IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2017, No. 2, pp. 162–176. DOI:10.13154/tosc.v2017.i2.162-176

Cryptanalysis of PMACx, PMAC2x, and SIVx

Kazuhiko Minematsu[1] and Tetsu Iwata[2]

[1] NEC Corporation, Tokyo, Japan
k-minematsu@nec.com
[2] Nagoya University, Nagoya, Japan
tetsu.iwata@nagoya-u.jp

Abstract. At CT-RSA 2017, List and Nandi proposed two variable input length pseudorandom functions (VI-PRFs) called PMACx and PMAC2x, and a deterministic authenticated encryption scheme called SIVx. These schemes use a tweakable block cipher (TBC) as the underlying primitive, and are provably secure up to the query complexity of 2^n, where n denotes the block length of the TBC. In this paper, we falsify the provable security claims by presenting concrete attacks. We show that with the query complexity of $O(2^{n/2})$, i.e., with the birthday complexity, PMACx, PMAC2x, and SIVx are all insecure.

Keywords: Cryptanalysis · PMACx · PMAC2x · SIVx · provable security

1 Introduction

There are several ways to construct a message authentication code (MAC), a pseudorandom function (PRF), or an authenticated encryption with associated data (AEAD) scheme, and a block cipher like AES has been used as one of the main building blocks. In other words, MACs, PRFs, and AEAD schemes can be constructed as a mode of operation of a block cipher. A tweakable block cipher (TBC), put forward by Liskov, Rivest, and Wagner [LRW11], is a generalization of a block cipher that takes additional input called a tweak. It turns out that a TBC is a useful building block to design efficient MACs, PRFs, and AEAD schemes that have high security, particularly in light of the recent development of efficient TBCs, such as Deoxys and Joltik [JNP14], and SKINNY [BJK+16].

Let n be the block length in bits of a block cipher or a TBC. A MAC, a PRF, or an AEAD scheme that are secure up to the $2^{n/2}$ query complexity are often called to be upBB (up to the birthday bound) secure, while a scheme that remains secure beyond $2^{n/2}$ query complexity is often called to have BBB (beyond the birthday bound) security.

At CT-RSA 2017, List and Nandi proposed variable input length PRFs (VI-PRFs) called PMACx and PMAC2x [LN17], based on the work of Naito [Nai15]. These PRFs use a TBC as the underlying primitive, and are provably BBB secure up to the query complexity of 2^n. The output length of PMAC2x is $2n$ bits and that of PMACx is n bits. Based on PMAC2x, List and Nandi also proposed a provably BBB secure deterministic AEAD scheme (DAE) called SIVx [LN17].

In this paper, we show that with the query complexity of $O(2^{n/2})$, PMACx, PMAC2x, and SIVx are all insecure, falsifying the provable security claims. We show that there exist distinguishing attacks against them. We also show that there exist forgery attacks against SIVx. Our attacks on PMACx and PMAC2x exploit the fact that two different tweak values are used to process the last input block depending on its length. In these schemes, the input is padded if the length is not a positive multiple of n bits, otherwise it is not padded. This minimizes the number of TBC calls.

Received: 2017-03-01, Accepted: 2017-05-01, Published: 2017-06-19

Table 1: Summary of our results. In the table, n is the block length of the underlying TBC and q is the number of queries. The attacks against PMACx and PMAC2x are distinguishing attacks. We have both distinguishing and forgery attacks against SIVx.

Scheme	Type	Provable security bound	Attack complexity
PMACx	PRF	$O(q^2/2^{2n} + q^3/2^{3n})$ [LN17]	$q = O(2^{n/2})$ [Sect. 3.2]
PMAC2x	PRF	$O(q^2/2^{2n} + q^3/2^{3n})$ [LN17]	$q = O(2^{n/2})$ [Sect. 3.1]
SIVx	DAE	$O(q^2/2^{2n} + q^3/2^{3n})$ [LN17]	$q = O(2^{n/2})$ [Sect. 5]

While conceptually similar techniques have been employed in upBB-secure block cipher-based MACs [BR00, IK03], for the case of PMACx and PMAC2x, this creates security issues that allow the birthday complexity distinguishing attack. The same distinguishing attack applies to SIVx, and the distinguishing attack can be translated into a forgery attack. Furthermore, we point out that SIVx allows more flexible attacks. See Table 1 for the summary of our results.

We note that other related schemes like PMAC_Plus [Yas11], PMAC_TBC1k [Nai15], and PMAC_TBC3k [Nai15] do not use this type of padding method and do not have the security issue presented in this paper.

2 PMACx and PMAC2x

We first fix notation. Let $\{0,1\}^*$ be the set of all finite bit strings, and for an integer $i \geq 0$, let $\{0,1\}^i$ be the set of all bit strings of i bits. We write $(\{0,1\}^n)^+$ to denote the set of all finite bit strings of length positive multiple of n. For a bit string X, let $|X|$ be its length in bits. We write ε for the empty string. Let n be a block length and let $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable block cipher (TBC), where $\mathcal{K}$ is a non-empty set of keys and $\mathcal{T}$ is a non-empty set of tweaks, and for any $(K, T_W) \in \mathcal{K} \times \mathcal{T}$, $\widetilde{E}(K, T_W, \cdot) = \widetilde{E}_K^{T_W}(\cdot)$ is a permutation over $\{0,1\}^n$. We assume that $\mathcal{T} = \{0,1,2,3\} \times \{0,1\}^t$ for $t = n-2$, and for instance we write $C \leftarrow \widetilde{E}_K^{0,i}(M)$ to mean C is a ciphertext block of M under key K and tweak $(0,i)$, where i is naturally encoded as a t-bit string. Let $0^i \in \{0,1\}^i$ be the i-bit string of all zero, and for two bit strings X and Y, let $X \,\|\, Y$ or simply XY be their concatenation. For a bit string X, let $(X[1], \ldots, X[m]) \xleftarrow{n} X$ be a parsing operation into n-bit blocks. If $X \neq \varepsilon$, then $X[1], \ldots, X[m]$ are unique bit strings such that $X[1] \,\|\, \cdots \,\|\, X[m] = X$, $|X[i]| = n$ for $1 \leq i \leq m-1$, and $1 \leq |X[m]| \leq n$. If $X = \varepsilon$, then we let $X[1] \xleftarrow{n} X$ where $X[1] = \varepsilon$. The set of n-bit strings $\{0,1\}^n$ is regarded as the finite field with 2^n elements $\mathrm{GF}(2^n)$, and for two elements $X, Y \in \mathrm{GF}(2^n)$, $X \cdot Y$ denotes their multiplication with some irreducible polynomial. We often consider the case where X is a generator $X = 2$, e.g., $2 \cdot Y$. For a bit string X s.t. $0 \leq |X| \leq n-1$, we define the one-zero padding function as $\mathtt{ozp}(X) = X \,\|\, 10^{n-|X|-1}$. For $X \in \{0,1\}^n$ and integer $0 \leq i \leq n$, let $\mathtt{msb}_i(X)$ denote the first (leftmost) i bits of X and $\mathtt{lsb}_i(X)$ denote the last (rightmost) i bits of X.

With the above notation, PMACx is a keyed function that uses $\widetilde{E}$ as the underlying primitive. Let $\mathsf{PMACx}[\widetilde{E}] : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^n$ be PMACx with $\widetilde{E}$. It takes arbitrary length $M \in \{0,1\}^*$ as input, and outputs an n-bit string T. We write $T \leftarrow \mathsf{PMACx}[\widetilde{E}_K](M)$ instead of $T \leftarrow \mathsf{PMACx}[\widetilde{E}](K, M)$. Similarly, let $\mathsf{PMAC2x}[\widetilde{E}] : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^{2n}$ be PMAC2x with $\widetilde{E}$. It takes $M \in \{0,1\}^*$ as input and outputs a $2n$-bit string (U, V), and we write $(U, V) \leftarrow \mathsf{PMAC2x}[\widetilde{E}_K](M)$. They are defined in Fig. 1 and illustrated in Fig. 2 and in Fig. 3.

