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

Tetsu Iwata*[1] Kazuhiko Minematsu[2]
Thomas Peyrin†[3] Yannick Seurin‡[4]

[1]Nagoya University (Japan) and [2]NEC (Japan)
[3]NTU (Singapore) and [4]ANSSI (France)

CRYPTO 2017, California USA
August 22, 2017

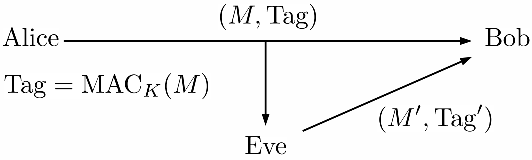
* Supported by JSPS KAKENHI, Grant-in-Aid for Scientific Research (B), Grant Number 26280045

† Supported by Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06) and Temasek Labs (DSOCL16194)

‡ Partially supported by French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015

Introduction: Message Authentication Code (MAC)

- Symmetric-key Crypto for tampering detection
- MAC : $\mathcal{K} \times \{0,1\}^* \to \mathcal{T}$
- Alice computes Tag $=$ MAC$(K, M) =$ MAC$_K(M)$ and sends (M, Tag) to Bob
- Bob checks if (M, Tag) is authentic by computing tag locally
- If MAC$_K(*)$ is a variable-input-length PRF, it is secure

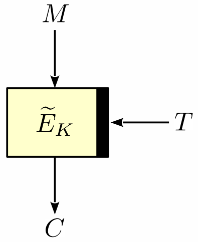


Tweakable Block Cipher (TBC)

Extension of ordinal Block Cipher (BC), formalized by Liskov et al. [LRW02]

- $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$, tweak $T \in \mathcal{T}$ is a public input
- $(K, T) \in \mathcal{K} \times \mathcal{T}$ specifies a permutation over $\mathcal{M}$
- Let $\mathcal{M} = \{0,1\}^n$ and $\mathcal{T} = \{0,1\}^t$

We implicitly assume additional small tweak $i = 1, 2, \ldots,$ used for *domain separation*, and write as $\widetilde{E}^i_K(T, X)$ when necessary



Building TBC

Block cipher modes for TBC: LRW [LRW02] and XEX [Rog04]

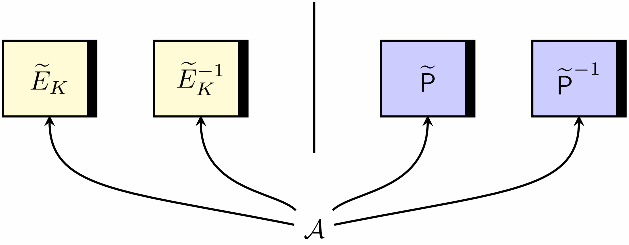
- Efficient but security is up to the birthday bound ($O(2^{64})$ attack when AES is used)
- Beyond-the-birthday-bound (BBB) security is possible (e.g. [Min09][LST12][LS15]) but not really efficient

Dedicated designs:

- HPC [Sch98]
- Threefish in Skein hash function [FLS+10]
- Deoxys-BC, Joltik-BC, KIASU-BC [JNP14a], SCREAM [GLS+14],
  - in the CAESAR submissions
- SKINNY [BJK+16], QARMA [Ava17], ...

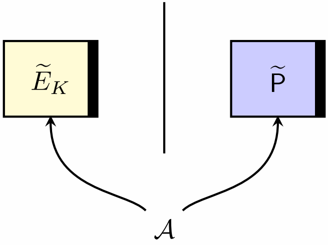
Security notions of TBC [LRW02]

- Indistinguishable from the set of independent uniform random permutations indexed by tweak
  - Tweakable uniform random permutation (TURP) denoted by $\widetilde{\mathsf{P}}$
  - Tweak is chosen by the adversary
- CCA-secure TBC = TSPRP



Security notions of TBC [LRW02]

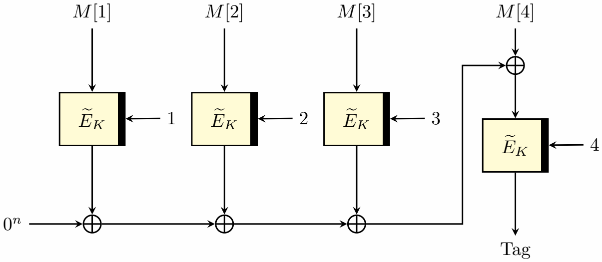
- Indistinguishable from the set of independent uniform random permutations indexed by tweak
  - Tweakable uniform random permutation (TURP) denoted by $\widetilde{\mathsf{P}}$
  - Tweak is chosen by the adversary
- CCA-secure TBC = TSPRP
- CPA-secure TBC = TPRP



