How to Recover the Full Plaintext of XCB [⋆]

Peng Wang[1], Shuping Mao[2(✉)], Ruozhou Xu[3], Jiwu Jing[1], and Yuewu Wang[1]

[1] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
{p-wang, jwjing, wangyuewu}@ucas.ac.cn
[2] Beijing Electronic Science & Technology Institute, Beijing, China
maoshuping19@mails.ucas.ac.cn
[3] State Grid Information & Telecommunication Branch, Beijing, China
xuruozhou21@mails.ucas.ac.cn

Abstract. XCB, a tweakable enciphering mode, is part of IEEE Std. 1619.2 for shared storage media. We show that all versions of XCB are not secure through three plaintext recovery attacks. A key observation is that XCB behaves like an LRW1-type tweakable block cipher for single-block messages, which lacks CCA security. The first attack targets one-block XCB, using three queries to recover the plaintext. The second one requires four queries to recover the plaintext that excludes one block. The last one requires seven queries to recover the *full* plaintext. The first attack applies to any scheme that follows the XCB structure, whereas the latter two attacks work on all versions of XCB, exploiting the separable property of the underlying universal hash function. To address these flaws, we propose the XCB* structure, an improved version of XCB that adds only two XOR operations. We prove that XCB* is STPRP-secure when using AXU hash functions, SPRPs, and a secure random-IV-based stream cipher.

Keywords: XCB · Tweakable enciphering mode · LRW1 · CCA.

1 Introduction

The term *tweakable enciphering mode* (TEM) [18], also referred to as *tweakable wide-block cipher mode* [15], is a length-preserving encryption scheme that provides strong tweakable pseudorandom permutation (STPRP) security. TEM is a permutation for a key and a public input called "tweak". In an ideal TEM, each tweak independently induces a random permutation. The security of a TEM is defined as its indistinguishability from the ideal one, even when an attacker can select tweaks and make encryption and decryption queries.

TEM is well-suited for applications like full disk encryption. By using the data unit index (e.g., sector number) as a tweak, it induces independent pseudorandom permutations to encrypt each data unit. In the last decade, significant efforts have been made in designing TEMs and proving their security. Notable constructions based on block ciphers include CMC [18], EME [19], EME* [16],

[⋆] We note that the work of Bhati et. al. [3] predated and inspired this work.

XCB [25,26], HCTR [35], PEP [6], TET [17], HEH [33], HCH [7], HCI [30], MXCB [30], TCT$_1$ [34], TCT$_2$ [34], FMix [4], Adiantum [9], and HCTR2 [10], etc. NIST [8] has recently announced plans to develop and standardize a tweakable enciphering mode called the "accordion mode", highlighting its ability to accommodate varying input lengths. This initiative has sparked several designs including ddd-AES [12], HCTR+ [11], DbHCTR [23], etc.

The design of TEM often utilizes block ciphers and encounters two primary challenges: incorporating the tweak into the permutation and extending the fixed block length of the block cipher to support variable input lengths.

For the first challenge, if we limit the message length to one block, TEM becomes the foundational concept of a tweakable block cipher (TBC) which was proposed by Liskov et al. in their seminal work [24]. This allows us to focus solely on methods for incorporating tweaks. Common TBC constructions based on block ciphers include LRW1, LRW2 [24], CLRW2 [22], and TNT [2], etc. Among these constructions, it is notable that LRW1 is the only one secure against chosen plaintext attacks (CPA) but not against chosen ciphertext attacks (CCA).

For the second challenge, TEMs typically utilize three frameworks to build a variable-input-length permutation [7]: Encrypt-Mix-Encrypt, Hash-CTR-Hash, and Hash-ECB-Hash. Here, "Hash" refers to a *universal hash function*, which is a component more efficient than a block cipher in the construction of modes.

XCB was initially proposed by McGrew et al. [25] in 2004 without a security proof. We refer to it as XCBv1. In 2007, they proposed another version [26], referred to as XCBv2, with a security proof of STPRP. Subsequently, in 2010, XCBv2 was included into the IEEE 1619.2 standard [20].

In 2013, Chakraborty et al. [5] showed that XCBv2 is not secure when the message length is not a multiple of block length, using a distinguishing attack. They also gave security proofs for XCBv1 and the multiple-block-length XCBv2, referred to as XCBv2fb.

Recently, Bhati et al. [3] proposed an attack against XCBv2fb that can recover the plaintext, except for one block data, using only two queries. So it is a *partial* plaintext recovery attack.

In summary, even though the security of XCBv2 [26] and XCBv2fb [5] has been compromised, the current attacks do not pose a threat to XCBv1. Therefore, according to existing literature, XCBv1 [25] is still considered secure. Additionally, there are no *full* plaintext recovery attacks on any version of XCB.

XCB consists of two algorithms: encryption $\mathbf{E}$ and decryption $\mathbf{D}$. Specifically, $\mathbf{E}$ takes a key K, a tweak T and a plaintext P as input and outputs a ciphertext $C = \mathbf{E}_K^T(P)$. $\mathbf{D}_K^T$ is the inverse of $\mathbf{E}_K^T$. For simplicity, we will omit the key in the notation for both encryption and decryption in the following discussion.