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

1. $X[0] \leftarrow 0^n$, $Y[0] \leftarrow 0^n$
2. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
3. **for** $i = 1$ **to** $m - 1$ **do**
4. $\quad Z[i] \leftarrow \widetilde{E}_K^{0,i}(M[i])$
5. $\quad X[i] \leftarrow X[i-1] \oplus Z[i]$
6. $\quad Y[i] \leftarrow 2 \cdot (Y[i-1] \oplus Z[i])$
7. **if** $|M[m]| = n$ **then**
8. $\quad Z[m] \leftarrow \widetilde{E}_K^{0,m}(M[m])$
9. **else**
10. $\quad Z[m] \leftarrow \widetilde{E}_K^{1,m}(\mathtt{ozp}(M[m]))$
11. $X \leftarrow X[m-1] \oplus Z[m]$
12. $Y \leftarrow 2 \cdot (Y[m-1] \oplus Z[m])$
13. **return** (X, Y)

Algorithm $\mathsf{PMAC2x}[\widetilde{E}_K](M)$

1. $(X, Y) \leftarrow \mathsf{PHASHx}[\widetilde{E}_K](M)$
2. $\widehat{X} \leftarrow \mathtt{msb}_t(X)$
3. $\widehat{Y} \leftarrow \mathtt{msb}_t(Y)$
4. $U \leftarrow \widetilde{E}_K^{2,\widehat{Y}}(X)$
5. $V \leftarrow \widetilde{E}_K^{3,\widehat{X}}(Y)$
6. **return** (U, V)

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

1. $(U, V) \leftarrow \mathsf{PMAC2x}[\widetilde{E}_K](M)$
2. $T \leftarrow U \oplus V$
3. **return** T

Figure 1: Definitions of PMAC2x and PMACx

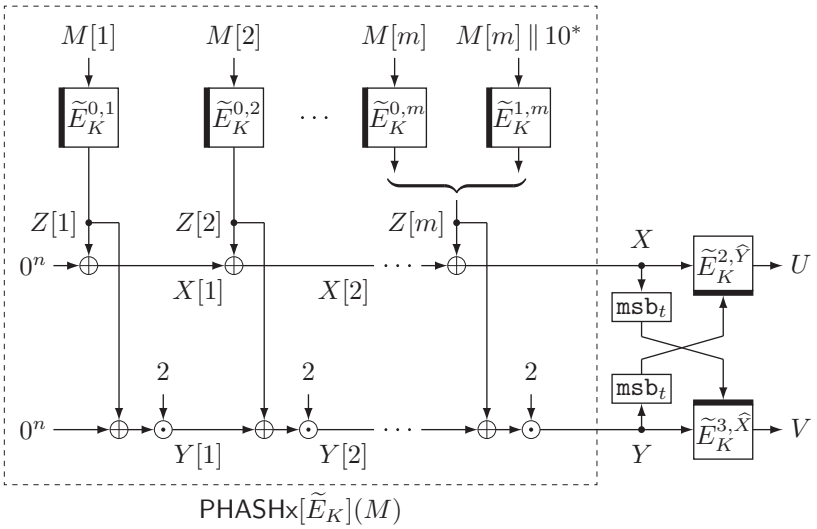

Figure 2: Illustration of PMAC2x. If $|M[m]| = n$, then we let $Z[m] \leftarrow \widetilde{E}_K^{0,m}(M[m])$. Otherwise we let $Z[m] \leftarrow \widetilde{E}_K^{1,m}(\mathtt{ozp}(M[m]))$. The output of $\mathsf{PHASHx}[\widetilde{E}_K](M)$ is (X, Y).

3 Attacks against PMACx and PMAC2x

3.1 Attack against PMAC2x

We present our attack against PMAC2x. Let $\mathcal{O}$ be an oracle which is either $\mathsf{PMAC2x}[\widetilde{E}_K]$ or a random function oracle, which we write \$-oracle, that always returns a $2n$-bit random string. According to [LN17], these two oracles are hard to distinguish unless the number of queries is close to 2^n. However, we show that with a high probability, the adversary can distinguish between $\mathsf{PMAC2x}[\widetilde{E}_K]$ and the \$-oracle with $2^{n/2}$ queries.

For a set of bit strings $\{X_1, \ldots, X_q\}$ for some $q \geq 1$, where $X_i \in \{0,1\}^*$, we say that $\{X_1, \ldots, X_q\}$ is distinct to mean $X_i \neq X_j$ holds for all $1 \leq i < j \leq q$.

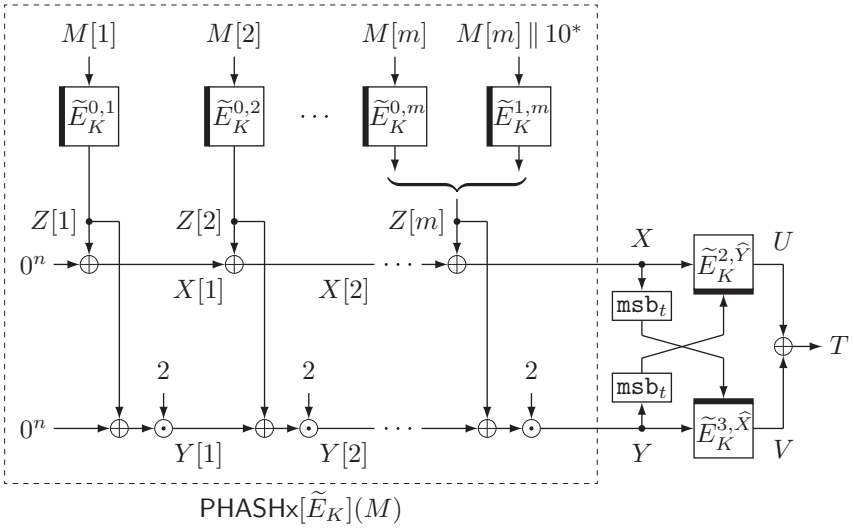

Figure 3: Illustration of PMACx

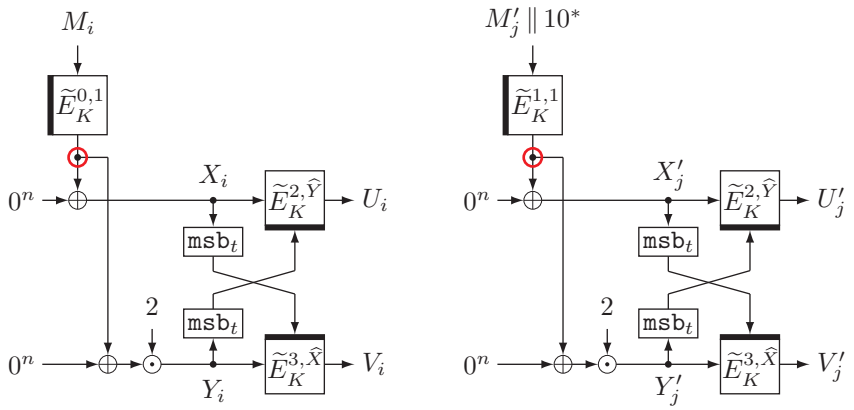

Figure 4: Our attack against PMAC2x. The left figure illustrates Step 1 and the right one illustrates Step 2. We see that the collision between the two red circles implies both $X_i = X'_j$ and $Y_i = Y'_j$.

