXU in the following lemma.

Lemma 2. *Let H be defined as in Eq. (5). Then H is $(n, t, 1/2^{n+\min\{n,t\}})$-pAXU.*

Proof. Assume first that $t \le n$. Then, by definition of $\oplus_t$, one has

$$H_{(L_\ell,L_r)}(T,i) = (2^{i-1}L_\ell, \mathtt{msb}_t(2^{i-1}L_r) \oplus T).$$

Hence, we must upper bound

$$p \stackrel{\text{def}}{=} \Pr_{(L_\ell,L_r)}\left[\left((2^{i-1}+2^{j-1})L_\ell, \mathtt{msb}_t((2^{i-1}+2^{j-1})L_r) \oplus T \oplus T'\right) = (\delta, 0^t)\right]$$

for any distinct inputs $(T,i),(T',j) \in \mathcal{T}_J$ and any $\delta \in \{0,1\}^n$.

If $i = j$, then necessarily $T \neq T'$, and hence

$$\mathtt{msb}_t((2^{i-1}+2^{j-1})L_r) \oplus T \oplus T' = T \oplus T' \neq 0^t.$$

Thus the probability p is zero.

If $i \neq j$, then $2^{i-1} \neq 2^{j-1}$. Therefore, $2^{i-1}+2^{j-1}$ is a non-zero element over $\mathrm{GF}(2^n)$ and thus

$$\begin{aligned} p &= \Pr_{(L_\ell,L_r)}[(2^{i-1}+2^{j-1})L_\ell = \delta, \mathtt{msb}_t((2^{i-1}+2^{j-1})L_r) \oplus T \oplus T' = 0^t] \\ &= \Pr_{(L_\ell,L_r)}[(2^{i-1}+2^{j-1})L_\ell = \delta, \mathtt{msb}_t((2^{i-1}+2^{j-1})L_r) = T \oplus T'] \\ &= \frac{1}{2^n} \cdot \frac{1}{2^t} = \frac{1}{2^{n+t}}. \end{aligned}$$

For the case $t > n$, observe that by definition of $\oplus_t$,

$$H_{(L_\ell,L_r)}(T,i) = (2^{i-1}L_\ell, (2^{i-1}L_r \,\|\, 0^{t-n}) \oplus T).$$

Hence, we can use the previous analysis for the special case $t = n$, so that p is at most $1/2^{2n}$. In all cases, p is at most $1/2^{n+\min\{n,t\}}$. □

Combining Lemmas 1 and 2, we obtain the following for the construction of Eq. (3) when $\widetilde{E}^8_K$ is replaced by a TURP.

Lemma 3. *Let* $\mathsf{XT}[\widetilde{\mathsf{P}}, H]$ *be defined as in Eq.* (4) *where* $\widetilde{\mathsf{P}} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *is a TURP and* H *is defined as in Eq.* (5)*. Then, for any adversary making at most* q *queries,*

$$\mathtt{Adv}^{\mathtt{tprp}}_{\mathsf{XT}[\widetilde{\mathsf{P}},H]}(\mathcal{A}) \le \frac{q^2}{2^{n+\min\{n,t\}+1}}.$$

4.2 Collision Probability of ZHASH

Let $\widetilde{E}' : \mathcal{K}' \times \mathcal{T}_J \times \{0,1\}^n \to \{0,1\}^n$ be a TBC with tweak space $\mathcal{T}_J = \mathcal{T} \times \mathcal{J}$ where $\mathcal{T} = \{0,1\}^t$ and $\mathcal{J} = \{1,\ldots,2^n-1\}$ as before. We define $\mathbb{ZHASH}[\widetilde{E}']$ as shown in Figure 4 and depicted in Figure 5. Note that, assuming that masking keys L_ℓ and L_r are uniformly random rather than derived through $\widetilde{E}^9_K$, $\mathsf{ZHASH}[\widetilde{E}]$ is exactly $\mathbb{ZHASH}[\mathsf{XT}[\widetilde{E}^8, H]]$, with H defined as in Eq. (5).

Let $\widetilde{\mathsf{P}}_J : \mathcal{T}_J \times \{0,1\}^n \to \{0,1\}^n$ be a TURP. The following lemma plays a central role in our security proof.

Algorithm $\mathbb{ZHASH}[\widetilde{E}'_{K'}](X)$
($|X|$ is a positive multiple of $n+t$)

1. $U \leftarrow 0^n, V \leftarrow 0^t$
2. $(X[1], \ldots, X[m]) \xleftarrow{n+t} X$
3. **for** $i = 1$ **to** m **do**
4. $\quad (X_\ell, X_r) \xleftarrow{n,t} X[i]$
5. $\quad C_\ell \leftarrow \widetilde{E}'_{K'}((X_r, i), X_\ell)$
6. $\quad C_r \leftarrow C_\ell \oplus_t X_r$
7. $\quad U \leftarrow 2(U \oplus C_\ell)$
8. $\quad V \leftarrow V \oplus C_r$
9. **return** (U, V)

Fig. 4. Pseudocode for the $\mathbb{ZHASH}$ construction using $\widetilde{E}' : \mathcal{K}' \times \mathcal{T}_J \times \{0,1\}^n \to \{0,1\}^n$ with $\mathcal{T}_J = \{0,1\}^t \times \{1, 2, \ldots, 2^n - 1\}$.

Lemma 4. *For any $m, m' \leq 2^{\min\{n,(n+t)/2\}}$, we have*

$$\mathsf{Coll}_{\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J]}(n+t, m, m') \leq \frac{4}{2^{n+\min\{n,t\}}}.$$

Proof. Without loss of generality, we assume $m \leq m'$. Let $X = (X[1], \ldots, X[m])$ and $X' = (X'[1], \ldots, X'[m'])$ be two distinct messages of $(n+t)$-bit blocks. Let $(U, V) = \mathbb{ZHASH}[\widetilde{\mathsf{P}}_J](X)$ and $(U', V') = \mathbb{ZHASH}[\widetilde{\mathsf{P}}_J](X')$ be the outputs. We define $X_r[i]$, $X_\ell[i]$, $C_\ell[i]$, and $C_r[i]$ following Figure 4 augmented with the loop index i. Let $\Delta U = U \oplus U'$, $\Delta V = V \oplus V'$, etc. A collision of $\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J]$ outputs is equivalent to $(\Delta U, \Delta V) = (0^n, 0^t)$.

We perform a case analysis. We first focus on the case $t \leq n$, and consider four sub-cases.

Case 1: $m = m'$, $\exists h \in \{1, \ldots, m\}$, $X[h] \neq X'[h]$, $X[i] = X'[i]$ for $\forall i \neq h$. Then we have

$$\Delta U = \bigoplus_{1 \leq i \leq m} 2^{m-i+1} \Delta C_\ell[i] = 2^{m-h+1} \Delta C_\ell[h],$$
$$\Delta V = \bigoplus_{1 \leq j \leq m} \Delta C_r[j] = \Delta C_r[h].$$

Since the mapping $(X_\ell[i], X_r[i]) \mapsto (C_\ell[i], C_r[i])$ is a permutation, we have $(C_\ell[h], C_r[h]) \neq (C'_\ell[h], C'_r[h])$ and thus we have either $\Delta C_\ell[h] \neq 0^n$ or $\Delta C_r[h] \neq 0^t$. This implies $\Delta U \neq 0^n$ or $\Delta V \neq 0^t$.

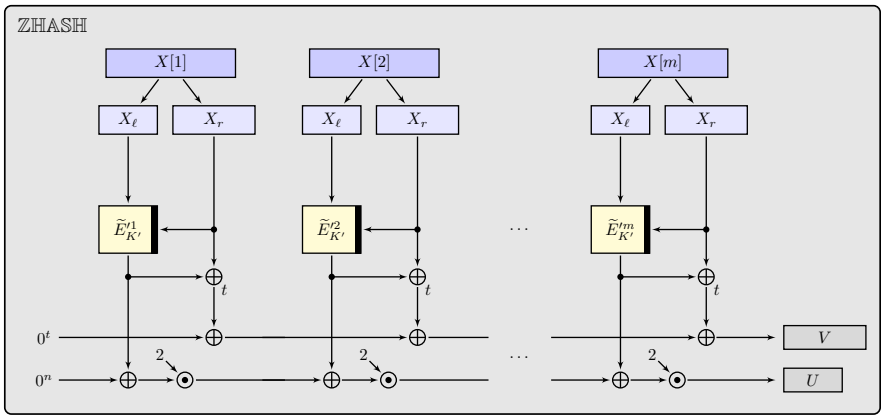

Fig. 5. The ZHASH hash function.