Building MAC with TBC : PMAC1

PMAC1 by Rogaway [Rog04], introduced in the proof of PMAC

- Parallel
- Security is up to the birthday bound wrt the block size (n)
  - $\mathtt{Adv}^{\mathtt{tprp}}_{\mathsf{PMAC1}}(\sigma) = O(\sigma^2/2^n)$ for σ queried blocks
  - Thus $n/2$-bit security

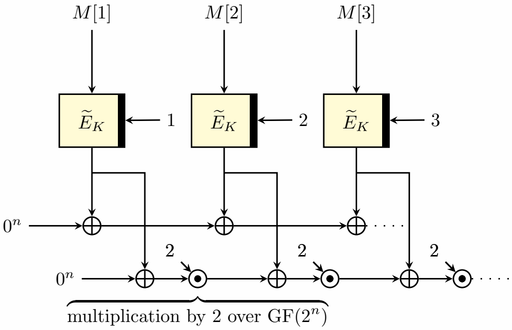


PMAC1

Building MAC with TBC: PMAC_TBC1k

PMAC_TBC1k by Naito [Nai15]

- $2n$-bit chaining similar to PMAC_Plus [Yas11]
  - Finalization by $2n$-bit PRF built from TBC
- BBB-secure: improve security of PMAC1 to n **bits**
- Same computation cost as PMAC1 (except for the finalization)

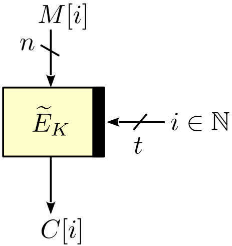


PMAC_TBC1k (message hashing part)

Efficiency of MAC

These TBC-based MACs are **not** optimally efficient

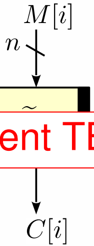
- They process **n-bit input per 1 TBC call**
- t-bit tweak does not process message – reserved for block index



Efficiency of MAC

These TBC-based MACs are **not** optimally efficient

- They process **n-bit input per 1 TBC call**
- t-bit tweak does not process message – reserved for block index



Optimally-efficient TBC-based MAC?

Our proposals: ZMAC ("The MAC") and ZAE

ZMAC is

- The first **optimally efficient** TBC-based MAC
  - $(n+t)$-bit input per 1 TBC call
- Parellel, and **BBB-secure**
  - $\min\{n,(n+t)/2\}$-bit security, e.g. n-bit-secure when $t \geq n$

ZAE is

- An application of ZMAC to Determinisitic Authenticated Encryption (DAE) [RS06]
- **Better efficiency and security than SCT** presented at CRYPTO 2016 [PS16]

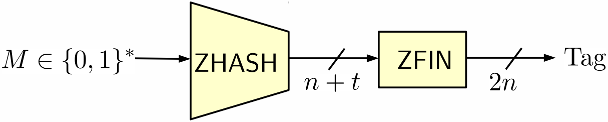
Both using TBC as a sole primitive, and secure if TBC is a TPRP

Structure of ZMAC

A simple composition of message hashing and finalization (Carter-Wegman MAC):

- ZMAC $=$ ZFIN $\circ$ ZHASH
- ZHASH : $\mathcal{M} \rightarrow \{0,1\}^{n+t}$ is a computational universal hash function
- ZFIN : $\{0,1\}^{n+t} \rightarrow \{0,1\}^{2n}$ is a PRF
  - Output truncation if needed

Unified specs for any t ($t = n$ or $t < n$ or $t > n$)

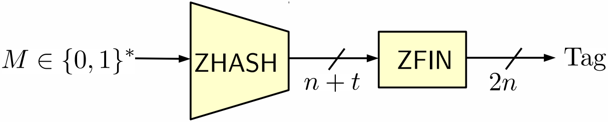


Structure of ZMAC

A simple composition of message hashing and finalization (Carter-Wegman MAC):

- ZMAC $=$ ZFIN $\circ$ ZHASH
- ZHASH : $\mathcal{M} \rightarrow \{0,1\}^{n+t}$ is a computational universal hash function
- ZFIN : $\{0,1\}^{n+t} \rightarrow \{0,1\}^{2n}$ is a PRF
  - Output truncation if needed

