# Cryptanalysis of PMACx, PMAC2x, and SIVx

Kazuhiko Minematsu[1] Tetsu Iwata[2,*]

[1]NEC Corporation, Japan

[2]Nagoya University, Japan

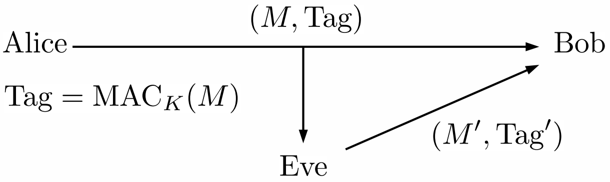
FSE 2018
March 7, 2018, Bruges, Belgium

* Supported in part by JSPS KAKENHI, Grant-in-Aid for Scientific Research (B), 26280045. Work was carried out while visiting Nanyang Technological University, Singapore.

# Introduction : MAC

Message Authentication Code (MAC) : $\mathcal{K} \times \mathcal{M} \to \mathcal{T}$

- Tag $T = \text{MAC}_K(M)$ for message $M$, using key $K$
- If $\text{MAC}_K$ is a PRF, it is a secure MAC




Blockcipher modes of operation for MAC : CBC-MAC, CMAC, etc.

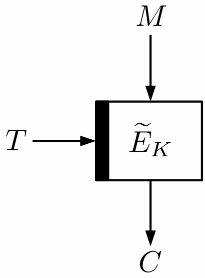
# MACs from TBC

## Recent trend

Use tweakable blockcipher (TBC) for MAC to improve simplicity/efficiency/security

TBC is an extension of ordinal BC, formalized by Liskov et al. [LRW02]

- $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$, tweak $T \in \mathcal{T}$ is a public input
- $(K, T) \in \mathcal{K} \times \mathcal{T}$ specifies a permutation over $\mathcal{M}$



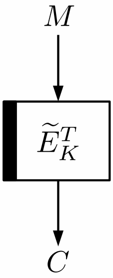
# MACs from TBC

## Recent trend

Use tweakable blockcipher (TBC) for MAC to improve simplicity/efficiency/security

TBC is an extension of ordinal BC, formalized by Liskov et al. [LRW02]

- $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$, tweak $T \in \mathcal{T}$ is a public input
- $(K, T) \in \mathcal{K} \times \mathcal{T}$ specifies a permutation over $\mathcal{M}$



# MACs from TBC

Two TBC-based MACs :

- PMAC1 by Rogaway at Asiacrypt'04 [Rog04] :
  - Simple. Introduced as an abstraction of PMAC for security proof
  - Parallelizable
  - Efficient, $n$ msg bits per 1 $n$-bit-block TBC call
  - Secure up to $2^{n/2}$ queries : *birthday bound* (upBB) security
- PMAC_TBC1k by Naito at ProvSec'15 [Nai15] :
  - Extend the chain value of PMAC1 in a similar to Yasuda's PMAC_plus [Yas11]
  - Parallelizable
  - Efficient, almost the same # of TBC calls as PMAC1
  - Secure up to $2^n$ queries : *beyond birthday bound* (BBB) security

# The proposals of List and Nandi, and our contributions

List and Nandi at CT-RSA'17 [LN17]: refine and extend [Nai15].

- PMAC2x and PMACx for MAC
- SIVx for Deterministic Authenticated Encryption (DAE)

Claimed BBB security for them : secure for $q \ll 2^n$ queries

# The proposals of List and Nandi, and our contributions

List and Nandi at CT-RSA'17 [LN17]: refine and extend [Nai15].

- PMAC2x and PMACx for MAC
- SIVx for Deterministic Authenticated Encryption (DAE)