Case 2: $m = m'$, $\exists h, s \in \{1, \ldots, m\}$, $h \neq s$, $X[h] \neq X'[h]$, $X[s] \neq X'[s]$. Then we have

$$\Delta U = 2^{m-h+1}\Delta C_\ell[h] \oplus 2^{m-s+1}\Delta C_\ell[s] \oplus \underbrace{\bigoplus_{\substack{1\le i\le m\\ i\neq h,s}} 2^{m-i+1}\Delta C_\ell[i]}_{\Delta_1},$$

$$\Delta V = \Delta C_r[h] \oplus \Delta C_r[s] \oplus \underbrace{\bigoplus_{\substack{1\le i\le m\\ i\neq h,s}} \Delta C_r[i]}_{\Delta_2}.$$

Observe that Δ_1 and Δ_2 are functions of variables of the form $\widetilde{\mathsf{P}}_J((T,i),X'')$ where $i \notin \{h,s\}$ and T and X'' are determined by X and X'. In particular, by definition of a TURP, they are independent (as random variables) from the other terms in the two right-hand sides. Hence, letting $\lambda_h = 2^{m-h+1}$ and $\lambda_s = 2^{m-s+1}$, and using that since $t \le n$, $C_r[i] = \mathtt{msb}_t(C_\ell[i]) \oplus X_r[i]$, we have

$$\begin{aligned}
\begin{cases} \Delta U = 0^n \\ \Delta V = 0^t \end{cases} &\iff \begin{cases} \lambda_h\Delta C_\ell[h] \oplus \lambda_s\Delta C_\ell[s] = \Delta_1 \\ \Delta C_r[h] \oplus \Delta C_r[s] = \Delta_2 \end{cases} \\
&\iff \begin{cases} \lambda_h\Delta C_\ell[h] \oplus \lambda_s\Delta C_\ell[s] = \Delta_1 \\ \mathtt{msb}_t(\Delta C_\ell[h]) \oplus \Delta X_r[h] \oplus \mathtt{msb}_t(\Delta C_\ell[s]) \oplus \Delta X_r[s] = \Delta_2 \end{cases} \\
&\iff \begin{cases} \lambda_h\Delta C_\ell[h] \oplus \lambda_s\Delta C_\ell[s] = \Delta_1 \\ \mathtt{msb}_t(\Delta C_\ell[h] \oplus \Delta C_\ell[s]) = \Delta_2 \oplus \Delta X_r[h] \oplus \Delta X_r[s]. \end{cases}
\end{aligned}$$

Hence, it follows that

$$\Pr\begin{bmatrix}\Delta U = 0^n \\ \Delta V = 0^t\end{bmatrix} \le \max_{\substack{\delta_1\in\{0,1\}^n \\ \delta_2\in\{0,1\}^t}} \Pr\begin{bmatrix}\lambda_h\Delta C_\ell[h] \oplus \lambda_s\Delta C_\ell[s] = \delta_1 \\ \mathtt{msb}_t(\Delta C_\ell[h] \oplus \Delta C_\ell[s]) = \delta_2\end{bmatrix}$$
$$\le \max_{\substack{\delta_1\in\{0,1\}^n \\ \delta_2\in\{0,1\}^t}} \sum_{\substack{\delta_3\in\{0,1\}^n \\ \mathtt{msb}_t(\delta_3)=\delta_2}} \Pr\begin{bmatrix}\lambda_h\Delta C_\ell[h] \oplus \lambda_s\Delta C_\ell[s] = \delta_1 \\ \Delta C_\ell[h] \oplus \Delta C_\ell[s] = \delta_3\end{bmatrix}.$$

Observe that since $h \neq s$, $\lambda_h \oplus \lambda_s \neq 0$ and the linear system inside the last probability above has a unique solution for any pair (δ_1, δ_3), namely

$$\Delta C_\ell[h] = (\lambda_s\delta_3 \oplus \delta_1)/(\lambda_h \oplus \lambda_s)$$
$$\Delta C_\ell[s] = \delta_3 \oplus (\lambda_s\delta_3 \oplus \delta_1)/(\lambda_h \oplus \lambda_s).$$

Moreover, the random variables $\Delta C_\ell[h]$ and $\Delta C_\ell[s]$ are independent (as they involve distinct tweaks) and their probability distributions are uniform over either $\{0,1\}^n$ or $\{0,1\}^n \setminus \{0^n\}$, implying that their point probabilities are at most $1/(2^n - 1)$. Hence,

$$\Pr\begin{bmatrix}\Delta U = 0^n \\ \Delta V = 0^t\end{bmatrix} \le \max_{\substack{\delta_1\in\{0,1\}^n \\ \delta_2\in\{0,1\}^t}} \sum_{\substack{\delta_3\in\{0,1\}^n \\ \mathtt{msb}_t(\delta_3)=\delta_2}} \frac{1}{(2^n-1)^2}$$
$$\le \frac{2^{n-t}}{(2^n-1)^2} \le \frac{4\cdot 2^{n-t}}{2^{2n}} \le \frac{4}{2^{n+t}}.$$

Case 3: $m' = m + 1$. Then, isolating the terms corresponding to block indices m and $m + 1$, we have

$$\Delta U = \bigoplus_{1\le i\le m} 2^{m-i+1}C_\ell[i] \oplus \bigoplus_{1\le i\le m+1} 2^{m+1-i+1}C'_\ell[i]$$
$$= 2(C_\ell[m] + 2C'_\ell[m] + C'_\ell[m+1] \oplus \Delta_1)$$

and

$$\Delta V = \bigoplus_{1\le\le m} C_r[i] \oplus \bigoplus_{1\le i\le m+1} C'_r[i]$$
$$= \mathtt{msb}_t(C_\ell[m] + C'_\ell[m] + C'_\ell[m+1]) \oplus \Delta_2,$$

where Δ_1 and Δ_2 are independent (as random variables) from $C_\ell[m]$, $C'_\ell[m]$, and $C'_\ell[m+1]$. Hence, exactly as for Case 2, the probability that $\Delta U = 0^n$ and $\Delta V = 0^t$ is at most

$$\max_{\substack{\delta_1\in\{0,1\}^n \\ \delta_2\in\{0,1\}^t}} \sum_{\substack{\delta_3\in\{0,1\}^n \\ \mathtt{msb}_t(\delta_3)=\delta_2}} \Pr\begin{bmatrix}C_\ell[m] + 2C'_\ell[m] + C'_\ell[m+1] = \delta_1 \\ C_\ell[m] + C'_\ell[m] + C'_\ell[m+1] = \delta_3\end{bmatrix}.$$

Letting $Y = C_\ell[m] + C'_\ell[m+1]$ and $Z = C'_\ell[m]$, the linear system in the probability above becomes

$$\begin{cases} Y + 2Z = \delta_1 \\ Y + Z = \delta_3, \end{cases}$$

which has a unique solution over $\mathrm{GF}(2^n)$ for any pair (δ_1, δ_3). Note that Y and Z are uniformly random and independent (since Y involves domain separation integer $m+1$ but Z does not) and hence, the system is satisfied with probability $1/2^{2n}$. Therefore,

$$\Pr\begin{bmatrix} \Delta U = 0^n \\ \Delta V = 0^t \end{bmatrix} \le \max_{\substack{\delta_1 \in \{0,1\}^n \\ \delta_2 \in \{0,1\}^t}} \sum_{\substack{\delta_3 \in \{0,1\}^n \\ \mathtt{msb}_t(\delta_3) = \delta_2}} \frac{1}{2^{2n}} = \frac{1}{2^{n+t}}.$$

Case 4: $m' \ge m+2$. Then, isolating terms corresponding to block indices m' and $m'-1$, we have

$$\Delta U = 2(2C'_\ell[m'-1] \oplus C'_\ell[m'] \oplus \Delta_1),$$
$$\Delta V = \mathtt{msb}_t(C'_\ell[m'-1] \oplus C'_\ell[m']) \oplus \Delta_2,$$

where Δ_1 and Δ_2 are independent of $C'_\ell[m'-1]$ and $C'_\ell[m']$. Moreover, $C'_\ell[m'-1]$ and $C'_\ell[m']$ are independent and uniformly random. Letting $Y = C'_\ell[m']$ and $Z = C'_\ell[m'-1]$, we can apply the same analysis as for Case 3, and therefore, the collision probability is at most $1/2^{n+t}$.