Now let $Q = 2^{n/2-1}$, and let $M_1, \ldots, M_Q$ be arbitrary bit strings such that $|M_i| = n$ for all $1 \le i \le Q$ and $\{M_1, \ldots, M_Q\}$ is distinct. Similarly, let $M'_1, \ldots, M'_Q$ be arbitrary bit strings such that $1 \le |M'_j| < n$ for all $1 \le j \le Q$ and $\{M'_1, \ldots, M'_Q\}$ is distinct. Our adversary works as follows. See Fig. 4.

1. For $i = 1, \ldots, Q$, query M_i to $\mathcal{O}$ and obtain $(U_i, V_i) \leftarrow \mathcal{O}(M_i)$.
2. For $j = 1, \ldots, Q$, query M'_j to $\mathcal{O}$ and obtain $(U'_j, V'_j) \leftarrow \mathcal{O}(M'_j)$.
3. Search for a collision between $\{(U_1, V_1), \ldots, (U_Q, V_Q)\}$ and $\{(U'_1, V'_1), \ldots, (U'_Q, V'_Q)\}$.
4. If a collision is found, i.e., if $(U_i, V_i) = (U'_j, V'_j)$ holds for some $(i, j) \in \{1, \ldots, Q\}^2$, then $\mathcal{O}$ is $\mathsf{PMAC2x}[\widetilde{E}_K]$. Otherwise $\mathcal{O}$ is a \$-oracle.

We next show that the above attack succeeds in distinguishing between $\mathsf{PMAC2x}[\widetilde{E}_K]$ and the \$-oracle with a high probability.

Case $\mathcal{O}$ = \$-oracle. In this case, for each $(i,j) \in \{1,\dots,Q\}^2$, we have $\Pr[(U_i,V_i) = (U'_j,V'_j)] = 1/2^{2n}$ and the probability to find a collision in Step 3 is negligibly small, which is $\Theta(Q^2/2^{2n}) = \Theta(1/2^n)$.

Case $\mathcal{O}$ = PMAC2x[$\widetilde{E}_K$]. In this case, for M_i and M'_j, let

$$\begin{cases} X_i = \widetilde{E}_K^{0,1}(M_i)\,, \\ Y_i = 2 \cdot X_i\,, \\ X'_j = \widetilde{E}_K^{1,1}(\mathsf{ozp}(M'_j))\,, \text{ and} \\ Y'_j = 2 \cdot X'_j\,. \end{cases}$$

We have $(X_i,Y_i) = \mathsf{PHASHx}[\widetilde{E}_K](M_i)$ and $(X'_j,Y'_j) = \mathsf{PHASHx}[\widetilde{E}_K](M'_j)$. See Fig. 4.

Now we assume that $\widetilde{E}_K$ is perfectly secure, i.e., it behaves ideally. This implies that $X_1,\dots,X_Q$ are non-repeating n-bit random strings, as they are outputs of a random permutation over $\{0,1\}^n$ specified by the tweak $(0,1)$. Similarly, $X'_1,\dots,X'_Q$ are non-repeating n-bit random strings which are outputs of a random permutation specified by the tweak $(1,1)$. Then the probability of collision $X_i = X'_j$ for some $X_i \in \{X_1,\dots,X_Q\}$ and $X'_j \in \{X'_1,\dots,X'_Q\}$ is at least $0.6 \cdot Q^2/2^n$. See for instance [NS90], or [BDJR97] for the closely related derivation of the probability.

Once $X_i = X'_j$ holds for some $(i,j) \in \{1,\dots,Q\}^2$, we also have $Y_i = Y'_j$, and thus $(U_i,V_i) = (U'_j,V'_j)$ holds as well.

Therefore, the attack succeeds with a high probability. More precisely, the advantage in distinguishing the two oracles is at least $0.6 \cdot Q^2/2^n - 1/2^n$.

3.2 Attack against PMACx

We next consider PMACx. Let $\mathcal{O}$ be an oracle which is either $\mathsf{PMACx}[\widetilde{E}_K]$ or a \$-oracle, where this time the \$-oracle always returns an n-bit random string. The above attack cannot be directly applied on PMACx, since the probability of a collision for the \$-oracle is not small for $2^{n/2}$ queries. However, we can generalize the above attack to break PMACx as follows.

Let $Q = 2^{n/2-1}$, and we fix $M[1]$ and $M'[1]$ s.t. $M[1] \neq M'[1]$ and $|M[1]| = |M'[1]| = n$ arbitrarily. Let $M_1[2],\dots,M_Q[2],M'_1[2],\dots,M'_Q[2]$ be arbitrary bit strings s.t. $|M_i[2]| = n$ for all $1 \le i \le Q$, $\{M_1[2],\dots,M_Q[2]\}$ is distinct, $1 \le |M'_j[2]| < n$ for all $1 \le j \le Q$, and $\{M'_1[2],\dots,M'_Q[2]\}$ is distinct. Now our adversary works as follows. See Fig. 5.

1. For $i = 1,\dots,Q$, query $M_i = (M[1],M_i[2])$ to $\mathcal{O}$ and obtain $T_i \leftarrow \mathcal{O}(M_i)$.
2. For $j = 1,\dots,Q$, query $M'_j = (M[1],M'_j[2])$ to $\mathcal{O}$ and obtain $T'_j \leftarrow \mathcal{O}(M'_j)$.
3. Search for a collision between $\{T_1,\dots,T_Q\}$ and $\{T'_1,\dots,T'_Q\}$.
4. Suppose that a collision is found, and suppose that $T_i = T'_j$ holds for $(i,j) \in \{1,\dots,Q\}^2$.
5. For (i,j) found in Step 4, query $M_{Q+1} = (M'[1],M_i[2])$ and $M'_{Q+1} = (M'[1],M'_j[2])$ to $\mathcal{O}$, and obtain $T_{Q+1} \leftarrow \mathcal{O}(M_{Q+1})$ and $T'_{Q+1} \leftarrow \mathcal{O}(M'_{Q+1})$.
6. If $T_{Q+1} = T'_{Q+1}$, then $\mathcal{O}$ is $\mathsf{PMACx}[\widetilde{E}_K]$. Otherwise $\mathcal{O}$ is a \$-oracle.

We show that the above attack succeeds in distinguishing between $\mathsf{PMACx}[\widetilde{E}_K]$ and the \$-oracle.

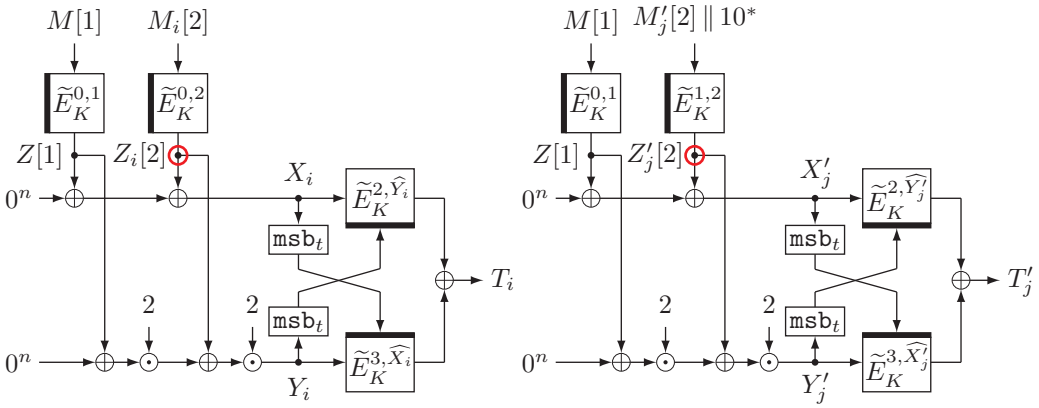

Figure 5: Our attack against PMACx. The attack makes use of a collision between $Z_i[2]$ and $Z'_j[2]$, which are highlighted with the red circles. Both $M[1]$'s in the left and right figures are kept constant during Steps 1 and 2, which are later changed to $M'[1]$ in Step 5.