A Warm-up full plaintext recovery attack. We first present a *full* plaintext recovery attack on XCBv2bf following the method in [3] by making an additional decryption query. The attack can retrieve the full plaintext. However, the attack still does not work on XCBv1.

Our main contribution is the proposal of three plaintext recovery attacks applicable to all versions of XCB. These attacks follow a *uniform* pattern of alternate querying $\mathbf{D}^{T'}$ and $\mathbf{E}^{T}$, with variations in the number of queries and the length of the message.

The key observation is that XCB, when processing a single-block message, behaves similarly to an LRW1-type tweakable block cipher, lacking CCA security.

Attack 1. When the message length is one block, similar to the distinguishing attack on LRW1 described in the paper presentation of [21], we have the equation:

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T} = I,$$

where I is an identity transformation. Of course, we can use this equation to distinguish XCB from the corresponding ideal TEM. We reformulate it as $\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}$. Hence, given (T, C), the corresponding plaintext can be retrieved in the following manner: first, use C to query $\mathbf{D}^{T'}$; then, query $\mathbf{E}^{T}$ using the obtained result; and finally, query $\mathbf{D}^{T'}$ again. This process results in obtaining $\mathbf{D}^{T}(C)$.

Attack 1 is independent of the specific properties of the components in XCB, and therefore, it can be applied to any scheme that follows the XCB structure, including XCBv1 [25], XCBv2 [26] and XCBv2fb [5].

Attack 2. For a longer ciphertext, such as a typical sector length of 512 bytes, how can we recover the plaintext? By making an additional query to $\mathbf{E}^{T}$, we can recover the plaintext except for one block.

The attack exploits the separable property of the universal hash function (UHF) in XCB. In simple terms, any string XORed with the input of the UHF h can be separated from the function, as demonstrated by the equation:

$$h(X \oplus \varDelta_1, Y \oplus \varDelta_2) = h(X, Y) \oplus g(\varDelta_1, \varDelta_2),$$

where g is a keyed function that is independent of both X and Y.

Attack 3. Surprisingly, by executing three additional queries, we can recover the entire plaintext of XCB. The reason is the following equation,

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}.$$

Attack 2 and 3 can be applied to any scheme that utilizes the XCB structure with separable UHFs, including XCBv1 [25], XCBv2 [26], XCBv2fb [5], HCI and MXCB [30].

We summarize all attacks in Table 1. In particular, regarding the standard version of XCB [20], we present the first *full* plaintext attack. Please note that the three attacks are quite *general* in nature; they only require knowledge of the XCB structure (Attack 1) or an additional separable property of UHFs (Attacks 2 and 3), without the necessity for further algorithmic details. In the following, when we mention XCB, it means all versions of XCB.

Modification. We modify the XCB structure into XCB* with only extra two XOR operations. We prove that, when h_1 and h_2 are almost-XOR-universal (AXU) hash functions, e and d are SPRPs, and c is a secure random-IV-based stream cipher, the XCB* structure is an STPRP.

Table 1. Summary of Attacks on XCB. We note that Attack 1 is applicable to all XCB-type schemes and Attack 2 & 3 are applicable to all XCB-type schemes with separable UHFs.

	Message length	Number of queries	Recovered bits	Target schemes	Ref.
Attack in [5]	$m > n$	2	N/A	XCBv2 [26]	[5]
Attack in [3]	$m > n$	2	$m - n$	XCBv2 [26], XCBv2fb [5]	[3]
A warm-up attack	$m > n$	3	m	XCBv2 [26], XCBv2fb [5]	4.2
Attack 1	$m = n$	3	n	XCBv1 [25], XCBv2 [26], XCBv2fb [5]	5.3
Attack 2	$m > n$	4	$m - n$	XCBv1 [25], XCBv2 [26], XCBv2fb [5], HCI [30], MXCB [30]	5.4
Attack 3	$m > n$	7	m		5.5

In the following, we give notations and definitions in Section 2, the XCB and its components in Section 3, previous attacks and a warm-up attack in Section 4, three plaintext recovery attacks in Section 5, modification in Section 6, discussions in Section 7 and conclusions in Section 8.

2 Preliminaries

Notations. Let $\mathcal{X}$ be a set. Let $x \overset{\$}{\leftarrow} \mathcal{X}$ denote selecting an element x from the set $\mathcal{X}$ uniformly at random. Let $\mathsf{Perm}(\mathcal{X})$ be a set of all length-preserving permutations on $\mathcal{X}$. If $\mathcal{X} = \{0,1\}^n$, we denote $\mathsf{Perm}(\mathcal{X})$ as $\mathsf{Perm}(n)$. Let $\pi \overset{\$}{\leftarrow} \mathsf{Perm}(\mathcal{X})$ be a random permutation on $\mathcal{X}$. Let $\mathsf{TPerm}(\mathcal{T}, \mathcal{X})$ be a set of all length-preserving tweakable permutations on $\mathcal{X}$ with tweak space of $\mathcal{T}$. Let $\tilde{\pi} \overset{\$}{\leftarrow} \mathsf{TPerm}(\mathcal{T}, \mathcal{X})$ be a random tweakable permutation, so that $\tilde{\pi}(T, \cdot)$ is a random permutation for each $T \in \mathcal{T}$. Let $\mathcal{A}$ be an adversary. Let $\mathcal{A}^{f(\cdot)} \Rightarrow b$ represent an algorithm that performs queries on the oracle f and outputs the bit b.