In the above analysis, the collision probability is bounded by $4/2^{n+t}$ for all cases, which proves the lemma for the case $t \le n$.

We next consider the case $t > n$. We let $\overline{X}_w[i] = \mathtt{lsb}_{t-n}(X[i])$ and $\overline{X}_r[i] = \mathtt{lsb}_n(\mathtt{msb}_{2n}(X[i]))$, i.e., the $(n+1)$-th to $2n$-th bits of $X[i]$. For $V \in \{0,1\}^t$, let $\overline{V} = \mathtt{msb}_n(V)$ and $\overline{W} = \mathtt{lsb}_{t-n}(V)$, thus $V = (\overline{V} \,\|\, \overline{W})$. The corresponding variables are also defined for X'.

We first focus on the case $m = m'$. When $\overline{X}_w[i] = \overline{X}'_w[i]$ for all $1 \le i \le m$, the analysis is the same as the case $t \le n$, since for each i-th input block, $\widetilde{\mathsf{P}}_J$ takes exactly the same values (between X and X') for the last $(t-n)$-bit of $\mathcal{T}$. Thus the output collision probability (in particular, the first $2n$-bit of output (U, V)) is at most $4/2^{2n}$.

If there exists an index i such that $\overline{X}_w[i] \ne \overline{X}'_w[i]$ and $\overline{X}_w[j] = \overline{X}'_w[j]$ for all $j \ne i$, we have $\Delta\overline{W} \ne 0^{t-n}$, that is, the non-zero difference in the last $(t-n)$ bits of ΔV. Hence the collision probability is zero.

If there exist two (or more) distinct indices i, j such that $\overline{X}_w[i] \ne \overline{X}'_w[i]$ and $\overline{X}_w[j] \ne \overline{X}'_w[j]$, the analysis is almost the same as (the Case 2 of) the case $t \le n$. The collision probability of (U, V) is at most $1/2^{2n}$.

Finally, we consider the case $m < m'$. For both $m' = m+1$ and $m' \ge m+2$, we can apply the same arguments as the corresponding cases for $t \le n$ and the collision probability of (U, V) is at most $1/2^{2n}$. Summarizing, the collision probability of (U, V) is at most $4/2^{2n}$. □

We remark that because of Case 1 when $t \leq n$, $\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J]$ is not almost XOR universal (i.e., the output differential probability is not guaranteed to be small).

4.3 PRF Security of Finalization

We prove that ZFIN is a fixed-input-length PRF with n-bit security. The key observation is that, given $V \in \{0,1\}^t$, ZFIN is reduced to a pair of independent instances of the sum of two independent random permutations, also called SUM2 by Lucks [Luc00]. More precisely, let SUM2 be a function that maps n-bit input to n-bit output, such that $\mathsf{SUM2}(X) \stackrel{\text{def}}{=} \mathsf{P}_1(X) \oplus \mathsf{P}_2(X)$ for $X \in \{0,1\}^n$, using two independent n-bit URPs P_1 and P_2.

On input (U, V), each n-bit output in ZFIN is equivalent to $\mathsf{SUM2}(U)$ for two independent n-bit URPs P_1 and P_2, and the sampling of the pair of these URPs is independent for each $V \in \{0,1\}^t$ and for the output blocks, thanks to the domain separation.

SUM2 has been actively studied and BBB bounds have been proved [Luc00, BI99]. Among them, Patarin [Pat08, Pat13] has proved that

$$\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{SUM2}}(\mathcal{A}) \leq O\left(\frac{q}{2^n}\right),$$

for any adversary $\mathcal{A}$ using q queries. However, the constant is not known in the literature. Here, following [PS16], we propose a well-accepted conjecture that SUM2 is an n-bit secure PRF with a small constant.

Conjecture 1. For any adversary with q queries, $\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{SUM2}}(\mathcal{A}) \leq Cq/2^n$ holds for some small constant $C > 0$.

For $i \in \{0, 4\}$, we let $\mathsf{ZFIN}_i[\widetilde{E}_K](U, V) = \mathsf{ZFIN}[\widetilde{E}_K](i, U, V)$. Based on Conjecture 1, the following lemma gives the PRF security of ZFIN_i in the information-theoretic setting, i.e., when $\widetilde{E}_K$ is replaced by a TURP $\widetilde{\mathsf{P}}_I : \mathcal{T}_I \times \{0,1\}^n \to \{0,1\}^n$.

Lemma 5. *Let $\mathcal{A}$ be an adversary against the PRF-security of $\mathsf{ZFIN}_i[\widetilde{\mathsf{P}}_I]$ making at most q queries. Then, for $i \in \{0, 4\}$, we have*

$$\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZFIN}_i[\widetilde{\mathsf{P}}_I]}(\mathcal{A}) \leq \sum_{T \in \{0,1\}^t} \frac{2Cq_T}{2^n} \leq \frac{2Cq}{2^n},$$

where q_T denotes the number of queries with $V = T$.

The proof is obtained by the standard hybrid argument and an observation that adaptive choice of q_T's does not help. Lemma 5 shows that ZFIN is a parallelizable and n-bit secure PRF with $(n + t)$-bit inputs using a TBC with n-bit blocks and t-bit tweaks.

Alternative Constructions. We could build the finalization function from [CDMS10, Min09]. Coron *et al.* [CDMS10] proposed a $2n$-bit SPRP construction using 3 TBC calls of n-bit block and tweak, and Minematsu [Min09] proposed a $2n$-bit SPRP construction using 2 TBC calls with two $\mathrm{GF}(2^n)$ multiplications. Both constructions achieve n-bit security with small constants. As they are also n-bit secure $2n$-bit PRFs (via standard PRP-PRF switching), we could use them. However, they are totally serial, hence if input to MAC is short (say 64 bytes) and we have a parallel TBC computation unit, this choice of finalization will be quite slower than ZFIN.

We could also use CENC by Iwata [Iwa06]. In a recent work by Iwata *et al.* [IMV16], it is shown that $\mathsf{P}(X \,\|\, 0) \oplus \mathsf{P}(X \,\|\, 1)$ for $X \in \{0,1\}^{n-1}$, called XORP[1], achieves n-bit PRF-security with constant 1, by making explicit that this was in fact already proved by Patarin [Pat10]. However, we think the finalization based on this construction would be slightly more complex than ours.

4.4 PRF Security of ZMAC

We are now ready to state and prove the security result for ZMAC.

Theorem 1. *Let $\mathcal{A}$ be an adversary against $\mathsf{ZMAC}[\widetilde{E}]$ making at most q queries of total length (in number of $(n+t)$-bit blocks) at most σ and running in time at most $\mathtt{time}$. Then there exists an adversary $\mathcal{B}$ against $\widetilde{E}$ making at most $\sigma+4q+2$ queries and running in time at most $\mathtt{time}+O(\sigma)$ such that*

$$\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZMAC}[\widetilde{E}]}(\mathcal{A}) \le \mathtt{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{B}) + \frac{2.5\sigma^2}{2^{n+\min\{n,t\}}} + \frac{4Cq}{2^n},$$

where the constant $C > 0$ is as specified in Conjecture 1.

Proof. Since ZMAC calls the underlying TBC $\widetilde{E}$ with a single key K, we can replace $\widetilde{E}_K$ by a TURP $\widetilde{\mathsf{P}}_I : \mathcal{T}_I \times \{0,1\}^n \to \{0,1\}^n$ and focus on the information-theoretic security of $\mathsf{ZMAC}[\widetilde{\mathsf{P}}_I]$. Derivation of the computational counterpart is standard.