Case $\mathcal{O}$ = \$-oracle. In this case, with a high probability, we find $(i, j) \in \{1, \ldots, Q\}^2$ in Step 4, but for the corresponding T_{Q+1} and T'_{Q+1} in Step 5, we have $\Pr[T_{Q+1} = T'_{Q+1}] = 1/2^n$.

Case $\mathcal{O}$ = PMACx[$\widetilde{E}_K$]. We follow the notation in Fig. 5, i.e., for M_i and M'_j, let

$$\begin{cases} Z[1] = \widetilde{E}_K^{0,1}(M[1])\,, \\ Z_i[2] = \widetilde{E}_K^{0,2}(M_i[2])\,, \text{ and} \\ Z'_j[2] = \widetilde{E}_K^{1,2}(\mathtt{ozp}(M'_j[2]))\,. \end{cases}$$

Let $(X_i, Y_i) = \mathsf{PHASHx}[\widetilde{E}_K](M_i)$ and $(X'_j, Y'_j) = \mathsf{PHASHx}[\widetilde{E}_K](M'_j)$. We have $X_i = Z[1] \oplus Z_i[2]$, $Y_i = 4 \cdot Z[1] \oplus 2 \cdot Z_i[2]$, $X'_j = Z[1] \oplus Z'_j[2]$, and $Y'_j = 4 \cdot Z[1] \oplus 2 \cdot Z'_j[2]$.

Then it holds that

$$\mathsf{PHASHx}[\widetilde{E}_K](M_i) \oplus \mathsf{PHASHx}[\widetilde{E}_K](M'_j) = (Z_i[2] \oplus Z'_j[2], 2 \cdot (Z_i[2] \oplus Z'_j[2]))\,.$$

Here, $Z_1[2], \ldots, Z_Q[2]$ are Q non-repeating n-bit random strings, and $Z'_1[2], \ldots, Z'_Q[2]$ are also Q non-repeating n-bit random strings. Then as in the attack against PMAC2x, with a high probability, we have $(i, j) \in \{1, \ldots, Q\}^2$ such that $Z_i[2] \oplus Z'_j[2] = 0^n$, in which case we have $\mathsf{PHASHx}[\widetilde{E}_K](M_i) \oplus \mathsf{PHASHx}[\widetilde{E}_K](M'_j) = (0^n, 0^n)$. We see that changing $M[1]$ to $M'[1]$ does not prevent having a collision, and we always have $T_{Q+1} = T'_{Q+1}$ in Step 7.

Therefore, the attack succeeds with a high probability, as the distinguishing advantage is at least $0.6 \cdot Q^2/2^n - 1/2^n$.

3.3 Remarks on the Attacks against PMACx and PMAC2x

3.3.1 Messages Can Be Longer

We first remark that $M[1]$ and $M'[1]$ in the attack of PMACx can be longer. Specifically, we can fix $M[1], \ldots, M[m-1] \in \{0,1\}^n$ arbitrarily, and then perform Steps 1–4 in Sect. 3.2 using $M_i = (M[1], \ldots, M[m-1], M_i[m])$ and $M'_j = (M[1], \ldots, M[m-1], M'_j[m])$ to find a collision, and then substitute $M[1], \ldots, M[m-1]$ with $M'[1], \ldots, M'[m-1]$, where $(M[1], \ldots, M[m-1]) \neq (M'[1], \ldots, M'[m-1])$, to perform Steps 6 and 7.

The same is true for PMAC2x. We may fix $M[1], \ldots, M[m-1] \in \{0,1\}^n$ arbitrarily, and then perform Steps 1–4 in Sect. 3.1 using $M_i = (M[1], \ldots, M[m-1], M_i[m])$ and $M'_j = (M[1], \ldots, M[m-1], M'_j[m])$ to find a collision.

3.3.2 Almost Universal Forgery

The above remark about PMACx suggests that an almost universal forgery is possible. Here the almost universal forgery is a type of forgery where the adversary is given $M^* \in \{0,1\}^*$, and the goal is to output $(M^* \,\|\, S, T^*)$ for some $S \in \{0,1\}^*$, where $T^* = \mathsf{PMACx}[\widetilde{E}_K](M^* \,\|\, S)$, without making a query $M^* \,\|\, S$, i.e., the adversary is requested to produce a correct output for an input that has M^* as the prefix.

This is possible simply by using $\mathtt{ozp}(M^*)$ as $M'[1], \ldots, M'[m-1]$ in the above remark, where we define $\mathtt{ozp}(X) = X \,\|\, 10^{n-(|X| \bmod n)-1}$ when $|X| \geq n$. Specifically, after obtaining colliding $M_i[m]$ and $M'_j[m]$ in Steps 1–4, the adversary makes a query $\mathtt{ozp}(M^*) \,\|\, M_i[m]$ to obtain T_{Q+1}, and the forgery is $(\mathtt{ozp}(M^*) \,\|\, M'_j[m], T_{Q+1})$. Since $|\mathtt{ozp}(M^*)|$ is a multiple of n, $M_i[m]$ and $M'_j[m]$ are again used as the last input blocks, and thus the forgery is always accepted.

It is straightforward to see that the almost universal forgery is possible against PMAC2x by following the attack in Sect. 3.3.1 that uses a collision of $M_i = (M[1], \ldots, M[m-1], M_i[m])$ and $M'_j = (M[1], \ldots, M[m-1], M'_j[m])$ for $m \geq 2$.

3.3.3 More Colliding Input Pairs

In the attack against PMACx in Sect. 3.2, once we obtain a colliding input pair $M_i = (M[1], M_i[2])$ and $M'_j = (M[1], M'_j[2])$ in Steps 1–4, we obtain another colliding input pair $M_{Q+1} = (M'[1], M_i[2])$ and $M'_{Q+1} = (M'[1], M'_j[2])$. It is easy to see that more colliding input pairs can be obtained by changing $M'[1]$.

It is also easy to see that more colliding input pairs can be obtained in PMAC2x by following the attack mentioned in Sect. 3.3.1.

4 SIVx

SIVx is a deterministic AEAD (DAE for short) scheme that uses a PRF and an IV-based encryption scheme called IVCTRT proposed by Peyrin and Seurin [PS16]. These two functions are composed in the same way as SIV [RS06]. Both functions use a tweakable block cipher, $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$. Following [LN17], we assume that $\mathcal{T} = \{0,1,2,\ldots,7\} \times \{0,1\}^t$ for $t = n-4$, although $t = n-3$ is sufficient. The PRF internally used in SIVx takes $(A, M) \in (\{0,1\}^*)^2$ as a vector input to produce a $2n$-bit tag. Although this vector-input PRF is based on PMAC2x, and in fact [LN17] called it PMAC2x, the specification is crucially different. To distinguish them we here write vPMAC2x to denote the PRF inside SIVx.