For a binary string X, $|X|$ denotes the length of X in bits. Let $\mathrm{msb}_r(X)$ and $\mathrm{lsb}_r(X)$ be the r leftmost and the r rightmost bits of X respectively. For $X, Y \in \{0,1\}^n$, $X \oplus Y$ and XY respectively denote addition and multiplication in $GF(2^n)$. Let $\mathrm{bin}_s(i)$ be the s-bit binary representation of i. $X[a, b]$ denotes the substring of X from the ath bit through the bth, and indexing starts at 0.

Block cipher. A block cipher (or BC for short) $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a function with key space $\mathcal{K}$ and message space $\{0,1\}^n$ such that for every key $K \in \mathcal{K}$, $E(K, \cdot)$ is a permutation on $\{0,1\}^n$. The inverse E is denoted by D such that $D(K, \cdot)$ is the inverse of $E(K, \cdot)$. We write $E(K, P)$ $(D(K, C))$ as $E_K(P)$ $(D_K(C))$ and sometimes omit K for convenience.

Tweakable enciphering mode. A tweakable enciphering mode (or TEM for short) $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ is a function with key space $\mathcal{K}$, tweak space $\mathcal{T}$, and message space $\mathcal{X}$ such that for every key $K \in \mathcal{K}$ and every tweak $T \in \mathcal{T}$,

$\mathbf{E}(K, T, \cdot)$ is a length-preserving permutation on $\mathcal{X}$. The inverse of $\mathbf{E}$ is denoted by $\mathbf{D}$ such that $\mathbf{D}(K, T, \cdot)$ is the inverse of $\mathbf{E}(K, T, \cdot)$. We write $\mathbf{E}(K, T, P)$ ($\mathbf{D}(K, T, C)$) as $\mathbf{E}_K^T(P)$ ($\mathbf{D}_K^T(C)$) and sometimes omit K for convenience. If $\mathcal{X}$ is $\{0, 1\}^n$, then the TEM is referred to as a tweakable block cipher (TBC). Additionally, if $\mathcal{T}$ is empty, the TEM becomes a block cipher (BC).

Strong tweakable pseudorandom permutation (STPRP). Here, we consider the adversary $\mathcal{A}$ that can query both the encryption and the decryption oracles. Let $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ be a tweakable enciphering mode. Let $\tilde{\pi} \xleftarrow{\$} \mathsf{TPerm}(\mathcal{T}, \mathcal{X})$. The STPRP advantage of $\mathcal{A}$ is defined as:

$$\mathbf{Adv}_{\mathbf{E}}^{\text{stprp}}(\mathcal{A}) = \Pr_{K \xleftarrow{\$} \mathcal{K}} \left[\mathcal{A}^{\mathbf{E}_K, \mathbf{D}_K} \Rightarrow 1\right] - \Pr_{\tilde{\pi} \xleftarrow{\$} \mathsf{TPerm}(\mathcal{T}, \mathcal{X})} \left[\mathcal{A}^{\tilde{\pi}, \tilde{\pi}^{-1}} \Rightarrow 1\right].$$

If $\mathcal{T} = \emptyset$, we write the notion of STPRP as SPRP.

Universal hash function. $h : \mathcal{K} \times \mathcal{X} \times \mathcal{Y} \to \{0, 1\}^n$ is ϵ-almost-XOR-universal (ϵ-AXU) if for any given two distinct inputs $(X, Y), (X', Y') \in \mathcal{X} \times \mathcal{Y}$ and any output $Z \in \{0, 1\}^n$,

$$\Pr_{K \xleftarrow{\$} \mathcal{K}} [h_K(X, Y) \oplus h_K(X', Y') = Z] \leq \epsilon.$$

When Z is fixed to 0^n, h is called ϵ-almost universal (ϵ-AU).

Separable UHF. The universal hash function h is separable if there exists a keyed function g, such that for any X, Y, Δ_1, Δ_2 ($|X| = |\Delta_1|$, $|Y| = |\Delta_2|$), the following equation holds,

$$h_K(X \oplus \Delta_1, Y \oplus \Delta_2) = h_K(X, Y) \oplus g_K(\Delta_1, \Delta_2).$$