Let $G : \mathcal{K}_G \times (\{0,1\}^{n+t})^+ \to \{0,1\}^{n+t}$ and $F : \mathcal{K}_F \times \{0,1\}^{n+t} \to \{0,1\}^{2n}$. Let $\mathsf{CW3}[G_{K_1}, F_{K_2}, F_{K_3}]$ be the three-key Carter-Wegman construction with independent keys (K_1, K_2, K_3) as defined by Black and Rogaway [BR05], i.e.,

$$\mathsf{CW3}[G_{K_1}, F_{K_2}, F_{K_3}](M) = \begin{cases} F_{K_2}(G_{K_1}(\mathtt{ozp}(M))) & \text{if } M \in (\{0,1\}^{n+t})^+, \\ F_{K_3}(G_{K_1}(\mathtt{ozp}(M))) & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathsf{ZMAC}[\widetilde{\mathsf{P}}_I]$ is a instantiation of CW3. Indeed,

$$\mathsf{ZMAC}[\widetilde{\mathsf{P}}_I] = \mathsf{CW3}\big[\mathsf{ZHASH}[\widetilde{\mathsf{P}}_I], \mathsf{ZFIN}_0[\widetilde{\mathsf{P}}_I], \mathsf{ZFIN}_4[\widetilde{\mathsf{P}}_I]\big],$$

and independence between the three components follows from domain separation of tweaks which implies that for distinct integers $i, j \in \mathcal{I}$, $\widetilde{\mathsf{P}}^i_I$ and $\widetilde{\mathsf{P}}^j_I$ are independent TURPs with tweak space $\mathcal{T} = \{0,1\}^t$. Besides, as already observed in

Section 4.2, since the masking keys $L_\ell = \widetilde{\mathsf{P}}_I^9(0^t, 0^n)$ and $L_r = \widetilde{\mathsf{P}}_I^9(0^{t-1}1, 0^n)$ are uniformly random, one has

$$\mathsf{ZHASH}[\widetilde{\mathsf{P}}_I] = \mathbb{ZHASH}\big[\mathsf{XT}[\widetilde{\mathsf{P}}_I^8, H]\big],$$

with H as defined by Eq. (5). Hence, by replacing $\mathsf{XT}[\widetilde{\mathsf{P}}_I^8, H]$ by a TURP $\widetilde{\mathsf{P}}_J : \mathcal{T}_J \times \{0,1\}^n \to \{0,1\}^n$ and ZFIN_0, resp. ZFIN_4 by independent random functions R_0, resp. R_1 from $\{0,1\}^{n+t}$ to $\{0,1\}^n$, we have that there exists an adversary $\mathcal{B}'$ against $\mathsf{XT}[\widetilde{\mathsf{P}}_I^8, H]$ making at most σ queries and an adversary $\mathcal{B}''$ against $\mathsf{ZFIN}_{0/4}[\widetilde{\mathsf{P}}_I]$ making at most q queries such that

$$\begin{aligned}
\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZMAC}[\widetilde{\mathsf{P}}_I]}(\mathcal{A}) &= \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{CW3}[\mathbb{ZHASH}[\mathsf{XT}[\widetilde{\mathsf{P}}_I^8,H]],\mathsf{ZFIN}_0[\widetilde{\mathsf{P}}_I],\mathsf{ZFIN}_4[\widetilde{\mathsf{P}}_I]]}(\mathcal{A}) \\
&\le \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{CW3}[\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J],\mathsf{R}_0,\mathsf{R}_1]}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{tprp}}_{\mathsf{XT}[\widetilde{\mathsf{P}}_I^8,H]}(\mathcal{B}') \\
&\quad + \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZFIN}_0[\widetilde{\mathsf{P}}_I]}(\mathcal{B}'') + \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZFIN}_4[\widetilde{\mathsf{P}}_I]}(\mathcal{B}'') \\
&\le \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{CW3}[\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J],\mathsf{R}_0,\mathsf{R}_1]}(\mathcal{A}) + \frac{\sigma^2}{2^{n+\min\{n,t\}+1}} + \frac{4Cq}{2^n}, \qquad (6)
\end{aligned}$$

where the last inequality follows from Lemmas 3 and 5.

From Lemma 2 of [BR05] and Lemma 4, we have

$$\begin{aligned}
\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{CW3}[\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J],\mathsf{R}_0,\mathsf{R}_1]}(\mathcal{A}) &\le \max_{m_1,\ldots,m_q} \sum_{i\neq j} \mathsf{Coll}_{\mathbb{ZHASH}[\widetilde{\mathsf{P}}_J]}(n+t, m_i, m_j) \\
&\le \max_{m_1,\ldots,m_q} \sum_{i\neq j} \frac{4}{2^{n+\min\{n,t\}}} \\
&\le \frac{2q^2}{2^{n+\min\{n,t\}}}, \qquad (7)
\end{aligned}$$

where the maximum is taken over all $m_1, \ldots, m_q$ such that $\sum_i m_i = \sigma$. Combining (6) and (7), we obtain the information-theoretic bound. □

4.5 Other Variants of ZMAC

ZMAC has a wide range of variants, depending on the required level of security. We briefly discuss some of them.

Eliminating The Input-Length Effect. ZMAC ensures security as long as the total number of $(n+t)$-bit blocks σ throughout queries is small compared to $2^{\min\{n,(n+t)/2\}}$. If one wants to completely remove the effect of the input length as in [Nai15, LN17] (i.e., to get security as long as the number of *queries* q is small compared to $2^{\min\{n,(n+t)/2\}}$), we suggest to use $\mathbb{ZHASH}$. The underlying TBC $\widetilde{E}$ needs to have a tweak space of the form $\{0,1\}^t \times \mathcal{J} \times \mathcal{I}$, where $\mathcal{J} = \{1, 2, \ldots, B\}$ for some $B > 0$ and $\mathcal{I}$ is a set of domain separation integers. Here, the effective

tweak space of $\widetilde{E}$ is $\{0,1\}^t \times \mathcal{J}$ and the effective tweak-length is $t' = t + \log_2 B$ bits.

For finalization, we can use $\mathsf{ZFIN}[\widetilde{E}]$ with an adequate domain separation. From Lemma 4, the message hashing has a constant collision probability of $4/2^{n+\min\{n,t\}}$ for both cases of $t \leq n$ and $t > n$. The security bounds (for both $t \leq n$ and $t > n$) are $O(q^2/2^{n+\min\{n,t\}})$ plus the PRF bound of $\mathsf{ZFIN}[\widetilde{E}]$, thus, security does not degrade with the total input length.

On the downside, since we waste $\log_2 B$ effective tweak bits to process the input block index, this mode processes only $n+t$ input bits per TBC call rather than the optimal amount $n+t'$. This is a trade-off between efficiency and security.

Birthday Security. If we only require up-to-birthday bound security, then we could simply use $\mathsf{XT}[\widetilde{E}]$ in the same manner to PMAC, that is, the message hashing is mostly the same as ZHASH, however we XOR all TBC outputs C_ℓ in Figure 1 to form the final n-bit output. The finalization is done by a single TBC call with an adequate domain separation, and hashing and finalization are composed by CW3.

From Lemma 3 and the security proof for (TBC-based) PMAC1 found in [Rog04], this variant has PRF advantage $O(\sigma^2/2^{n+\min\{n,t\}} + q^2/2^n)$, which is slightly better than "standard" birthday bound $O(\sigma^2/2^n)$. Efficiency is optimal since $n+t$ input bits are processed per TBC call for any $\widetilde{E}$ having effective tweak space of t bits, for any $t > 0$.

5 Application to Authenticated Encryption: ZAE

As an application of ZMAC, we provide an efficient construction of a Deterministic Authenticated Encryption (DAE) scheme [RS06] from a TBC called ZAE.