Now, vPMAC2x applies PHASH to A and M in parallel, using different tweak sets, and taking XOR of two PHASH outputs, then performs the same final processing as in the original PMAC2x to output a $2n$-bit tag T. For encryption, IVCTRT takes T as its IV and performs additive encryption for M. SIVx is defined in Fig. 6, and vPMAC2x is illustrated in Fig. 7. The encryption algorithm $\mathsf{SIVx}[\widetilde{E}_K](\cdot,\cdot)$ takes associated data (AD) A and a plaintext M as input and outputs a ciphertext C and a tag T. The decryption algorithm $\mathsf{SIVx}[\widetilde{E}_K]^{-1}(\cdot,\cdot,\cdot)$ takes A, C, T as the input and outputs M or $\perp$ indicating that the input is invalid.

In our attacks, we will use the fact that IVCTRT encrypts M additively, but we remark that further details of IVCTRT are irrelevant to our attacks.

Algorithm $\mathsf{SIVx}[\widetilde{E}_K](A, M)$

1. $T \leftarrow \mathsf{vPMAC2x}[\widetilde{E}_K](A, M)$
2. $IV \leftarrow (\mathtt{msb}_t(T) \,\|\, \mathtt{lsb}_n(T))$
3. $C \leftarrow \mathsf{IVCTRT}[\widetilde{E}_K](IV, M)$
4. **return** (C, T)

Algorithm $\mathsf{SIVx}^{-1}[\widetilde{E}_K](A, C, T)$

1. $IV \leftarrow (\mathtt{msb}_t(T) \,\|\, \mathtt{lsb}_n(T))$
2. $M \leftarrow \mathsf{IVCTRT}^{-1}[\widetilde{E}_K](IV, C)$
3. $\widehat{T} \leftarrow \mathsf{vPMAC2x}[\widetilde{E}_K](A, M)$
4. **if** $\widehat{T} = T$ **then return** M
5. **else return** $\perp$

Algorithm $\mathsf{vPMAC2x}[\widetilde{E}_K](A, M)$

1. $(X^A, Y^A) \leftarrow \mathsf{PHASHx}^{4,5}[\widetilde{E}_K](A)$
2. $(X^M, Y^M) \leftarrow \mathsf{PHASHx}^{6,7}[\widetilde{E}_K](M)$
3. $X \leftarrow X^A \oplus X^M$
4. $Y \leftarrow Y^A \oplus Y^M$
5. $\widehat{X} \leftarrow \mathtt{msb}_t(X)$
6. $\widehat{Y} \leftarrow \mathtt{msb}_t(Y)$
7. $U \leftarrow \widetilde{E}_K^{2,\widehat{Y}}(X)$
8. $V \leftarrow \widetilde{E}_K^{3,\widehat{X}}(Y)$
9. $T \leftarrow (U \,\|\, V)$
10. **return** T

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

1. $I \leftarrow \mathtt{msb}_t(IV)$, $J \leftarrow \mathtt{lsb}_n(IV)$
2. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
3. **for** $i = 1$ **to** $m - 1$ **do**
4. $\quad C[i] \leftarrow \widetilde{E}_K^{1,I+(i-1)}(J) \oplus M[i]$
5. $S[m] \leftarrow \widetilde{E}_K^{1,I+(m-1)}(J)$
6. $C[m] \leftarrow \mathtt{msb}_{|M[m]|}(S[m]) \oplus M[m]$
7. $C \leftarrow (C[1] \,\|\, \cdots \,\|\, C[m])$
8. **return** C

Algorithm $\mathsf{IVCTRT}^{-1}[\widetilde{E}_K](IV, C)$

1. $M \leftarrow \mathsf{IVCTRT}[\widetilde{E}_K](IV, C)$
2. **return** M

Figure 6: Definition of SIVx, following the description of [LN17]. In the above definition, $\mathsf{PHASHx}^{i,j}[\widetilde{E}_K]$ is a variant of $\mathsf{PHASHx}[\widetilde{E}_K]$ defined in Fig. 1 where $\widetilde{E}_K^{i,*}$ and $\widetilde{E}_K^{j,*}$ are used instead of $\widetilde{E}_K^{0,*}$ and $\widetilde{E}_K^{1,*}$. We note that $\mathsf{vPMAC2x}[\widetilde{E}_K](A, M)$ above was written as $\mathsf{PMAC2x}[\widetilde{E}_K](A, M)$ by List and Nandi. We also note that there are two inconsistencies in the pseudocode and the figure of [LN17]. First, in vPMAC2x, the tweak value $(4, 5)$ is used for A and $(6, 7)$ used for M in the pseudocode, however, this is swapped in the figure. Second, the finalization of vPMAC2x in the figure of [LN17] uses X and Y as the block inputs to TBC. However, X^M and Y^M are used in the corresponding pseudocode. Our definition follows the figure.

5 Attacks against SIVx

5.1 Padding Attack against SIVx

Before describing our attacks, we briefly describe the privacy and authenticity notions for DAE [RS06]. Let $\mathsf{SIVx}[\widetilde{E}_K]$ and $\mathsf{SIVx}^{-1}[\widetilde{E}_K]$ be the encryption and decryption oracles of SIVx defined as Fig. 6. The privacy notion is the indistinguishability of two games, $\mathcal{O}_e = \mathsf{SIVx}[\widetilde{E}_K]$ and $\mathcal{O}_e =$ \$-oracle, using non-repeating encryption queries to $\mathcal{O}_e$. Here, the \$-oracle returns a random string of length $|M| + 2n$ bits on query (A, M). The authenticity notion is the probability of successful forgery, using encryption queries to $\mathcal{O}_e = \mathsf{SIVx}[\widetilde{E}_K]$ and decryption queries to $\mathcal{O}_d = \mathsf{SIVx}^{-1}[\widetilde{E}_K]$. Here, a forgery means an answer from $\mathsf{SIVx}^{-1}[\widetilde{E}_K]$ which is not $\perp$, and we exclude queries leading to a trivial win (e.g. a decryption query (A, C, T) after an encryption query (A, M) and the corresponding answer (C, T)). A break of privacy notion means a high advantage (the absolute difference in the probabilities of decision that $\mathcal{O}_e$ is $\mathsf{SIVx}[\widetilde{E}_K]$ after queries) in distinguishing two