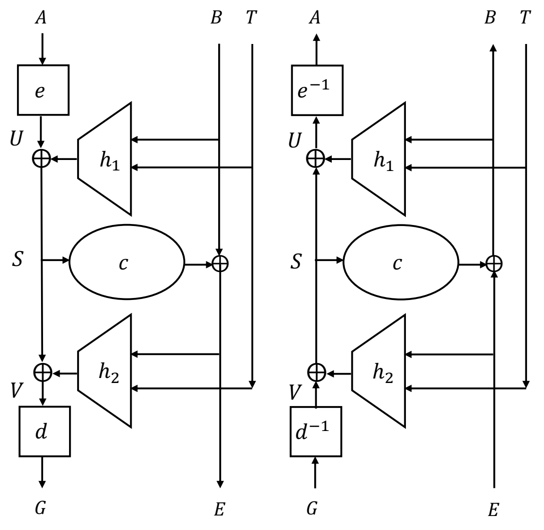
3 XCB and Its Components

3.1 The XCB structure

XCB has two main versions: XCBv1 [25] and XCBv2 [26] (XCBv2fb [5] is the multiple-block-length XCBv2). We provide detailed descriptions of XCBv1 and XCBv2 in Appendix A. To make our attacks as general as possible, we abstract the underlying structure of all versions of XCB, as shown in Figure 1.

The encryption $\mathbf{E}$ and decryption $\mathbf{D}$ in the XCB structure use the following components: e, d, c, h_1 and h_2. e and d are keyed permutations on $\{0, 1\}^n$, and instantiated by the encryption and decryption of a block cipher respectively. Their inverses are e^{-1} and d^{-1} respectively. c is a stream cipher, instantiated by the CTR mode. h_1 and h_2 are two universal hash functions. Note that they all contain keys. For simplicity, we omit the keys in the text and figures. The message of XCB is written as $A \| B$, where A is an n-bit string fitting for the block length of the block cipher and B is the rest of the message.

The main differences between the two versions of XCB are as follows. See Appendix A for more details.



```
Algorithm $\mathbf{E}^T(A||B)$
  $U \leftarrow e(A)$
  $S \leftarrow U \oplus h_1(B,T)$
  $E \leftarrow B \oplus c(S)$
  $V \leftarrow S \oplus h_2(E,T)$
  $G \leftarrow d(V)$
  return $G||E$
```

```
Algorithm $\mathbf{D}^T(G||E)$
  $V \leftarrow d^{-1}(G)$
  $S \leftarrow V \oplus h_2(E,T)$
  $B \leftarrow E \oplus c(S)$
  $U \leftarrow S \oplus h_1(B,T)$
  $A \leftarrow e^{-1}(U)$
  return $A||B$
```

Fig. 1. The XCB structure

- Data sequential arrangement: In XCBv1, following the generic XCB structure, the plaintext is denoted as $A\|B$, with its corresponding ciphertext as $G\|E$. In contrast, XCBv2 rearranges the plaintext to $B\|A$, resulting in the ciphertext being reordered as $E\|G$. The sequential arrangement of input and output data does not impact security. For simplicity, we will use the XCBv1 arrangement in the subsequent sections.
- Key size: In XCBv1, the key length is the same as the block length, which means XCBv1 can use AES-128. In contrast, XCBv2's key length matches the key length of the block cipher, allowing it to use all versions of AES.
- Keys for UHFs: The keys for the components are generated from the main key. In XCBv1, each of the two UHFs employs a different key, while in XCBv2, both functions share the same key.

3.2 Separable UHFs

We first define a common function used by UHFs in XCBv1 and XCBv2.

$$h_H(X,Y) = X_1 H^{u+v+1} \oplus X_2 H^{u+v} \oplus \cdots \oplus X_{u-1} H^{v+3} \oplus X_u H^{v+2}$$

$$\oplus Y_1H^{v+1} \oplus Y_2H^{v} \oplus \cdots \oplus Y_{v-1}H^{3} \oplus Y_vH^2$$
$$\oplus (\mathrm{bin}_{\frac{n}{2}}(|X|) \parallel \mathrm{bin}_{\frac{n}{2}}(|Y|))H \tag{1}$$

where $X_1\|X_2\|\cdots\|X_u = \mathrm{pad}(X)$, $Y_1\|Y_2\|\cdots\|Y_v = \mathrm{pad}(Y)$, $|X_i| = n$, $|Y_j| = n$, $i = 1, 2, \cdots, u$ and $j = 1, 2, \cdots, v$. If $|X|$ is a multiple of n, then $\mathrm{pad}(X) = X$. Otherwise, $\mathrm{pad}(X) = X\|0^r$, where r is the smallest number required to make $|(X\|0^r)|$ a multiple of n.

In XCBv1, h_1 and h_2 are defined as:

$$h_i(X, T) = h_{H_i}(X, T), i = 1, 2,$$

where H_1 and H_2 are two distinct keys for UHFs.

In XCBv2, h_1 and h_2 are defined as:

$$h_1(X, T) = h_H(0^n\|T, \mathrm{pad}(X)\|0^n),$$
$$h_2(X, T) = h_H(T\|0^n, \mathrm{pad}(X)\|\mathrm{bin}_{\frac{n}{2}}(|(T\|0^n)|) \parallel \mathrm{bin}_{\frac{n}{2}}(|X|)),$$

where H is the same key shared by UHFs.

Then we have the following lemma.