Claimed BBB security for them : secure for $q \ll 2^n$ queries

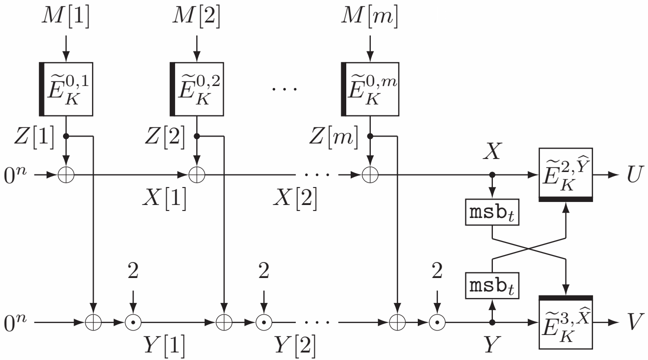
## Our contributions

We invalidate the security claims for all of them,

- by showing attacks w/ $q \approx 2^{n/2}$ queries (thus upBB-secure at best).
- for both distinguisher and (very powerful) forgery

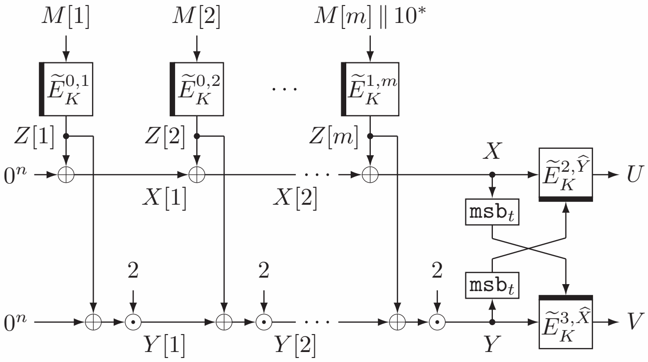
# PMAC2x [LN17]

- Parallel application of TBC to message blocks $M[i]$
- $2n$-bit chain and $2n$-bit output $(U, V)$
- When the last block is **full** ($|M[m]| = n$): no pad



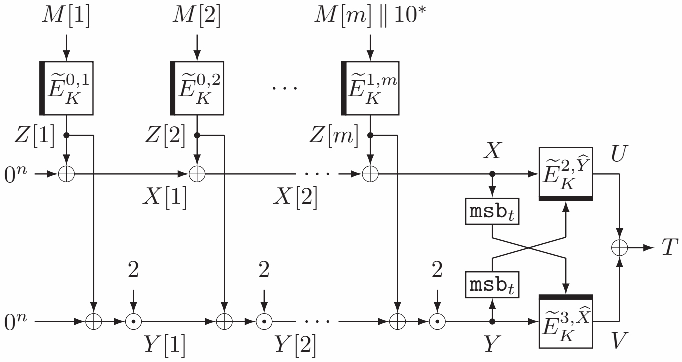
# PMAC2x [LN17]

- Parallel application of TBC to message blocks $M[i]$
- $2n$-bit chain and $2n$-bit output $(U, V)$
- When the last block is **partial** ($|M[m]| < n$): pad and change the tweak of TBC for $M[m]$



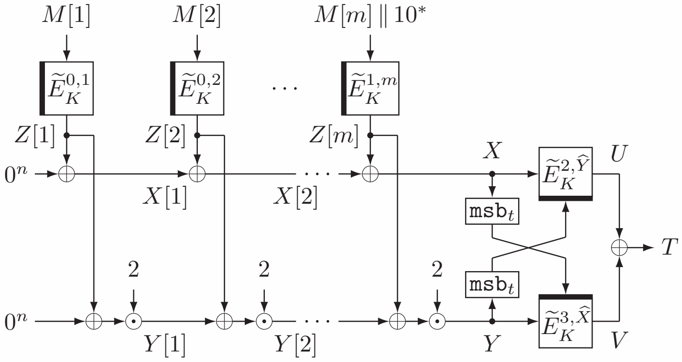
# PMACx [LN17]

- $n$-bit-output variant of PMAC2x obtained by $T = U \oplus V$
- Same handling of last block as PMAC2x



# PMACx [LN17]

- $n$-bit-output variant of PMAC2x obtained by $T = U \oplus V$
- Same handling of last block as PMAC2x




**Security bounds for PMAX2x and PMACx [LN17]**

$O(q^2/2^{2n} + q^3/2^{3n})$, thus BBB-secure

# Differences from PMAC_TBC1k [Nai15]

The structures are the same, but

1. Output extension (from $n$ to $2n$ by PMAC2x), w/o additional cost
2. Refined security bounds
3. More efficient padding
   - PMAC_TBC1k : $M$ is always padded. If $|M| \bmod n = 0$ (integral blocks) we need one more block
   - PMAC2x : $M$ is padded only if $|M| \bmod n \neq 0$.
   - Similar to PMAC1. Improved short-input efficiency