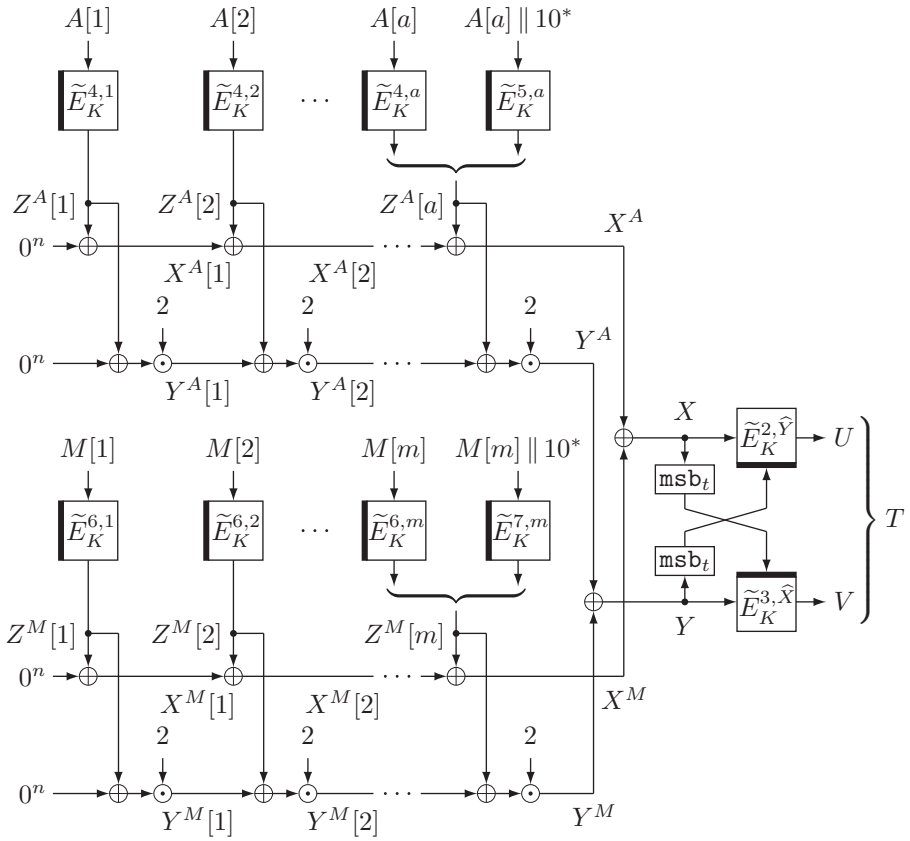

Figure 7: vPMAC2x, the PRF of SIVx

games. A break of authenticity notion means a high probability in a successful forgery. We also say a distinguishing attack to mean an attack against the privacy notion, and a forgery attack to mean an attack against the authenticity notion.

We first point out that the attack presented in Sect. 3.1, which exploits the padding, also works on SIVx. For attacking the privacy notion, we let $Q = 2^{n/2-1}$ and fix arbitrary $M \in \{0,1\}^*$, $A \in (\{0,1\}^n)^+$, $A_1, \ldots, A_Q$, and $A'_1, \ldots, A'_Q$, where $|A_i| = n$, $\{A_1, \ldots, A_Q\}$ is distinct, $1 \leq |A'_j| < n$, and $\{A'_1, \ldots, A'_Q\}$ is distinct. After making $2Q$ queries of $((A, A_1), M), \ldots, ((A, A_Q), M)$ and $((A, A'_1), M), \ldots, ((A, A'_Q), M)$ to $\mathcal{O}_e$, the adversary obtains $(C_1, T_1), \ldots, (C_Q, T_Q)$ and $(C'_1, T'_1), \ldots, (C'_Q, T'_Q)$. With a high probability, we find a collision between $\{T_1, \ldots, T_Q\}$ and $\{T'_1, \ldots, T'_Q\}$ which is not the case for a \$-oracle.

Similarly, forgery attacks are possible. After finding colliding T_i and T'_j with the above distinguishing attack, with the corresponding $((A, A_i), M)$ and $((A, A'_j), M)$, the adversary replaces A by any A' s.t. $A' \neq A$ and $|A'| = |A|$, makes a query $((A', A_i), M)$ to $\mathcal{O}_e$ to obtain (C', T'), and then sends a forgery $((A', A'_j), C', T')$ to $\mathcal{O}_d$. The forgery is always accepted.

Therefore, there exist attacks against SIVx with a query complexity of $O(2^{n/2})$ for both privacy and authenticity notions.

5.2 Another Attack against SIVx

We next present a set of attacks different from that of Sect. 5.1. These attacks have the same complexity as before, however, they do not rely on the flaw of the padding, and are

more flexible for the input blocks to be modified. We first describe a distinguishing attack.

Overview of the Attack for the Case $a = m$. Starting with initial AD of a blocks and an initial plaintext of m blocks, where $a = m$, we alter the last block in the AD and that in the plaintext at the same time. The first $a-1$ blocks of AD and the plaintexts are kept constant, and we make $Q = 2^{n/2}$ of such queries, each with a different value for the last AD and plaintext blocks. The i-th query (A_i, M_i) is of the form

$$\begin{cases} A_i = (A[1], \ldots, A[a-1], \langle i \rangle) \\ M_i = (M[1], \ldots, M[a-1], \langle i \rangle) \end{cases}$$

where $i = 1, \ldots, Q$, $\langle i \rangle$ is the n-bit encoding of i, and $A[1], \ldots, A[a-1], M[1], \ldots, M[a-1]$ are fixed. We are then going to look for a pair of queries that share the same value for T. If we find such a pair, it means that we have distinguished SIVx from an ideal AEAD scheme.

This attack works for the reason below. We observe that if the two distinct queries have the same value of X and Y, then we also have the same value of T. Let X_i, Y_i, and T_i be the values of X, Y, and T for the i-th query (A_i, M_i), respectively. We also use the same notation for other internal variables.

Now consider two queries (A_i, M_i) and (A_j, M_j) for distinct $i, j \in \{1, \ldots, Q\}$. In order to have $X_i = X_j$, we need the XOR difference between $\widetilde{E}_K^{4,a}(\langle i \rangle)$ and $\widetilde{E}_K^{4,a}(\langle j \rangle)$ to be the same as the XOR difference between $\widetilde{E}_K^{6,a}(\langle i \rangle)$ and $\widetilde{E}_K^{6,a}(\langle j \rangle)$. See Fig. 8 illustrating the XOR difference of internal variables for (A_i, M_i) and (A_j, M_j). If the differences are the same, then they will cancel out, and we have $X_i = X_j$. The probability of this event is about 2^{-n} per pair of queries, thus with $Q = 2^{n/2}$ queries, we have a good chance of seeing a pair for which $X_i = X_j$ is true.

We see that once $X_i = X_j$ holds, then this implies $Y_i = Y_j$, and hence we have $T_i = T_j$, which is unlikely to hold for the case of an ideal AEAD scheme.

Concrete Attack for the Case $a = m$. We now present the concrete attack below. Recall that $\langle i \rangle$ is the n-bit encoding of i, and we let $Q = 2^{n/2}$.

1. Fix $(A[1], \ldots, A[a-1]) \in (\{0,1\}^n)^+$ and $(M[1], \ldots, M[m-1]) \in (\{0,1\}^n)^+$ arbitrarily, where $a = m$.
2. For $i = 1, \ldots, Q$, let $A_i = (A[1], \ldots, A[a-1], \langle i \rangle)$ and $M_i = (M[1], \ldots, M[a-1], \langle i \rangle)$, and query (A_i, M_i) to $\mathcal{O}_e$ to obtain $(C_i, T_i) \leftarrow \mathcal{O}_e(A_i, M_i)$.
3. Search for a collision among $\{T_1, \ldots, T_Q\}$.
4. If a collision is found, i.e., if $T_i = T_j$ holds for some distinct $i, j \in \{1, \ldots, Q\}$, then $\mathcal{O}_e$ is $\mathsf{SIVx}[\widetilde{E}_K]$.

Let us analyze the success probability of the above attack.

Case $\mathcal{O}_e$ = \$-oracle. In this case, for any distinct $i, j \in \{1, \ldots, Q\}$, we have $\Pr[T_i = T_j] = 1/2^{2n}$ and the probability to find a collision in Step 4 is $\Theta(Q^2/2^{2n}) = \Theta(1/2^n)$.