Lemma 1. *The universal hash functions h_1, h_2 in XCBv1 and XCBv2 are separable.*

Proof. Note that Theorem 2 in [26] mentions that the universal hash functions in XCBv2 are "linear", and this property naturally ensures their separability. We first prove the separability of h. From (1) we have

$$\begin{aligned}
&h(X \oplus \Delta_1, Y \oplus \Delta_2)\\
=&(X_1 \oplus \Delta_{11})H^{u+v+1} \oplus \cdots \oplus (X_u \oplus \Delta_{1u})H^{v+2} \oplus (Y_1 \oplus \Delta_{21})H^{v+1} \oplus \cdots\\
&\oplus (Y_v \oplus \Delta_{2v})H^2 \oplus (\mathrm{bin}_{\frac{n}{2}}(|X \oplus \Delta_1|) \parallel \mathrm{bin}_{\frac{n}{2}}(|Y \oplus \Delta_2|))H\\
=&X_1H^{u+v+1} \oplus \cdots \oplus X_uH^{v+2} \oplus Y_1H^{v+1} \oplus \cdots \oplus Y_vH^2\\
&\oplus (\mathrm{bin}_{\frac{n}{2}}(|X|) \parallel \mathrm{bin}_{\frac{n}{2}}(|Y|))H\\
&\oplus \Delta_{11}H^{u+v+1} \oplus \cdots \oplus \Delta_{1u}H^{v+2} \oplus \Delta_{21}H^{v+1} \oplus \cdots \oplus \Delta_{2v}H^2\\
=&h(X, Y) \oplus g(\Delta_1, \Delta_2),
\end{aligned}$$

where $X_1\|...\|X_u = \mathrm{pad}(X)$, $\Delta_{11}\|...\|\Delta_{1u} = \mathrm{pad}(\Delta_1)$, $Y_1\|...\|Y_v = \mathrm{pad}(Y)$, $\Delta_{21}\|...\|\Delta_{2v} = \mathrm{pad}(\Delta_2)$ and

$$g(\Delta_1, \Delta_2) = \Delta_{11}H^{u+v+1} \oplus \cdots \oplus \Delta_{1u}H^{v+2} \oplus \Delta_{21}H^{v+1} \oplus \cdots \oplus \Delta_{2v}H^2.$$

In XCBv1, h_1, h_2 use h directly with two keys, so they are separable. In XCBv2, it is easy to verify that $\mathrm{pad}(X \oplus \Delta) = \mathrm{pad}(X) \oplus \mathrm{pad}(\Delta)$ for any X and Δ satisfying $|X| = |\Delta|$, indicating that they are also separable. □

In the following text, if h_1 and h_2 are separable, we denote

$$h_1(X \oplus \Delta_1, T \oplus \Delta_2) = h_1(X, T) \oplus g_1(\Delta_1, \Delta_2),$$
$$h_2(X \oplus \Delta_1, T \oplus \Delta_2) = h_2(X, T) \oplus g_2(\Delta_1, \Delta_2).$$

We note that all versions of XCB, including XCBv1, XCBv2 and XCBv2fb follow the XCB structure in Figure 1 and their UHFs are all separable.

4 Previous Attacks on XCB and A New Attack

4.1 Previous attacks

Recall that, in XCBv2, h_1 and h_2 are defined as:

$$h_1(X,T) = h_H(0^n \| T, \mathrm{pad}(X) \| 0^n),$$

$$h_2(X,T) = h_H(T \| 0^n, \mathrm{pad}(X) \| \mathrm{bin}_{\frac{n}{2}}(|(T\|0^n)|) \,\|\, \mathrm{bin}_{\frac{n}{2}}(|X|),$$

where H is the same key shared by UHFs. $|X|$ is a multiple of n, so $\mathrm{pad}(X) = X$. h_1 and h_2 are separable and use the same key, therefore there exist a common function g such that $h_1(X \oplus \varDelta, T) = h_1(X\oplus, T) \oplus g(\varDelta, 0)$ and $h_2(X \oplus \varDelta, T) = h_2(X\oplus, T) \oplus g(\varDelta, 0)$.

Chakraborty et al. [5] observed that the padding function of pad is not injective. For example, for $B = 0^{3n}$ and $B' = 0^{3n-1}$, $\mathrm{pad}(B) = \mathrm{pad}(B') = 0^{3n}$. Therefore, in XCBv2, h_1 is not an almost universal hash function, due to the equation $h_1(B,T) = h_1(B',T)$. If we query the encryption oracle with $(T, A\|B)$ and $(T, A\|B')$, we get $G\|E$ and $G'\|E'$ respectively. The component c receives the same input, resulting in identical outputs. Therefore, $\mathrm{msb}_n(B \oplus E) = \mathrm{msb}_n(c(e(A) \oplus h_1(B,T))) = \mathrm{msb}_n(c(e(A) \oplus h_1(B',T))) = \mathrm{msb}_n(B' \oplus E')$. So, we can use the equation of $\mathrm{msb}_n(B \oplus E) = \mathrm{msb}_n(B' \oplus E')$ to distinguish XCBv2 from a tweakable random permutation using two encryption queries. To prevent the attack, they restricted the message to multiples of the block length and called it XCBv2fb. They also provided security proofs for both XCBv1 and XCBv2fb.