Unified specs for any t ($t = n$ or $t < n$ or $t > n$)

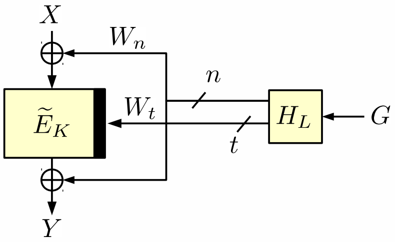


We focus on ZHASH, the most innovative part in ZMAC

How ZHASH works: tweak extension

Optimal efficiency implies t-bit tweak of $\widetilde{E}$ must be extended to incorporate block index
This can be done by XTX [MI15], an extension of LRW and XEX:

- Global tweak $G \in \mathcal{G}$, $|\mathcal{G}| > 2^t$
- Keyed function $H : \mathcal{L} \times \mathcal{G} \to (\{0,1\}^n \times \{0,1\}^t)$
- XTX$[\widetilde{E}, H]_{K,L}(G, X) = \widetilde{E}_K(W_t, W_n \oplus X) \oplus W_n$ with $(W_n, W_t) = H_L(G)$

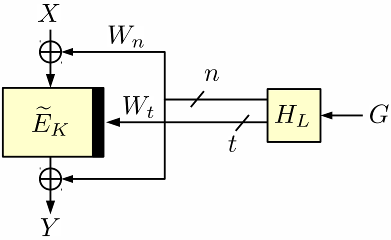


How ZHASH works: security of XTX/XT

XTX is secure if H is ϵ-partial AXU (pAXU) [MI15] :

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

that is, n-bit part is close to differentially uniform and t-bit part has a small collision probability

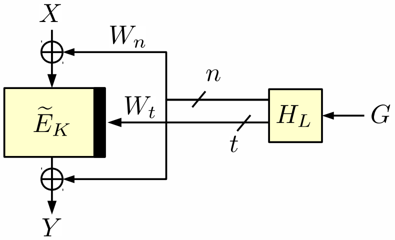


How ZHASH works: security of XTX/XT

In our case, $G \in \underbrace{\{0,1\}^t}_{\text{message part}} \times \underbrace{\mathbb{N}}_{\text{block index}}$ †, and block index is **a counter**

Then XTX can be instantiated and optimized by

- Using the "doubling" trick as XEX
- Omitting the outer mask to Y (as decryption is not needed)



† Omitting domain separation variable

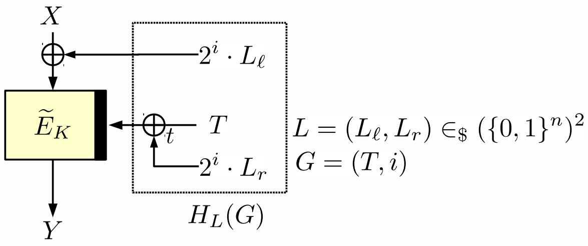
How ZHASH works: security of XTX/XT

The resulting scheme is **XT** , using $H_L(G)$ defined as

$$H_{(L_\ell,L_r)}(T,i) = (2^{i-1}L_\ell, 2^{i-1}L_r \oplus_t T), \text{ using two } n\text{-bit keys } (L_\ell, L_r)$$

Details:

- $2^i X$ is X multiplied by 2 over GF(2^n) for i times
  - Computation is easy by caching $2^{i-1}X$ as done in XEX
- $X \oplus_t Y = \texttt{msb}_t(X) \oplus Y$ if $t \leq n$, $(X \,\|\, 0^{t-n}) \oplus Y$ if $t > n$
  - Chop-or-pad before sum



How ZHASH works: security of XTX/XT

Lemma

Let $\widetilde{\mathsf{P}} : \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a TURP and H is ϵ-pAXU. Then,

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

and our H is $1/2^{n+\min\{n,t\}}$-pAXU. Thus,

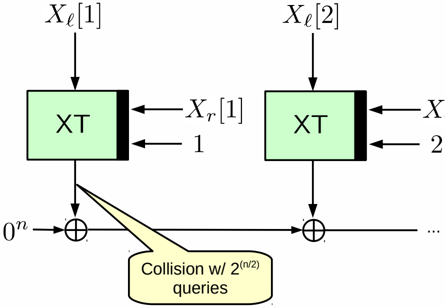
$$\mathtt{Adv}^{\mathtt{tprp}}_{\mathsf{XT}[\widetilde{\mathsf{P}},H]}(q) \leq \frac{q^2}{2^{n+\min\{n,t\}+1}}.$$