Case $\mathcal{O}_e = \mathsf{SIVx}[\widetilde{E}_K]$. For any distinct $i, j \in \{1, \ldots, Q\}$, we have

$$\begin{aligned} X_i \oplus X_j &= Z_i^A[a] \oplus Z_j^A[a] \oplus Z_i^M[a] \oplus Z_j^M[a] \\ &= \widetilde{E}_K^{4,a}(\langle i \rangle) \oplus \widetilde{E}_K^{4,a}(\langle j \rangle) \oplus \widetilde{E}_K^{6,a}(\langle i \rangle) \oplus \widetilde{E}_K^{6,a}(\langle j \rangle)\,, \\ Y_i \oplus Y_j &= 2 \cdot Z_i^A[a] \oplus 2 \cdot Z_j^A[a] \oplus 2 \cdot Z_i^M[a] \oplus 2 \cdot Z_j^M[a] \\ &= 2 \cdot (X_i \oplus X_j)\,, \end{aligned}$$

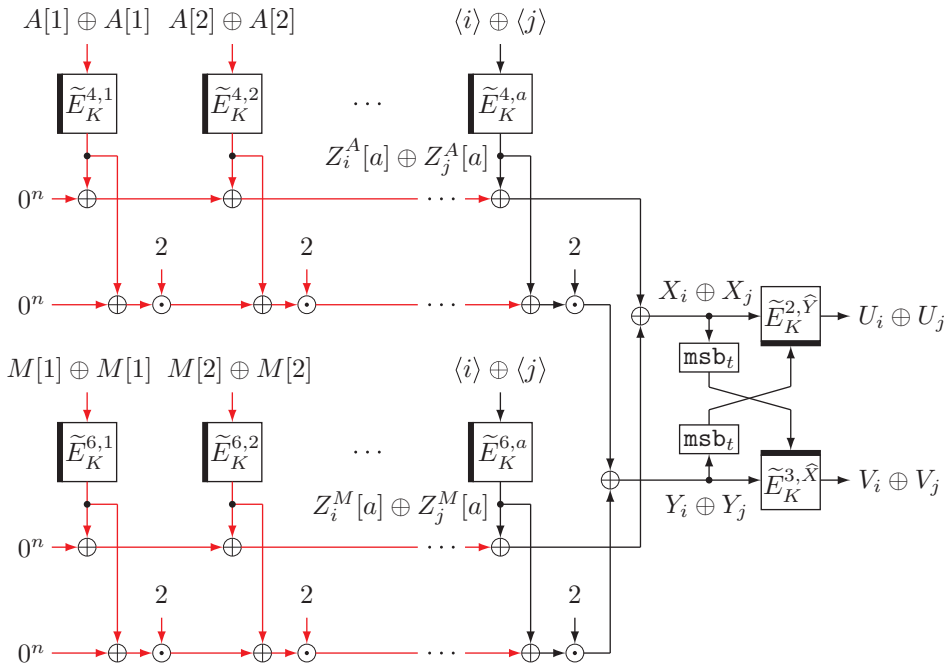

Figure 8: The attack against SIVx. The figure illustrates the XOR difference of internal variables for (A_i, M_i) and (A_j, M_j). $A[1], \ldots, A[a-1], M[1], \ldots, M[a-1]$ are fixed, and the red lines indicate the zero difference.

and the probability of $X_i \oplus X_j = 0^n$ for some $1 \le i < j \le Q$ is not small, assuming $\widetilde{E}_K$ is ideal as in the previous attacks. In more detail, for two independent n-bit random permutations $P_1 = \widetilde{E}_K^{4,a}$ and $P_2 = \widetilde{E}_K^{6,a}$, we observe that finding a collision in $\{X_1, \ldots, X_Q\}$ is equivalent to finding a collision on the outputs of a function defined as $x \to P_1(x) \oplus P_2(x)$, i.e., the sum of two permutations which we call SUM2. We write $p_1(q)$ to denote the maximum probability of finding a collision for SUM2 using (possibly adaptive) q queries. What we need is a lower bound of $p_1(Q)$. By replacing the function $x \to P_1(x) \oplus P_2(x)$ with a random function $R : \{0,1\}^n \to \{0,1\}^n$, the above task is reduced to finding a collision on the output of R. Let $p_2(q)$ be the maximum probability of finding a collision for R with q queries. Then $|p_1(q) - p_2(q)|$ is no greater than the distinguishing advantage between SUM2 and R, which is at most $q^3/(3 \cdot 2^{2n-1})$ from Lucks [Luc00]. Therefore, $p_1(q) \ge p_2(q) - q^3/(3 \cdot 2^{2n-1})$ holds. Now it is standard that $p_2(q)$ is at least $0.3 \cdot q(q-1)/2^n$. See for instance [BDJR97]. Therefore, $p_1(q)$ is at least about $0.3 - 1/2^{n/2}$ when $q = 2^{n/2}$.

Extensions. We next present extensions of the above mentioned attack so that the AD and the plaintext can be different in length, and the blocks to alter do not have to be the last ones.

We start with AD of a blocks and a plaintext of m blocks, and alter the s-th block in the AD and the t-th block in the plaintext at the same time, where $a - s = m - t$. Other blocks are fixed arbitrarily. So the i-th query (A_i, M_i) is of the form

$$\begin{cases} A_i = (A[1], \ldots, A[s-1], A_i[s], A[s+1], \ldots, A[a]), \\ M_i = (M[1], \ldots, M[t-1], M_i[t], M[t+1], \ldots, M[m]), \end{cases}$$

where $i = 1, \ldots, Q$, $A_i[s] = M_i[t] = \langle i \rangle$, $A[1], \ldots, A[s-1], A[s+1], \ldots, A[a], M[1], \ldots, M[t-1], M[t+1], \ldots, M[m] \in \{0,1\}^n$ are fixed arbitrarily, and we let $Q = 2^{n/2}$. We make Q

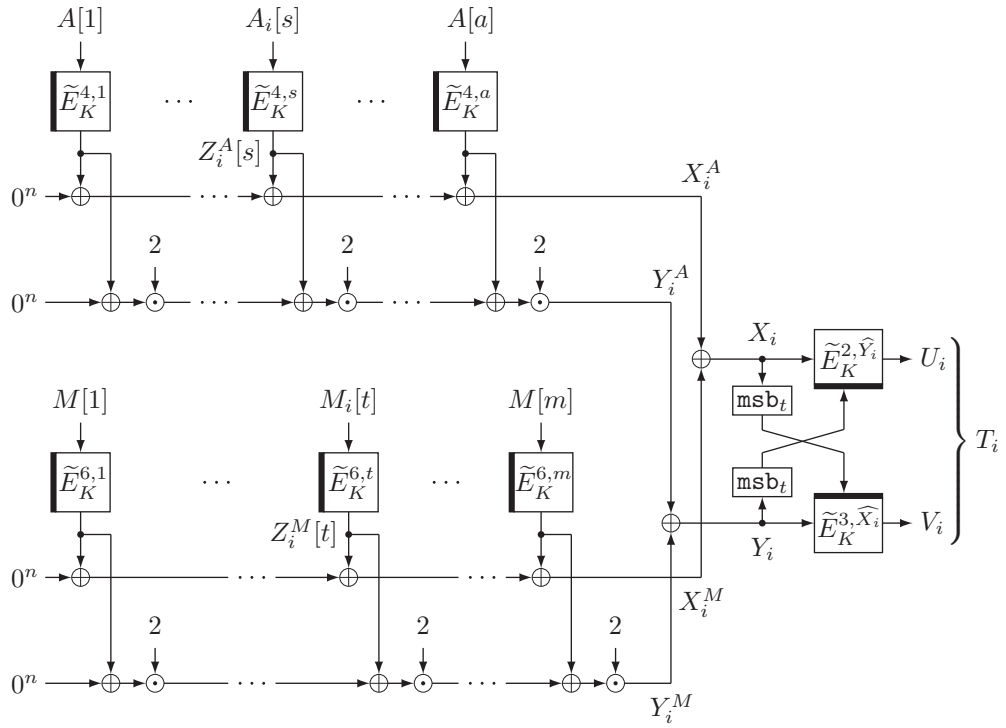

Figure 9: Our attack against SIVx. a and m can be different, and this figure assumes $|A[a]| = |M[m]| = n$. $A_i[s] = M_i[t] = \langle i \rangle$, and all other blocks are fixed. s and t are chosen to satisfy $a - s = m - t$.

queries and if we find a pair of queries that share the same value for T, then we decide that the oracle is SIVx.