Recently, Bhati et al. [3] gave a *partial* plaintext recovery attack against XCBv2fb with two queries. Given $(T, G\|E)$ (the corresponding plaintext is $A\|B$), the attacker perform the following steps:

- Query the decryption oracle $\mathbf{D}$ with $(T, G\|(E \oplus \varDelta))$ and get $A_1\|B_1$.
- Query the encryption oracle $\mathbf{E}$ with $(T, A_1\|(B_1 \oplus \varDelta))$ and get $G_2\|E_2$.
- Output $B = E \oplus B_1 \oplus E_2 \oplus \varDelta$.

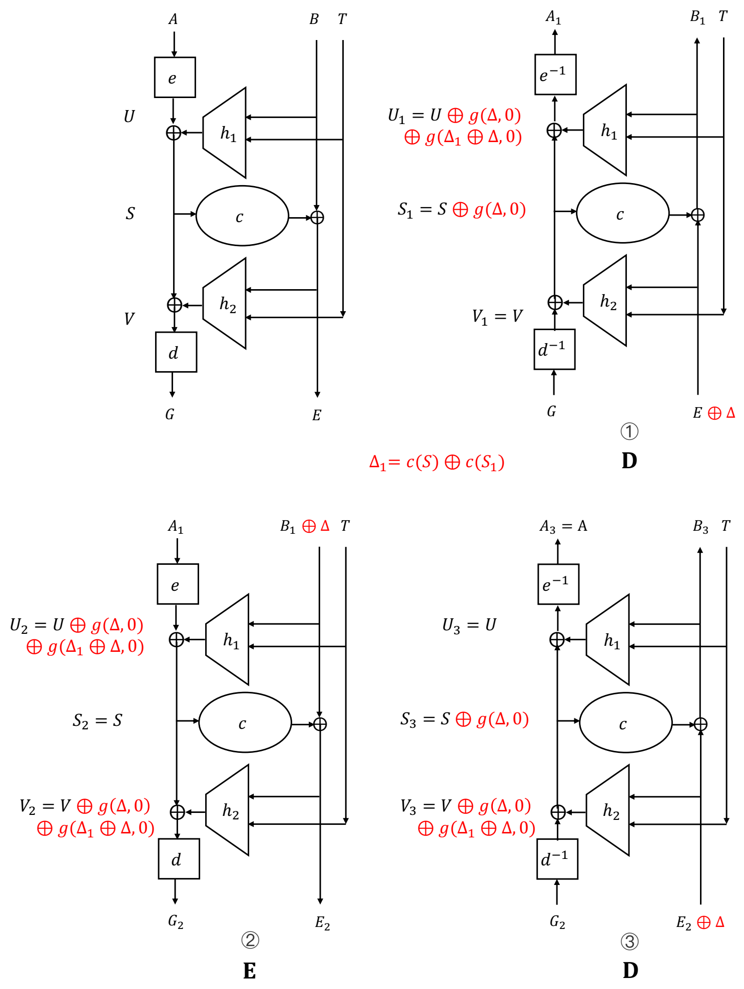
Due to the separable property of UHFs and the fact that h_1 and h_2 share the same key, the input differential to the component c induced during the decryption query, canceled out by the differential induced in the encryption query. Hence, $B \oplus E = (B_1 \oplus \varDelta) \oplus E_2$, allowing the recovery of the *partial* plaintext B.

4.2 A warm-up full plaintext recovery attack on XCBv2fb

If the attacker continues to query the decryption oracle with $(T, G_2\|(E_2 \oplus \varDelta))$ and obtain $A_3\|B_3$, then $A = A_3$. Therefore we get a *full* plaintext recovery attack against XCBv2fb with three queries.

We demonstrate the correctness of the attacks through Figure 2. Additionally, we verify the attacks using the standard version of XCBv2 in Appendix B.

In the process of encrypting or decrypting with XCB, as shown in Figure 1, we define three intermediate values: U, S, and V. More specifically, for $\mathbf{E}^T(A\|B) = G\|E$ or $\mathbf{D}^T(G\|E) = A\|B$, we have



A
B
T
e
U
h_1
S
c
V
h_2
d
G
E
A_1
B_1
T
e^{-1}
$U_1 = U \oplus g(\Delta, 0) \oplus g(\Delta_1 \oplus \Delta, 0)$
$S_1 = S \oplus g(\Delta, 0)$
$V_1 = V$
d^{-1}
G
$E \oplus \Delta$
①
D
$\Delta_1 = c(S) \oplus c(S_1)$
A_1
$B_1 \oplus \Delta$
$U_2 = U \oplus g(\Delta, 0) \oplus g(\Delta_1 \oplus \Delta, 0)$
$S_2 = S$
$V_2 = V \oplus g(\Delta, 0) \oplus g(\Delta_1 \oplus \Delta, 0)$
G_2
E_2
②
E
$A_3 = A$
B_3
$U_3 = U$
$S_3 = S \oplus g(\Delta, 0)$
$V_3 = V \oplus g(\Delta, 0) \oplus g(\Delta_1 \oplus \Delta, 0)$
G_2
$E_2 \oplus \Delta$
③
D

Fig. 2. The process of full plaintext recovery attack against XCBv2fb

- $U = e(A)$,
- $V = d^{-1}(G)$,
- $S = U \oplus h_1(B, T) = V \oplus h_2(E, T)$.