Therefore, **XT has $\min\{n, (n+t)/2\}$-bit, BBB-security**

How ZHASH works: chaining scheme

Given XT, it's easy to apply it in the PMAC-like single-chaining hashing scheme

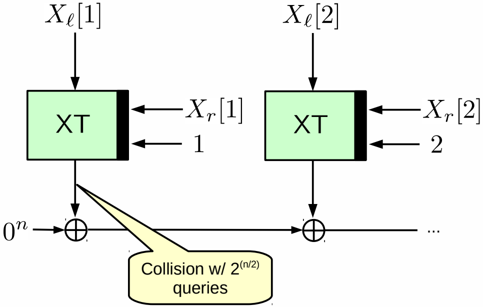
- Message is divided into $(n+t)$-bit blocks, $(X_\ell[i], X_r[i])$ for $i = 1, 2, \ldots$
- This is optimally efficient, but security is up to the birthday bound



How ZHASH works: chaining scheme

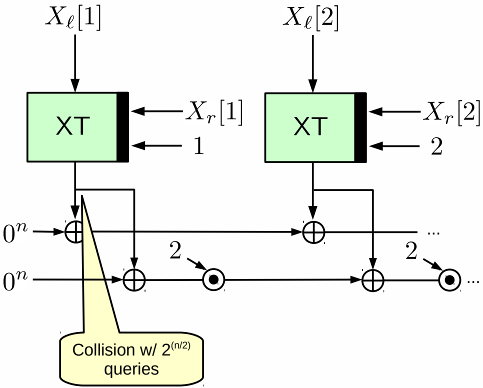
Given XT, it's easy to apply it in the PMAC-like single-chaining hashing scheme

- Message is divided into $(n+t)$-bit blocks, $(X_\ell[i], X_r[i])$ for $i = 1, 2, \ldots$
- This is optimally efficient, but security is up to the birthday bound
- Need a larger chaining value



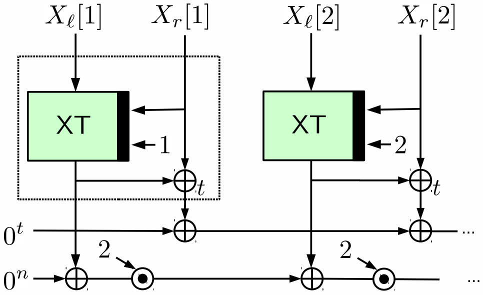
How ZHASH works: chaining scheme

- Naive use of $2n$-bit chaining scheme [Nai15][Yas11] doesn't work
  - XT output collision still breaks the scheme



How ZHASH works: chaining scheme

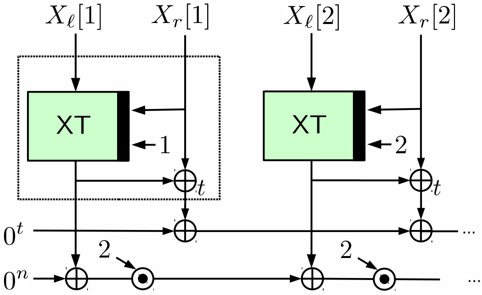
- Key observation: to avoid these collision attacks, the process of (X_ℓ, X_r) (the dotted box) **must be a permutation**
- A Feistel-like **1-round** permutation works ($\mathbb{ZHASH}$)



$\mathbb{ZHASH}$

How ZHASH works: chaining scheme

- Key observation: to avoid these collision attacks, the process of (X_ℓ, X_r) (the dotted box) **must be a permutation**
- A Feistel-like **1-round** permutation works ($\mathbb{ZHASH}$)



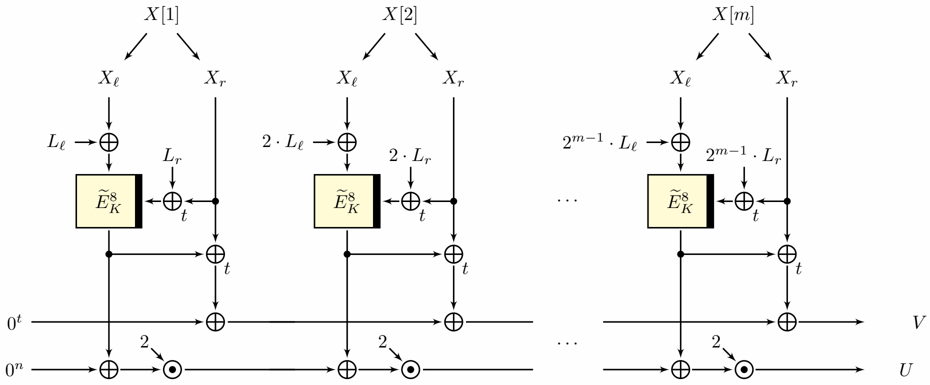
$\mathbb{ZHASH}$

Lemma

$\mathbb{ZHASH}$ (w/ XT using TURP) is ϵ-almost universal for $\epsilon = 4/2^{n+\min\{n,t\}}$