Let us briefly recall the syntax and the security definition for a DAE scheme (see [RS06] for details). A DAE scheme DAE is a tuple $(\mathcal{K}, \mathcal{AD}, \mathcal{M}, \mathcal{C}, \mathsf{DAE.Enc}, \mathsf{DAE.Dec})$, where $\mathcal{K}$, $\mathcal{AD}$, $\mathcal{M}$, and $\mathcal{C}$ are non-empty sets and DAE.Enc and DAE.Dec are deterministic algorithms. The encryption algorithm DAE.Enc takes as input a key $K \in \mathcal{K}$, associated data $AD \in \mathcal{AD}$, and a plaintext $M \in \mathcal{M}$, and returns a ciphertext $C \in \mathcal{C}$. The decryption algorithm DAE.Dec takes as input a key $K \in \mathcal{K}$, associated data $AD \in \mathcal{AD}$, and a ciphertext $C \in \mathcal{C}$, and returns either a message $M \in \mathcal{M}$ or the special symbol $\perp$ indicating that the ciphertext is invalid. We write $\mathsf{DAE.Enc}_K(AD, M)$, resp. $\mathsf{DAE.Dec}_K(AD, C)$ for $\mathsf{DAE.Enc}(K, AD, M)$, resp. $\mathsf{DAE.Dec}(K, AD, C)$. As usual, we require that for any tuple $(K, AD, M) \in \mathcal{K} \times \mathcal{AD} \times \mathcal{M}$, one has

$$\mathsf{DAE.Dec}(K, AD, \mathsf{DAE.Enc}(K, AD, M)) = M.$$

The associated data AD is authenticated but not encrypted, and may include a nonce, which is why DAE is sometimes called nonce-misuse resistant authenticated encryption (MRAE), since for such a scheme the repetition of a nonce does not hurt authenticity and only allows the adversary to detect repetitions of inputs (AD, M) to the encryption algorithm.

Definition 4. *Let* DAE *be a DAE scheme. The advantage of an adversary* $\mathcal{A}$ *in breaking the DAE-security of* DAE *is defined as*

$$\mathtt{Adv}^{\mathtt{dae}}_{\mathsf{DAE}}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{DAE.Enc}_K,\mathsf{DAE.Dec}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$,\perp} \Rightarrow 1]\right|,$$

where oracle $\$(\cdot,\cdot)$*, on input* (AD, M)*, returns a random bit string of length*[10] $|\mathsf{DAE.Enc}_K(AD, M)|$*, and oracle* $\perp(\cdot,\cdot)$ *always returns* $\perp$*. The adversary* $\mathcal{A}$ *is not allowed to repeat an encryption query or to submit a decryption query* (AD, C) *if a previous encryption query* (AD, M) *returned* C*.*

In addition to ZMAC, our construction will rely on a (random) IV-based encryption (ivE) scheme IVE. Such a scheme consists of a tuple $(\mathcal{K}, \mathcal{IV}, \mathcal{M}, \mathcal{C}, \mathsf{IVE.Enc}, \mathsf{IVE.Dec})$, where $\mathcal{K}$, $\mathcal{IV}$, $\mathcal{M}$, and $\mathcal{C}$ are non-empty sets and $\mathsf{IVE.Enc}$ and $\mathsf{IVE.Dec}$ are deterministic algorithms. The encryption algorithm $\mathsf{IVE.Enc}$ takes as input a key $K \in \mathcal{K}$, an initialization value $IV \in \mathcal{IV}$, and a plaintext $M \in \mathcal{M}$, and returns a ciphertext $C \in \mathcal{C}$. The decryption algorithm $\mathsf{IVE.Dec}$ takes as input a key $K \in \mathcal{K}$, an IV $IV \in \mathcal{IV}$, and a ciphertext $C \in \mathcal{C}$, and returns a message $M \in \mathcal{M}$. Given $K \in \mathcal{K}$, we let $\mathsf{IVE.Enc}^{\$}_K$ denote the randomized algorithm which takes as input $M \in \mathcal{M}$, draws $IV \xleftarrow{\$} \mathcal{IV}$, computes $C = \mathsf{IVE.Enc}(K, IV, M)$, and returns (IV, C).

Definition 5. *Let* IVE *be an IV-based encryption scheme. The advantage of an adversary* $\mathcal{A}$ *in breaking the ivE-security of* IVE *is defined as*

$$\mathtt{Adv}^{\mathtt{ive}}_{\mathsf{IVE}}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{IVE.Enc}^{\$}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1]\right|,$$

where oracle $\$(\cdot)$*, on input* $M \in \mathcal{M}$*, returns a random bit string of length* $|\mathsf{IVE.Enc}^{\$}_K(M)|$*.*

For our purposes, we consider the IV-based encryption mode IVCTRT proposed in [PS16, Appendix B]. This mode uses a TBC $\widetilde{E}$ with tweak space $\mathcal{T}' = \{0,1\}^t \times \mathcal{I}$ and message space $\{0,1\}^n$, and has $2n$-bit IVs. We assume $10 \in \mathcal{I}$ as all calls to $\widetilde{E}$ in IVCTRT will use domain separation integer 10 which is distinct from all those used in ZMAC. The encryption $\mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Enc}(IV, M)$ of a message M with initialization value IV under key K is defined as follows. The IV and the message are parsed as

$$(IV[1], IV[2]) \xleftarrow{n,n} IV$$
$$(M[1], \ldots, M[m]) \xleftarrow{n} M.$$

Let $IV'[1] = IV[1] \oplus_t 0^t$, i.e., $IV[1]$ is either padded with zeros up to t bits when $t > n$ or truncated to t bits when $t \leq n$. Then, the ciphertext is $C = (C[1], \ldots, C[m])$ where $X \boxplus Y$ denotes t-bit modular addition,

$$C[i] = M[i] \oplus \widetilde{E}^{10}_K(IV'[1] \boxplus i, IV[2]) \qquad \text{for } i = 1, \ldots, m-1,$$
$$C[m] = M[m] \oplus \mathtt{msb}_{|M[m]|}(\widetilde{E}^{10}_K(IV'[1] \boxplus m, IV[2])).$$

[10] We assume that the length of $\mathsf{DAE.Enc}_K(AD, M)$ is independent from the key K.

Algorithm $\mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Enc}(IV, M)$

1. $(IV[1], IV[2]) \xleftarrow{n,n} IV$
2. $IV[1] \leftarrow IV[1] \oplus_t 0^t$
3. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
4. **for** $i = 1$ **to** $m - 1$ **do**
5. $\quad C[i] \leftarrow M[i] \oplus \widetilde{E}_K^{10}(IV[1] \boxplus i, IV[2])$
6. $S \leftarrow \mathtt{msb}_{|M[m]|}(\widetilde{E}_K^{10}(IV[1] \boxplus m, IV[2]))$
7. $C[m] \leftarrow M[m] \oplus S$
8. $C \leftarrow (C[1] \| \ldots \| C[m])$
9. **return** C

Algorithm $\mathtt{encode}(AD, M)$

1. $\mathtt{Len} \leftarrow \mathtt{str}_{n/2}(|AD|) \| \mathtt{str}_{n/2}(|M|)$
2. $X \leftarrow (\mathtt{ozp}(AD) \| \mathtt{ozp}(M) \| \mathtt{Len})$
3. **return** X

Algorithm $\mathsf{ZAE}[\widetilde{E}_K].\mathsf{Enc}(AD, M)$

1. $X \leftarrow \mathtt{encode}(AD, M)$
2. $IV \leftarrow \mathsf{ZMAC}[\widetilde{E}_K](X)$
3. $C \leftarrow \mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Enc}(IV, M)$
4. **return** $C' = (IV, C)$

Algorithm $\mathsf{ZAE}[\widetilde{E}_K].\mathsf{Dec}(AD, C')$

1. $(IV, C) \xleftarrow{2n,|C'|-2n} C'$
2. $M \leftarrow \mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Dec}(IV, C)$
3. $X \leftarrow \mathtt{encode}(AD, M)$
4. $IV' = \mathsf{ZMAC}[\widetilde{E}_K](X)$
5. **if** $IV' = IV$ **then return** M
6. **else return** $\perp$

Fig. 6. Pseudocode for the ZAE deterministic authenticated encryption scheme. Algorithm $\mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Dec}$ is similar to $\mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Enc}$ and hence omitted.