For the i-th query, the corresponding three values are denoted as U_i, S_i and V_i, $i = 1, 2, 3$.

The intermediate values in each step were illustrated in Figure 2. We see that $A = A_3$ and $B = E \oplus B_1 \oplus E_2 \oplus \Delta$. We also provide experimental validation in Appendix B.

The above attacks all target the standard version XCBv2 or its restricted version XCBv2fb. In XCBv1, h_1 and h_2 use different keys, therefore they do not share a common function g under separability property. After two queries, the intermediate value S_3 no longer go back to S, making the attack in [3] and our warm-up attack ineffective.

In the following, we propose a set of more *general* plaintext recovery attacks that can be applied to all versions of XCB. Consequently, none of the XCB versions, including XCBv1, are secure.

5 Recover the (Full) Plaintext of XCB

5.1 XCB is an LRW1-type TEM

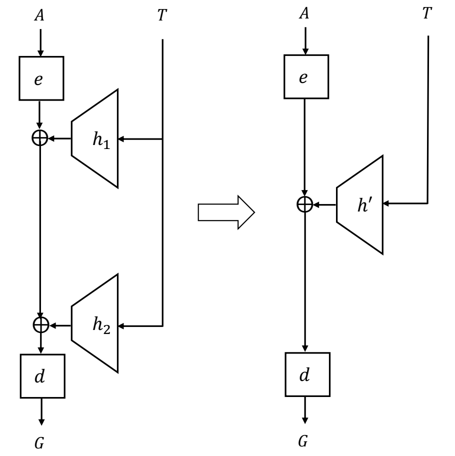


Fig. 3. The encryption **E** of XCB when $m = n$

A key observation is that when we minimize the length of the message to one block (at this point $m = n$ and the input B is an empty string), XCB becomes an LRW1-type construction. Figure 3 gives the encryption of XCB in this case. In this scenario, the hash value of tweak T is XORed into the state between two keyed permutations. Specifically,

$$\mathbf{E}^T(A) = d(e(A) \oplus h'(T)),$$

where $h'(T) = h_1(T) \oplus h_2(T)$.

The encryption of LRW1 is

$$\mathbf{E}^T(P) = e(e(P) \oplus T),$$

where e is a block cipher. LRW1 is a CPA secure tweakable block cipher when e is a PRP, and there exists a 4-query CCA attack to distinguish it from a tweakable random permutation as shown in the paper presentation of [21][4], in which one slide illustrated that $\mathbf{D}^{T_4}\mathbf{E}^{T_3}\mathbf{D}^{T_2}\mathbf{E}^{T_1} = I$ for tweaks $T_i, i = 1, 2, 3, 4$ satisfying $\bigoplus_{i=1}^{4} T_i = 0$. So we can use this property to distinguish LRW1 from a tweakable random permutation.

In XCB ($m = n$), the difference is that the tweak is processed by a UHF. We just choose tweaks satisfying $T_1 = T_3 = T$ and $T_2 = T_4 = T'$, so that

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T} = I$$

for XCB ($m = n$). We can use this property to perform a distinguishing attack with 4 queries. But if we turn the equation into

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T},$$

for any tweak-ciphertext pair (T, G), the queries to $\mathbf{D}^{T'}$, $\mathbf{E}^{T}$ and $\mathbf{D}^{T'}$ in succession is equivalent to direct decryption of $\mathbf{D}^{T}$. Therefore, we only use 3 queries to recover the full plaintext. This is Attack 1.

We further extend the attack to the case with arbitrary message lengths, resulting in Attack 2 and Attack 3.

5.2 Overview of three attacks

Given a tweak-ciphertext pair $(T, G\|E)$ of XCB, how can one retrieve the corresponding plaintext $A\|B$ without direct decryption? We call it a full recovery plaintext attack when all $A\|B$ can be recovered, and we call it a partial recovery plaintext attack when only A or B can be recovered. In the following, we assume that the message length is m bits and the block length of the block cipher is n bits.

Figure 4 depicts three attacks that share a *uniform* pattern of alternating queries between the decryption and encryption oracles with two different tweaks. Given $(T, G\|E)$, the attacker first queries the decryption oracle with $(T', G\|E)$ and gets $(A_1\|B_1) = \mathbf{D}^{T'}(G\|E)$, where $T' = T \oplus \Delta$ is a different tweak from T, and $\Delta \neq 0$. The attacker then queries the encryption oracle with $(T, A_1\|B_1)$, and gets $(G_2\|E_2) = \mathbf{E}^{T}(A_1\|B_1)$. So the attacker queries $\mathbf{D}^{T'}$ and $\mathbf{E}^{T}$ alternatively. We assume that the attacker gets the following answers: $A_3\|B_3$, $G_4\|E_4$, $A_5\|B_5$, $G_6\|E_6$ and $A_7\|B_7$, as illustrated in Figure 4.