Full ZHASH

Input: $X = (X[1], \ldots, X[m])$, $|X[i]| = n + t$
Output (U, V), $|U| = n$, $|V| = t$

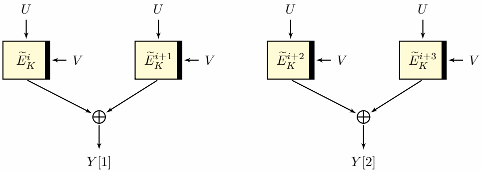


Details:

- $X \oplus_t Y = \mathtt{msb}_t(X) \oplus Y$ if $t \leq n$, $(X \,\|\, 0^{t-n}) \oplus Y$ if $t > n$
- $2 \cdot X$: multiplication by 2
- L_ℓ and L_r : two n-bit masks from $\widetilde{E}_K$ w/ domain separation

ZFIN

ZFIN simply encrypts U with tweak V twice (for each n-bit output) and takes a sum (with domain separation)



PRF security of ZFIN

- ZFIN is essentially "Sum of Permutations" [Luc00, BI99, Pat08a, Pat13, CLP14, MN17]
- From a recent result by Dai et al. [DHT17], ZFIN is **n-bit secure**

Lemma

$$\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZFIN}[\widetilde{\mathsf{P}}]}(q) \leq 2\left(\frac{q}{2^n}\right)^{3/2}$$

Security of ZMAC

Combining all lemmas,

Theorem

For $q \leq 2^{n-4}$ queries of total σ $(n+t)$-bit blocks,

$$\mathtt{Adv}^{\mathtt{prf}}_{\mathsf{ZMAC}[\widetilde{\mathsf{P}}]}(q, \sigma) \leq \frac{2.5\sigma^2}{2^{n+\min\{n,t\}}} + 4\left(\frac{q}{2^n}\right)^{3/2} .$$

Thus ZMAC is $\min\{n, (n+t)/2\}$-bit secure

ZAE deterministic authenticated encryption (DAE)

DAE [RS06] is a class of Authenticated Encryption (AE) with the following features:

- Standard nonce-based AE security when the associated data (AD) contains distinct nonce at encryption
- Best-possible, DAE security even if nonce is repeated (or there is no nonce)
  - Only the repetition of plaintext is leaked
  - Misuse-resistant AE (MRAE)

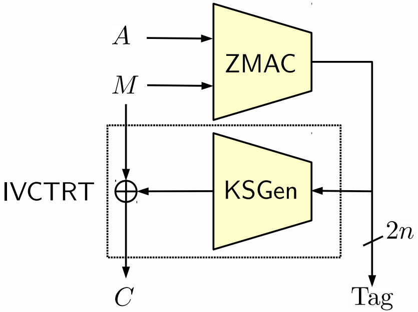
Building ZAE

Following the generic SIV construction, we need

- PRF: $\underbrace{\{0,1\}^*}_{\text{AD}(A)} \times \underbrace{\{0,1\}^*}_{\text{plaintext}(M)} \rightarrow \underbrace{\{0,1\}^{2n}}_{\text{Tag}}$
- (random) IV-based encryption: $\underbrace{\{0,1\}^{2n}}_{\text{Tag=IV}} \times \underbrace{\{0,1\}^*}_{\text{plaintext}(M)} \rightarrow \underbrace{\{0,1\}^*}_{\text{ciphertext}(C)}$

We instantiate

- PRF by ZMAC **with input encoding** for (A, M)
- IV-based enc by (a variant of) IVCTRT [PS16]