Our TBC-based BBB-secure DAE mode proposal ZAE follows the generic[11] SIV construction [RS06], where the PRF is instantiated with ZMAC and the IV-based encryption mode is instantiated with IVCTRT.

Let $\widetilde{E}$ be a TBC with tweak space $\mathcal{T}' = \{0,1\}^t \times \mathcal{I}$ where $\mathcal{I} \supseteq \{0, 1, \ldots, 10\}$ and message space $\{0,1\}^n$. The encryption $\mathsf{ZAE}[\widetilde{E}_K].\mathsf{Enc}(AD, M)$ of a message M with associated data AD under key K is the pair $C' = (IV, C)$ where

$$IV = \mathsf{ZMAC}[\widetilde{E}_K](\mathtt{encode}(AD, M))$$
$$C = \mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Enc}(IV, M).$$

The encode function is an injective mapping which pads AD and M independently using the ozp() function, so that the bit length of the resulting strings are multiples of $(n + t)$. Then, it concatenates these two strings and appends the $n/2$-bit representations of the lengths of AD and M (an n-bit representation can naturally be used if more than $2^{n/2}$ AD and M blocks are possible). The tag (synthetic IV) is $2n$ bits, which is inevitable for n-bit security of the SIV construction, since a collision of two tags would immediately break the scheme. See Figure 6 for the pseudocode and Figure 7 for a graphical representation of ZAE.

The security bound for ZAE is given in the following theorem. Here, we let the length of a query (encryption or decryption) be the block length of

[11] The name SIV is used in [RS06] to denote either a generic construction of a DAE scheme from a PRF and an IV-based encryption scheme, or the block cipher mode of operation resulting from instantiating the PRF with (a variant of) CMAC and the encryption scheme with the counter mode.

$\mathtt{encode}(AD, M)$, where $(IV, C) \xleftarrow{2n,|C'|-2n} C'$ and $M \leftarrow \mathsf{IVCTRT}[\widetilde{E}_K].\mathsf{Dec}(IV, C)$ for a decryption query (AD, C').

Theorem 2. *Let $\widetilde{E}$ be a TBC with tweak space $\mathcal{T}' = \{0,1\}^t \times \mathcal{I}$ and message space $\{0,1\}^n$. Let $\mathcal{A}$ be an adversary attacking $\mathsf{ZAE}[\widetilde{E}]$ making at most q (encryption or decryption) queries, such that the total length of all its queries is at most σ blocks of n bits*[12]*, and running in time at most $\mathtt{time}$. Then there exists an adversary $\mathcal{B}$ against $\widetilde{E}$ making at most $2\sigma + 4q + 2$ chosen-plaintext queries and running in time at most $\mathtt{time} + O(\sigma)$ such that*

$$\mathtt{Adv}^{\mathtt{dae}}_{\mathsf{ZAE}[\widetilde{E}]}(\mathcal{A}) \leq \mathtt{Adv}^{\mathtt{tprp}}_{\widetilde{E}}(\mathcal{B}) + \frac{3.5\sigma^2}{2^{n+\min\{n,t\}}} + \frac{4Cq}{2^n} + \frac{q}{2^{2n}},$$

where the constant C is from Conjecture 1.

Proof. We prove the information-theoretic security of $\mathsf{ZAE}[\widetilde{\mathsf{P}}]$ where $\widetilde{\mathsf{P}}$ is a TURP (the computational counterpart is standard). By Theorem 2 of [RS06], there exists an adversary $\mathcal{A}'$ attacking $\mathsf{ZMAC}[\widetilde{\mathsf{P}}]$ and an adversary $\mathcal{A}''$ attacking $\mathsf{IVCTRT}[\widetilde{\mathsf{P}}]$, both making at most q queries of total length σ, such that

$$\mathtt{Adv}^{\mathtt{dae}}_{\mathsf{ZAE}[\widetilde{\mathsf{P}}]}(\mathcal{A}) \leq \mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZMAC}[\widetilde{\mathsf{P}}]}(\mathcal{A}') + \mathtt{Adv}^{\mathtt{ive}}_{\mathsf{IVCTRT}[\widetilde{\mathsf{P}}]}(\mathcal{A}'') + \frac{q}{2^{2n}}. \tag{8}$$

According to [PS16, Appendix B], we have

$$\mathtt{Adv}^{\mathtt{ive}}_{\mathsf{IVCTRT}[\widetilde{\mathsf{P}}]}(\mathcal{A}'') \leq \frac{\sigma^2}{2^{n+\min\{n,t\}}}.$$

(In more details, the security bound from [PS16, Appendix B] is $\sigma^2/2^{n+t}$ assuming $IV'[1]$ is uniform in $\{0,1\}^t$, which is the case here only when $t \leq n$. When $t > n$, the security bound caps at $\sigma^2/2^{2n}$ since only the first n bits of $IV'[1]$ are random.) The result follows by combining these two equations with Theorem 1. The query complexity of $\mathcal{B}$ follows from the fact that ZAE makes at most 2 TBC calls per n-bit block of input and the complexity of ZFIN and masks. □

It is to be noted that for the encryption part IVCTRT there is no specific efficiency benefit in having access to a TBC with a larger tweak input than n bits. In contrary, for the ZMAC part, there is a direct gain in having a large tweak if this is not too costly (say much smaller than a factor of two), since this increases the amount of input bits per TBC call. In order to optimize performance, one can thus use a TBC with $t = n$ for the encryption part, but switch to a TBC with $t > n$ for the MAC part of the scheme, since building a TBC with a large tweak usually leads to (slightly) slower performances than a TBC with a small tweak [JNP14d].

Another direction to further increase performance of ZAE in practice, without reducing its security, is to use a counter addition on only $\min\{n,t\}$ bits instead of t bits, i.e. by redefining $X \boxplus Y$ for $Y \in \{1, \ldots, 2^{\min\{n,t\}}\}$ to denote

$$\mathtt{msb}_{\min\{n,t\}}(X) + Y \bmod 2^{\min\{n,t\}} \,\|\, \mathtt{lsb}_{t-\min\{n,t\}}(X),$$

[12] Note that, for simplicity, the lengths are counted in n-bit blocks.

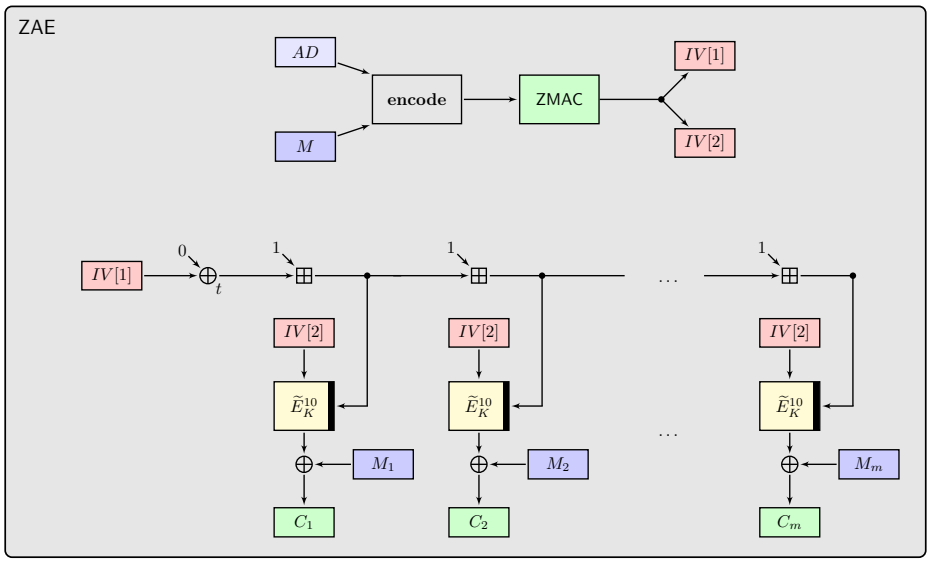

Fig. 7. The ZAE deterministic authenticated encryption scheme with associated data. Note that the n-bit value $IV[1]$ is mapped to the t-bit value $IV[1] \oplus_t 0^t$ to obtain the initial t-bit counter.

that is, addition over the first $\min\{n, t\}$ bits and the remaining bits intact. One could even consider having a LFSR-based counter instead of a modular addition based counter to improve hardware implementations. We have not used these improvements in ZAE specifications in order to simplify its description.