Now for two queries (A_i, M_i) and (A_j, M_j), we need the event $X_i = X_j$, and this time we require that the XOR difference between the result of encrypting $A_i[s]$ and $A_j[s]$ to be the same as the XOR difference between the result of encrypting $M_i[t]$ and $M_j[t]$. See Fig. 9 illustrating the data flow for the i-th query (A_i, M_i).

To have $Y_i = Y_j$, we have to make sure that s and t are equal number of blocks before the end of AD and plaintext, respectively, which is maintained by the condition $a - s = m - t$. Then, our differences are multiplied by the same number and so they also cancel out.

More precisely, we have

$$\begin{aligned} X_i \oplus X_j &= Z_i^A[s] \oplus Z_j^A[s] \oplus Z_i^M[t] \oplus Z_j^M[t] \\ &= \widetilde{E}_K^{4,s}(\langle i \rangle) \oplus \widetilde{E}_K^{4,s}(\langle j \rangle) \oplus \widetilde{E}_K^{6,t}(\langle i \rangle) \oplus \widetilde{E}_K^{6,t}(\langle j \rangle)\,, \\ Y_i \oplus Y_j &= 2^{a-s+1} \cdot (Z_i^A[s] \oplus Z_j^A[s]) \oplus 2^{m-t+1} \cdot (Z_i^M[t] \oplus Z_j^M[t]) \\ &= 2^{a-s+1} \cdot (X_i \oplus X_j)\,, \end{aligned}$$

where we used the condition that $a - s = m - t$. With the same reasoning as in the case of $a = m$, we observe a collision in $\{T_1, \ldots, T_Q\}$, and the attack succeeds with the same probability as before.

These attacks are distinguishing attacks against the tagging function of SIVx that use $Q = 2^{n/2}$ encryption queries. Thus they break the privacy notion of SIVx.

5.3 Extension to Forgery Attack

The above distinguishing attack is easily extended to a forgery attack. Suppose that the adversary first performs the last attack of Sect. 5.2 to obtain (A_i, M_i) and (A_j, M_j) with colliding tags, T_i and T_j, $T_i = T_j$ for some $i, j \in \{1, \ldots, Q\}$. We know that A_i and A_j only differ in their s-th blocks, and M_i and M_j only differ in their t-th blocks. The adversary then queries (A', M') to the encryption oracle to obtain the response (C', T'), where A' is obtained by changing A_i except for the s-th block, and M' is obtained by changing M_i except for the t-th block, and keeping their lengths. Suppose that $A' = (A'_{\mathrm{pre}} \| A_i[s] \| A'_{\mathrm{post}})$ and $M' = (M'_{\mathrm{pre}} \| M_i[t] \| M'_{\mathrm{post}})$, where $|A'_{\mathrm{pre}}| = n(s-1)$ and $|A'_{\mathrm{post}}| = n(a-s)$ and $|M'_{\mathrm{pre}}| = n(t-1)$ and $|M'_{\mathrm{post}}| = n(t-m)$. Then, the adversary can compute the key stream, $S' = M' \oplus C'$. Finally the adversary queries (A'', C'', T'') to the decryption oracle, where

$$\begin{cases} A'' = (A'_{\mathrm{pre}} \| A_j[s] \| A'_{\mathrm{post}}) \,, \\ C'' = (M'_{\mathrm{pre}} \| M_j[t] \| M'_{\mathrm{post}}) \oplus S' \,, \\ T'' = T' \,. \end{cases}$$

The decryption oracle, $\mathsf{SIVx}^{-1}[\widetilde{E}_K]$, computes $M'' = C'' \oplus S' = (M'_{\mathrm{pre}} \| M_j[t] \| M'_{\mathrm{post}})$ as a decrypted plaintext, and then computes $\widehat{T}'' = \mathsf{vPMAC2x}[\widetilde{E}_K](A'', M'')$ of Fig. 6 to see if $\widehat{T}'' = T''$ holds, which is always true because the internal (X, Y) value (i.e. the sum of PHASHx outputs) will collide with that of query (A', M'). Hence the decryption oracle always accepts this query, i.e., the answer is not $\perp$, and the adversary wins.

Implication of the Attacks. The attacks in Sect. 5.2 and Sect. 5.3 use a collision with the birthday complexity as in the attacks in Sect. 3 and in Sect. 5.1, and we here point out that their implications are different. The attacks on PMACx and PMAC2x could be avoided if we appropriately modify the padding method, which also avoids the attack of Sect. 5.1 against vPMAC2x. However, the attacks in Sect. 5.2 and Sect. 5.3 indicate that the weakness of SIVx cannot possibly be removed by merely changing the padding method, and for this reason the design of SIVx is fundamentally flawed.

6 Discussions and Conclusions

In this paper, we showed that there are attacks against PMACx, PMAC2x, and SIVx with the query complexity of $O(2^{n/2})$.

We here discuss what went wrong with PMACx, PMAC2x, and SIVx. For PMACx and PMAC2x, the critical flaw is in the XCBC/CMAC-like treatment of last message blocks in PHASHx. This causes an output collision of PHASHx with probability $1/2^n$ for a pair of messages having specific forms in the last blocks. Unfortunately, the security proof given in [LN17] does not cover this event. The event should be captured in Subcase 2 of Case 1 in the proof of [LN17, Theorem 1].

For SIVx, as it is based on PHASHx, the same problem remains. Moreover, vPMAC2x in SIVx is different from PMAC2x as pointed out in Sect. 4, so that vPMAC2x can process A and M in parallel, and the difference enables a different type of attacks of birthday complexity. The paper [LN17] does not provide any specific analysis on this structure.

PMACx and PMAC2x are built on PMAC_TBC1k and PMAC_TBC3k designed and proposed by Naito [Nai15], and List and Nandi pointed out that the security proof of [Nai15] has issues, claiming a correct security proof. We note that the attacks presented in this paper indicate that the proofs of List and Nandi still have issues, yet the erroneous parts are different from the one that is related to the issue of [Nai15].

We remark that PMAC_TBC1k and PMAC_TBC3k are not affected by our attacks as they employ a different padding scheme. Finally, List and Nandi updated the specifications of PMACx, PMAC2x, and SIVx in their ePrint version [LN16] of [LN17] as a response to our attacks. Our attacks are not applicable to these updated specifications.

Acknowledgements.

The authors thank Thomas Peyrin for discussions, and Eik List and Mridul Nandi for feedback. The authors deeply thank the FSE 2018 reviewers for helpful comments that improved the paper. The work by Tetsu Iwata was supported in part by JSPS KAKENHI, Grant-in-Aid for Scientific Research (B), Grant Number 26280045, and was carried out while visiting Nanyang Technological University, Singapore.

References

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.

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

[BR00] John Black and Phillip Rogaway. CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 197–215. Springer, 2000.

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

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014.

[LN16] Eik List and Mridul Nandi. Revisiting Full-PRF-Secure PMAC and Using It for Beyond-Birthday Authenticated Encryption. *IACR Cryptology ePrint Archive*, 2016:1174, 2016.

[LN17] Eik List and Mridul Nandi. Revisiting Full-PRF-Secure PMAC and Using It for Beyond-Birthday Authenticated Encryption. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 258–274. Springer, 2017.

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. *J. Cryptology*, 24(3):588–613, 2011.

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

[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.

[NS90] Kazuo Nishimura and Masaaki Sibuya. Probability To Meet in the Middle. *J. Cryptology*, 2(1):13–22, 1990.

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

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

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