# Differences from PMAC_TBC1k [Nai15]

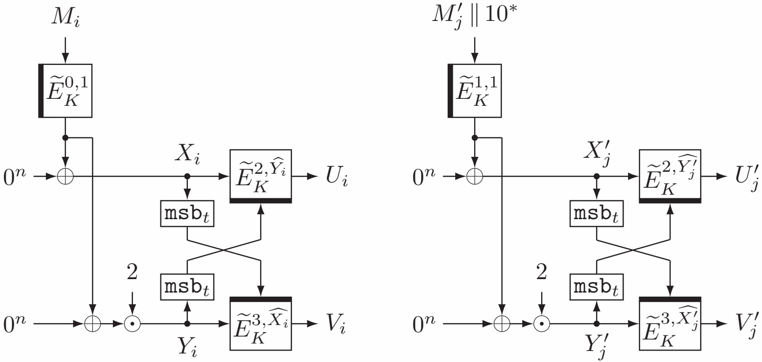
The structures are the same, but

1. Output extension (from $n$ to $2n$ by PMAC2x), w/o additional cost
2. Refined security bounds
3. **More efficient padding**
   - PMAC_TBC1k : $M$ is always padded. If $|M| \bmod n = 0$ (integral blocks) we need one more block
   - PMAC2x : $M$ is padded only if $|M| \bmod n \neq 0$.
   - Similar to PMAC1. Improved short-input efficiency

**The last one seems a nice optimization,
but contains a significant flaw**

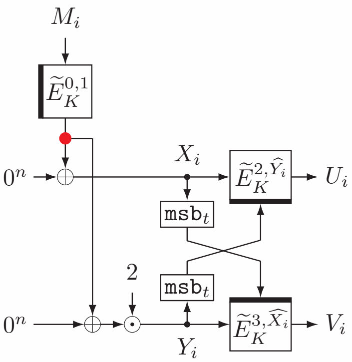
# Birthday attack on PMAC2x

- $Q = 2^{(n/2)-1}$, query $2Q = 2^{n/2}$ single-block messages
- The first set: distinct $M_1, \ldots, M_Q$ s.t. $|M_i| = n$ for $1 \leq i \leq Q$
- The second set: distinct $M'_1, \ldots, M'_Q$ s.t. $|M'_j| < n$ for $1 \leq j \leq Q$



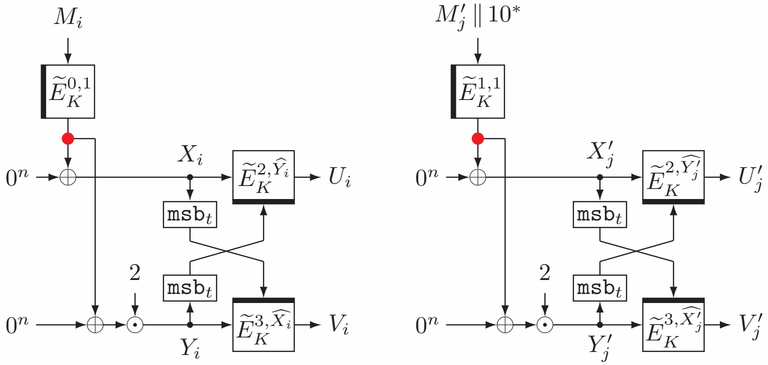
# Birthday attack on PMAC2x

- Two message sets are given to independent random permutations
- Thus, TBC outputs (●) can collide!
- W.H.P., $X_i = X'_j$ for some $i$ and $j$, in which case $Y_i = Y'_j$
- $(U_i, V_i) = (U'_j, V'_j)$ for PMAC2x, but this is unlikely for a random function that outputs $2n$ bits



# Birthday attack on PMAC2x

- Two message sets are given to independent random permutations
- Thus, TBC outputs (•) can collide!
- W.H.P., $X_i = X'_j$ for some $i$ and $j$, in which case $Y_i = Y'_j$
- $(U_i, V_i) = (U'_j, V'_j)$ for PMAC2x, but this is unlikely for a random function that outputs $2n$ bits