ZAE compares very favorably with existing TBC-based MRAE solutions both in terms of efficiency and security. Indeed, it can process $n + t$ message bits per TBC call for the MAC part, and n bits per TBC call for the encryption part. Other schemes such as SIV [RS06], SCT [PS16], or SIVx [LN17] can only handle n message bits per TBC call in the MAC part. Moreover, ZAE is secure beyond the birthday bound and hence provides better security than SIV (only birthday security) or SCT (only birthday security in the nonce-misuse setting) while leading to better performances.

We remark that ZMAC could also be used to improve OCB-like (more precisely its TBC-based generalization ΘCB [KR11]) or SCT-like designs: by changing the PMAC-like part that handles the associated data for ZMAC, one would fully benefit from the efficiency improvement provided by our design.

6 MAC and AE Instances

In this section, we give instantiation examples of ZMAC and ZAE. There are many possible ways to build a TBC, but in practice block cipher-based constructions

are generally less efficient than ad-hoc TBCs. Since our design leverages heavily the possibilities offered by a large tweak, a candidate such as Threefish [FLS+10] is not very interesting as it handles only 128 bits of tweak input for a block size of 256/512/1024 bits. The effective efficiency gain would be limited (and Threefish is much slower than AES on current platforms, due to AES-NI instruction sets).

One could also consider using block ciphers with large keys (in comparison to their block size), but as remarked in [JNP14d], it remains unclear if one can generally use the key input of a TBC as tweak input. For example, using AES-256 while allocating half of its key input as tweak is a very bad idea, considering the related-key attacks against AES-256, such as [BKN09].

Recently, Jean *et al.* [JNP14d] proposed a framework called TWEAKEY and a generic construction STK for building ad-hoc tweakable Substitution-Permutation Network (SPN) ciphers. The authors proposed three TBCs based on the STK framework, Deoxys-BC [JNP14a], Joltik-BC [JNP14b], and KIASU-BC [JNP14c], as part of three candidates for CAESAR authenticated encryption competition [CAE]. In particular, Deoxys-BC is the TBC used in the Deoxys CAESAR candidate (together with the SCT authenticated encryption mode), selected for the third round of the competition. Later, SKINNY [BJK+16], a lightweight family of TBCs based on similar ideas was proposed.

We will study here the performances of ZMAC and ZAE when instantiated with Deoxys-BC and the 128-bit block versions of SKINNY. Note that for a key size of 128 bits, both these ciphers offer versions with 128 or 256 bits of tweak input (respectively Deoxys-BC-256/SKINNY-128-256 and Deoxys-BC-384/SKINNY-128-384). It is interesting to compare the respective number of rounds (and thus efficiencies) of these different versions (see Table 2).

Table 2. Number of rounds of Deoxys-BC-256/ Deoxys-BC-384, and SKINNY-128-128/ SKINNY-128-256/ SKINNY-128-384.

TBC	$t = 0$	$t = n$	$t = 2n$
Deoxys-BC	–	14	16
SKINNY	40	48	56

This shows the strength of the ZMAC general design: for practical ad-hoc TBC constructions, it seems that adding twice more input to the TBC slows down the primitive by a much smaller factor than 2. Thus, we can expect the efficiency to improve with the tweak-length.

6.1 Handling the Domain Separation of TBC Instances

In ZMAC and ZAE, we use several independent TBC instances through domain separation integers. In detail, for ZMAC, one needs one TBC instance ($\widetilde{E}^9_K$) for the generating the masking keys L_ℓ and L_r, one instance ($\widetilde{E}^8_K$) for the hashing

part, 4 instances ($\widetilde{E}_K^0$, $\widetilde{E}_K^1$, $\widetilde{E}_K^2$, $\widetilde{E}_K^3$) for the finalization function when the message is a positive multiple of $(n+t)$ bits, and 4 instances ($\widetilde{E}_K^4$, $\widetilde{E}_K^5$, $\widetilde{E}_K^6$, $\widetilde{E}_K^7$) for the finalization function when the message is not a positive multiple of $(n+t)$ bits. This sums up to 10 instances. Moreover, ZAE requires one more instance ($\widetilde{E}_K^{10}$) for the encryption part.

For all instances, encoding can be achieved by simply reserving 4 bits of the tweak input of the TBC. This has the advantage of being very simple and elegant, but it also means that in practice the message block size of ZMAC will be a little unusual (as the tweak-length is usually a multiple of the block-length).

Another solution is to separate the instances using distinct field multiplications. This allows the message block size of ZMAC to be a multiple of the TBC block size. However, the number of distinct multiplications is non-negligible and will render the implementation much more complex.

Finally, a last solution could be to XOR into the state distinct words that are dependent of the secret key (for example generated just like the masks L_ℓ and L_r, but with different plaintext inputs). The advantage is that the implementation is simple and it allows the message block size of ZMAC to be a multiple of the TBC block size. However, more precomputations will be needed.

All these solutions represent different possible tradeoffs, and we note that this issue is present for most TBC-based MAC or AE schemes.

6.2 Efficiency Comparisons

In this subsection, we report the efficiency estimates of our operating modes ZMAC and ZAE, when the TBC is instantiated with Deoxys-BC and SKINNY, while comparing with existing MAC and AE schemes.

We do not perform a comprehensive comparison with schemes combining a (T)BC and a $2n$-bit algebraic UHFs, such as a 256-bit variant of GMAC [MV04]. In principle such schemes can achieve n-bit security. However, the additional implementation of an algebraic UHF would require more resources (memory for software and gates for hardware) than pure (T)BC modes, which is not desirable for the performance across multiple devices. Moreover, the existence of weak-key class for the popular polynomial hash functions, such as [HP08, PC15], can be an issue.

We will consider two scenarios: (1) long messages and (2) long messages with equally long associated data (AD). For these two scenarios, the cost of the precomputations or finalizations can be considered negligible (for benchmarking, we used 65536 bytes for long messages or AD). Moreover, we note that in ZMAC, the two calls for precomputation can be done in parallel, while the calls in the finalization function ZFIN can also all be done in parallel. For modern processors, where parallel encryptions (for bitslice implementations) or pipelined encryptions (for implementations using the AES-NI instructions set) are by far the most efficient strategy, having a finalization composed of four parallel encryption calls (like in ZMAC) or a single one (like in SCT) will not make a big difference in terms of efficiency.

Table 3. Estimated efficiencies (in c/B) of various MAC and AE primitives (for (1) long messages and (2) long message with equally long AD) on a Intel Skylake processor. For (2), the input bytes are the sum of message and AD bytes. NR denotes the nonce-respecting scenario. GCM-SIV is proposed by [GL15]. (*) Performances are reported for SIV instantiated with a fully parallelizable PRF (e.g., PMAC), while the specifications from [RS06] use a PRF based on CMAC which has a limited parallelizability.

mode	Cipher	Long M	Long M Long AD	Security
Message Authentication Code				
CMAC	AES-128	2.68	–	64
PMAC	AES-128	0.65	–	64
PMAC1	Deoxys-BC-256	0.87	–	64
PMAC_TBC1k	Deoxys-BC-256	0.87	–	128
ZMAC	Deoxys-BC-256	0.61	–	128
ZMAC	Deoxys-BC-384	0.52	–	128
ZMAC	SKINNY-128-256	2.06	–	128
ZMAC	SKINNY-128-384	1.60	–	128
(Deterministic) Authenticated Encryption				
OCB	AES-128	0.65	0.65	64 (NR)
GCM	AES-128	0.65	0.65	64 (NR)
ΘCB	Deoxys-BC-256	0.87	0.87	128 (NR)
SIV	AES-128	1.30*	0.97*	64
GCM-SIV	AES-128	0.95	0.80	64
SCT	Deoxys-BC-256	1.74	1.30	64 (128 for NR)
SIVx	Deoxys-BC-256	1.74	1.30	128
ZAE	Deoxys-BC-256	1.48	1.04	128
ZAE	Deoxys-BC-384	1.58	1.09	128
ZAE	Deoxys-BC-256/Deoxys-BC-384	1.46	1.03	128
ZAE	SKINNY-128-256	6.18	4.12	128
ZAE	SKINNY-128-256	6.38	3.98	128
ZAE	SKINNY-128-256/SKINNY-128-384	5.70	3.64	128