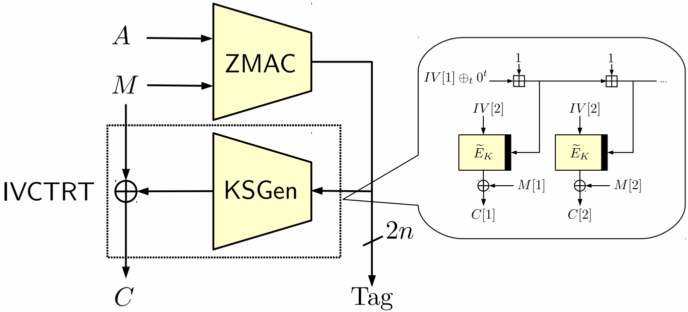
Building ZAE

Following the generic SIV construction, we need

- PRF: $\underbrace{\{0,1\}^*}_{\text{AD}(A)} \times \underbrace{\{0,1\}^*}_{\text{plaintext}(M)} \rightarrow \underbrace{\{0,1\}^{2n}}_{\text{Tag}}$
- (random) IV-based encryption: $\underbrace{\{0,1\}^{2n}}_{\text{Tag=IV}} \times \underbrace{\{0,1\}^*}_{\text{plaintext}(M)} \rightarrow \underbrace{\{0,1\}^*}_{\text{ciphertext}(C)}$

We instantiate

- PRF by ZMAC **with input encoding** for (A, M)
- IV-based enc by (a variant of) IVCTRT [PS16]



Security of ZAE

Security of ZAE: immediate from bounds of ZMAC, SIV, and IVCTRT

Theorem

For total $q \leq 2^{n-4}$ (encryption or decryption) queries and total σ queried blocks in n bits, we have

$$\mathtt{Adv}^{\mathtt{dae}}_{\mathsf{ZAE}[\widetilde{\mathsf{P}}]}(\mathcal{A}) \leq \frac{3.5\sigma^2}{2^{n+\min\{n,t\}}} + 4\left(\frac{q}{2^n}\right)^{3/2} + \frac{q}{2^{2n}}$$

This is better than SCT ($n/2$-bit DAE security)

For example, ZAE with $t = n$ has **n-bit DAE security**

Efficiency of ZAE

Efficiency of ZAE:

- $n(n+t)/(2n+t)$ **input bits per one TBC call**
  - always better than SCT ($n/2$ bits), which uses PMAC1 for MAC
- e.g. $2n/3$ bits for $t = n$, $4n/3$ bits for $t = 2n$

Instantiations of ZMAC and ZAE

We used Deoxys-BC [JNP+14] and SKINNY [BJK+16]

- Deoxys-BC: TBC in the CAESAR candidate Deoxys
  - AES-based, and AESNI can be used
  - 128-bit block, 256 or 384-bit TWEAKEY (Tweak and Key) [JNP+14]
- SKINNY: lightweight 64/128-bit TBC at CRYPTO 2016 [BJK+16]
- TBC performance evaluated under **random tweak**
  - can be slightly slower than counter tweak (depending on the implementation and platform)

Estimated performance examples on Intel Skylake, using AESNI

- Deoxys-BC-256-ZMAC runs at 0.61 c/B
- Deoxys-BC-256-ZAE runs at 1.48 c/B
  - 20 to 30 % gain from other MAC/DAE modes with same TBC
- See the paper for details

Performance considerations

The importance of TBC with large tweak (e.g. $t = 2n$)

- ZMAC operates faster as t grows
- TBC of large t may not be too slow: extending t by n usually does not double the number of rounds

ZAE performance optimization:

- For IVCTRT, $t = n$ is sufficient
- ZAE may be optimized by a combination of large-tweak variant ($t > n$) with small-tweak variant ($t = n$)
  - E.g. Deoxys-BC-384-ZMAC and Deoxys-BC-256-IVCTRT

Concluding remarks

We proposed ZMAC and ZAE, a highly secure and fast MAC and DAE based on TBC.

The power of XEX-like masking:

- We already see it in many blockcipher modes (e.g. PMAC, OCB)
- ZMAC shows it is also powerful for TBC modes
- As dedicated TBCs are becoming popular, this direction looks worth to be further explored

Future topics:

- Other applications (e.g. NAE, RAE or wide-block cipher)
- Even stronger security

Concluding remarks

We proposed ZMAC and ZAE, a highly secure and fast MAC and DAE based on TBC.

The power of XEX-like masking:

- We already see it in many blockcipher modes (e.g. PMAC, OCB)
- ZMAC shows it is also powerful for TBC modes
- As dedicated TBCs are becoming popular, this direction looks worth to be further explored

Future topics:

- Other applications (e.g. NAE, RAE or wide-block cipher)
- Even stronger security

Thank you!