# Extension to longer blocks and forgery

The attack can be easily extended to two directions

- Distinguisher for longer messages
  - One can prepend any fixed integer blocks
  - $M_i = M_{\mathtt{pre}} \parallel M_i[m]$ and $M'_j = M_{\mathtt{pre}} \parallel M'_j[m]$, for $|M_{\mathtt{pre}}| = n(m-1)$
  - works because TBC calls for message hashing are parallel
- Almost universal forgery attack
  - Perform the above attack to detect collisions for $M_i$ and $M'_j$
  - Chang the prefix from $M_{\mathtt{pre}}$ to (any integer blocks of )$\hat{M}_{\mathtt{pre}}$
  - Query the tag for $\hat{M}_i = \hat{M}_{\mathtt{pre}} \parallel M_i[m]$
  - The tag for $\hat{M}'_j = \hat{M}_{\mathtt{pre}} \parallel M'_j[m]$ will be the same

# Extension to longer blocks and forgery

The attack can be easily extended to two directions

- Distinguisher for longer messages
  - One can prepend any fixed integer blocks
  - $M_i = M_{\texttt{pre}} \parallel M_i[m]$ and $M'_j = M_{\texttt{pre}} \parallel M'_j[m]$, for $|M_{\texttt{pre}}| = n(m-1)$
  - works because TBC calls for message hashing are parallel
- Almost universal forgery attack
  - Perform the above attack to detect collisions for $M_i$ and $M'_j$
  - Chang the prefix from $M_{\texttt{pre}}$ to (any integer blocks of )$\hat{M}_{\texttt{pre}}$
  - Query the tag for $\hat{M}_i = \hat{M}_{\texttt{pre}} \parallel M_i[m]$
  - The tag for $\hat{M}'_j = \hat{M}_{\texttt{pre}} \parallel M'_j[m]$ will be the same

## Extension to PMACx

The attack can be extended to PMACx with slight modifications

# SIVx : application to DAE

DAE : authenticated encryption (AE) w/o nonce

- introduced by Rogaway and Shrimpton at EUROCRYPT'06 [RS06]
- takes associated data (AD) $A$, plaintext $M$
- outputs ciphertext $C$ and tag $T$

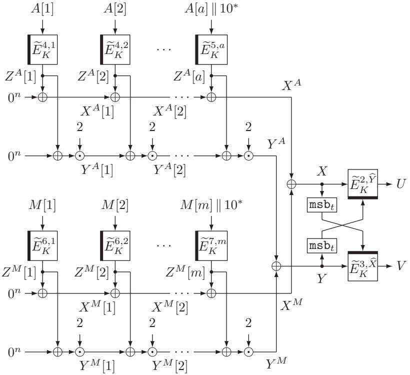
(Generic) SIV [RS06] : DAE construction using PRF $F$ and IV-based encryption $\mathcal{E}$

1. $T \leftarrow F_K(A, M)$
2. $C \leftarrow \mathcal{E}^T_{K'}(M)$ ($T$ as IV)
3. return $(C, T)$

Adopted by many DAE proposals: (BC-based instance of) SIV [RS06], SCT at CRYPTO'16 [PS16], ZAE at CRYPTO'17 [IMPS17]

# SIVx is an instance of SIV

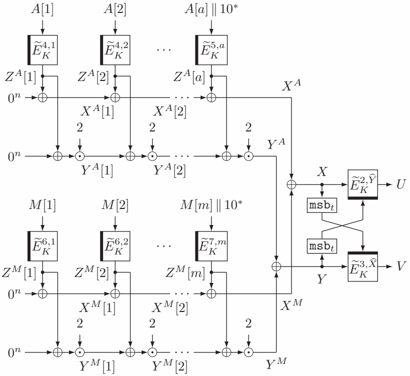
- a variant of PMAC2x as $F$ (vPMAC2x)
  - PHASHx (PMAC2x w/o final TBCs) independently applied to $A$ and $M$, using distinct tweaks
  - Take XOR of outputs, finalize as PMAC2x
- IVCTRT [PS16] as $\mathcal{E}$