On an Intel Skylake processor Intel Core i5-6600, we measure that for long messages AES-128 runs at 0.65 c/B (cycles/Byte), while Deoxys-BC-256 runs at 0.87 c/B, Deoxys-BC-384 runs at 0.99 c/B, SKINNY-128-256 at 4.12 c/B and SKINNY-128-384 at 4.8 c/B. However, these numbers assume that the tweak input of the ciphers is being used as a counter (as in SCT or SIVx). This can make an important difference depending on the TBC considered, especially for ciphers with a heavy key schedule. One can observe [BJK+16] that when the tweak input is considered random (in opposition to being a counter), there is not much efficiency penalty for SKINNY (probably due to the fact that the best SKINNY implementations use high-parallelism bitslice strategy). For Deoxys-BC, we have implemented a random tweak version and compared it with the case where the tweak is used as a counter. We could observe that in the case of AES-NI implementations a penalty factor on efficiency of 1.4 must be taken in account for Deoxys-BC-256, and a factor 1.8 for Deoxys-BC-384. We emphasize that these penalties will probably not appear for other types of implementations (table or bitslice implementations).

Taking into account all these considerations, we compare ZMAC and ZAE efficiencies with its competitors[13] in Table 3. One can see that ZMAC is the fastest MAC, while providing n-bit security. Moreover, ZAE offers better performances when compared to misuse-resistant competitors, while providing optimal n-bit security, even in nonce-misuse scenario.

It is interesting to note that, as foreseen in previous section, for ZAE the maximum speed might be achieved by using a TBC version with a large tweak for the MAC part, and a TBC version with a small tweak for the encryption part (typically Deoxys-BC-384 for the MAC part and Deoxys-BC-256 for the encryption part). This is because ZMAC really benefits from using a TBC with a large tweak, while the encryption part is not faster when using a TBC with a large tweak (and a TBC with a large tweak is supposed to be slightly slower).

Acknowledgements

The authors would like to thank the anonymous reviewers of CRYPTO 2017 for their helpful comments. The first author is supported by JSPS KAKENHI, Grant-in-Aid for Scientific Research (B), Grant Number 26280045, and the work was carried out in part while visiting Nanyang Technological University, Singapore. The third author is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06) and Temasek Labs (DSOCL16194). The fourth author has been partially supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015.

[13] We can mention that algebraic UHFs such as GHASH would probably perform twice slower for a $2n$-bit output and current best implementation on latest processors show that GHASH costs about 1/2 of an AES call. Therefore, we can estimate that ZHASH instantiated with Deoxys-BC-256 or Deoxys-BC-384 would require a bit more clock cycles than a $2n$-bit version of GHASH, while processing much more data at the same time (ZHASH can handle $n + t$ bit per TBC call).

References

[BI99] Mihir Bellare and Russell Impagliazzo. A tool for obtaining tighter security analyses of pseudorandom function based constructions, with applications to PRP to PRF conversion. IACR Cryptology ePrint Archive, Report 1999/024, 1999. Available at http://eprint.iacr.org/1999/024.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 (Proceedings, Part II)*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BKN09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[BR02] John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 384–397. Springer, 2002.

[BR05] John Black and Phillip Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. *J. Cryptology*, 18(2):111–131, 2005.

[CAE] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. See http://competitions.cr.yp.to/caesar.html.

[CDMS10] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A Domain Extender for the Ideal Cipher. In Daniele Micciancio, editor, *Theory of Cryptography Conference - TCC 2010*, volume 5978 of *LNCS*, pages 273–289. Springer, 2010.

[CLP14] Benoît Cogliati, Rodolphe Lampe, and Jacques Patarin. The Indistinguishability of the XOR of k Permutations. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *LNCS*, pages 285–302. Springer, 2014.

[FLS+10] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. SHA3 Submission to NIST (Round 3), 2010.

[GL15] Shay Gueron and Yehuda Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM Conference on Computer and Communications Security - CCS 2015*, pages 109–119. ACM, 2015.

[GLS+14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM and iSCREAM. Submitted to the CAESAR competition, 2014.

[HP08] Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*, pages 144–161. Springer, 2008.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *Fast Software Encryption - FSE 2003*, volume 2887 of *LNCS*, pages 129–153. Springer, 2003.

[IMV16] Tetsu Iwata, Bart Mennink, and Damian Vizár. CENC is Optimally Secure. 2016. Available at http://eprint.iacr.org/2016/1087.

[Iwa06] Tetsu Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *LNCS*, pages 310–327. Springer, 2006.

[JNP14a] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Deoxys v1. Submitted to the CAESAR competition, 2014.

[JNP14b] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Joltik v1. Submitted to the CAESAR competition, 2014.

[JNP14c] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. KIASU v1. Submitted to the CAESAR competition, 2014.

[JNP14d] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 (Proceedings, Part II)*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014.

[KR11] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *Fast Software Encryption - FSE 2011*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011.

[LN17] Eik List and Mridul Nandi. Revisiting Full-PRF-Secure PMAC and Using It for Beyond-Birthday Authenticated Encryption. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017*, volume 10159 of *LNCS*, pages 258–274. Springer, 2017. Full version at http://eprint.iacr.org/2016/1174.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002.

[Luc00] Stefan Lucks. The Sum of PRPs Is a Secure PRF. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 470–484. Springer, 2000.

[MI15] Kazuhiko Minematsu and Tetsu Iwata. Tweak-Length Extension for Tweakable Blockciphers. In Jens Groth, editor, *Cryptography and Coding - IMACC 2015*, volume 9496 of *LNCS*, pages 77–93. Springer, 2015.

[MI17] Kazuhiko Minematsu and Tetsu Iwata. Cryptanalysis of PMACx, PMAC2x, and SIVx. *IACR Trans. Symmetric Cryptol.*, 2017(2), 2017.

[Min09] Kazuhiko Minematsu. Beyond-Birthday-Bound Security Based on Tweakable Block Cipher. In Orr Dunkelman, editor, *Fast Software Encryption - FSE 2009*, volume 5665 of *LNCS*, pages 308–326. Springer, 2009.

[MV04] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004.

[Nai15] Yusuke Naito. Full PRF-Secure Message Authentication Code Based on Tweakable Block Cipher. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 167–182. Springer, 2015.

[Pat08] Jacques Patarin. A Proof of Security in $O(2^n)$ for the Xor of Two Random Permutations. In Reihaneh Safavi-Naini, editor, *Information Theoretic*

Security - ICITS 2008, volume 5155 of *LNCS*, pages 232–248. Springer, 2008. Full version available at http://eprint.iacr.org/2008/010.

[Pat10] Jacques Patarin. Introduction to Mirror Theory: Analysis of Systems of Linear Equalities and Linear Non Equalities for Cryptography. 2010. Available at http://eprint.iacr.org/2010/287.

[Pat13] Jacques Patarin. Security in $O(2^n)$ for the Xor of Two Random Permutations: Proof with the Standard H Technique. IACR Cryptology ePrint Archive, Report 2013/368, 2013. Available at http://eprint.iacr.org/2013/368.

[PC15] Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks Against Polynomial-Based MAC Schemes. *J. Cryptology*, 28(4):769–795, 2015. Earlier version at FSE 2013.

[PS16] Thomas Peyrin and Yannick Seurin. Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 (Proceedings, Part I)*, volume 9814 of *LNCS*, pages 33–63. Springer, 2016.

[Rog04] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, 2004.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, 2006.

[Yas10] Kan Yasuda. The Sum of CBC MACs Is a Secure PRF. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *LNCS*, pages 366–381. Springer, 2010.

[Yas11] Kan Yasuda. A New Variant of PMAC: Beyond the Birthday Bound. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *LNCS*, pages 596–609. Springer, 2011.