# Birthday attack against SIVx

Forgery against vPMAC2x implies forgery against SIVx

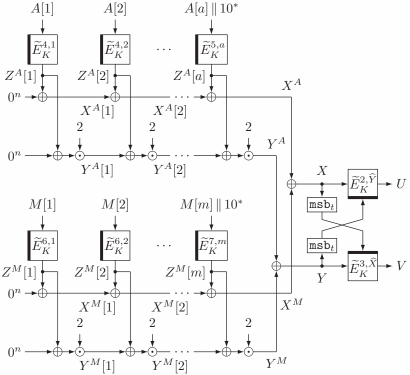
- The padding-based attack works as well as PMAC2x
- E.g. by fixing $M$ and attack AD part



# Birthday attack against SIVx

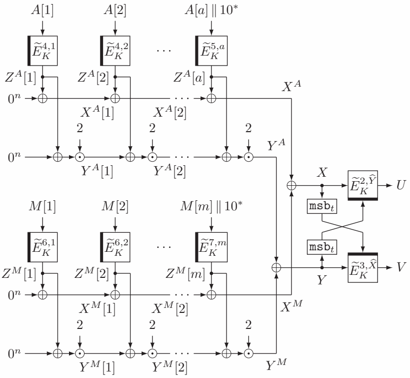
Even if padding is safe (e.g. as PMAC_TBC1k), still vulnerable

- Let $M_i = M_{\mathtt{pre}} \| M_i[m]$, $A_i = A_{\mathtt{pre}} \| A_i[m]$ (the same length)
- Query $(M_1, A_1), \ldots, (M_q, A_q)$ for $q = 2^{n/2}$
- The diff is only in the last blocks
- If $X_i \oplus X_j = 0^n$, $Y_i \oplus Y_j = 2(X_i \oplus X_j) = 0^n$ and the output collides



# Birthday attack against SIVx

- $X_i \oplus X_j = Z_i^A[m] \oplus Z_j^A[m] \oplus Z_i^M[m] \oplus Z_j^M[m]$
- $2^{n/2}$ queries are enough to see a collision on 4 outputs of two independent random permutations
- extension to $a \neq m$ is possible (see the paper)



# Concluding remarks : what went wrong

- (All) Wrong padding method : only useful for upBB-secure schemes
  - Each TBC output for $M[i]$ must be distinct for BBB-security
- (SIVx) Wrong parallel composition (XOR) of PHASHx
  - The cause is mostly from the fact that PHASHx is $O(2^{-2n})$-Almost universal but not $O(2^{-2n})$-Almost XOR universal !
  - (consider the single-block case: collision prob is zero but XOR differential prob is $1/(2^n - 1)$ or $0$)

# Concluding remarks : what went wrong

- (All) Wrong padding method : only useful for upBB-secure schemes
  - Each TBC output for $M[i]$ must be distinct for BBB-security
- (SIVx) Wrong parallel composition (XOR) of PHASHx
  - The cause is mostly from the fact that PHASHx is $O(2^{-2n})$-Almost universal but not $O(2^{-2n})$-Almost XOR universal !
  - (consider the single-block case: collision prob is zero but XOR differential prob is $1/(2^n - 1)$ or $0$)

## Fix on [LN17]

ePrint version of [LN17] fixed them

- Same padding as PMAC_TBC1k
- Encode $(A, M)$ and give to single PMAC2x for SIVx

# Concluding remarks : what went wrong

- (All) Wrong padding method : only useful for upBB-secure schemes
  - Each TBC output for $M[i]$ must be distinct for BBB-security
- (SIVx) Wrong parallel composition (XOR) of PHASHx
  - The cause is mostly from the fact that PHASHx is $O(2^{-2n})$-Almost universal but not $O(2^{-2n})$-Almost XOR universal !
  - (consider the single-block case: collision prob is zero but XOR differential prob is $1/(2^n - 1)$ or $0$)

## Fix on [LN17]

ePrint version of [LN17] fixed them

- Same padding as PMAC_TBC1k
- Encode $(A, M)$ and give to single PMAC2x for SIVx

## Lessons learned

- Be careful when you adopt techniques used in upBB-secure schemes to build BBB-secure schemes!

Thank you!