The attacks differ in the number of queries and the length of the message.

– Attack 1 needs first three queries and the message length $m = n$. The final output A_3 is the full plaintext A.

[4] Interestingly, despite assertions in numerous papers that LRW1 lacks CCA security, no concrete attack can be found in the literature. In 2012, during discussions about the design of McOE [13], we presented an attack on LRW1 via email to the authors. As a result, they removed one of McOE instantiations, namely McOE-D, from the paper, as it is based on LWR1. While writing the paper, we only identified an attack in the presentation of [21]. It may have been considered too simple to be worth mentioning.

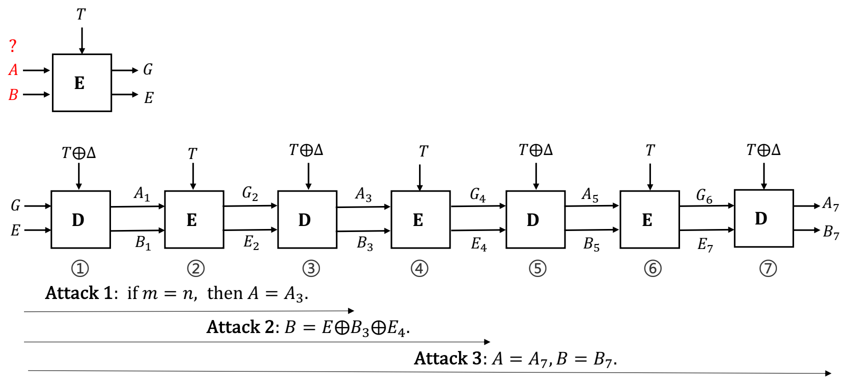


Fig. 4. Three plaintext recovery attacks

– Attack 2 needs first four queries and the message length $m > n$. The final output $E \oplus B_3 \oplus E_4$ is the partial plaintext B.
– Attack 3 needs all seven queries and the message length $m > n$. The resulting output, $A_7 \| B_7$, is the full plaintext $A \| B$.

5.3 Attack 1: full plaintext recovery attack with 3 queries ($m = n$)

Attack 1 performs the first three queries with a message length of $m = n$, and the output is A_3.

The correctness of Attack 1 comes from the following proposition.

Proposition 1. *In XCB, if the message length $m = n$, then for any different T and T',*

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}.$$

Proof. For $\mathbf{E}^T(A) = G$ or $\mathbf{D}^T(G) = A$, the intermediate values U and V are shown in Figure 5. The left side of the equation in Proposition 1 corresponds to three queries in Attack 1 and the right side corresponds to the direct decryption. For the i-th query, the intermediate values are U_i and V_i. ·

For distinct T and T', the intermediate values before and during the queries are

$$\begin{aligned}
&U = e(A),\\
&V = U \oplus h'(T) = V_1,\\
&U_1 = U \oplus h'(T) \oplus h'(T') = U_2,\\
&V_2 = U \oplus h'(T') = V_3,\\
&U_3 = U.
\end{aligned}$$

So $A_3 = e^{-1}(U) = e^{-1}(e(A)) = A$. That is, $\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}(G) = A = \mathbf{D}^{T}(G)$, so we have $\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}$. □

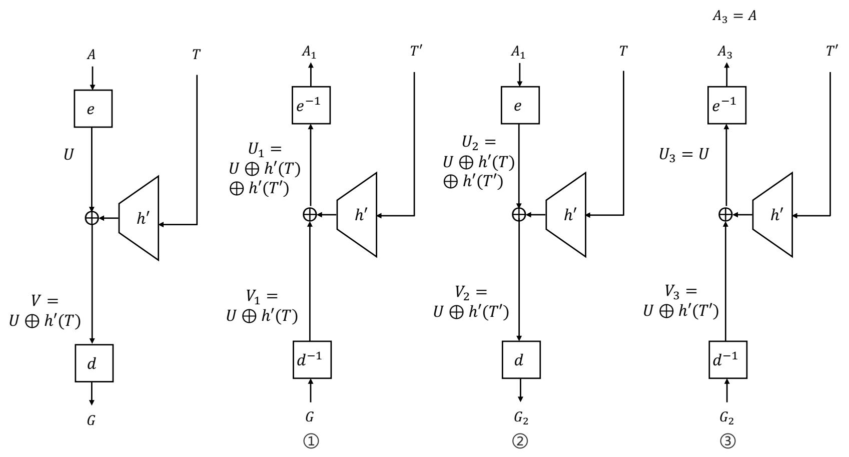


Fig. 5. The process of $\mathbf{D}^{T'}, \mathbf{E}^{T}, \mathbf{D}^{T'}$ (Query 1 to Query 3)

In the descriptions of XCBv1 [25] and XCBv2 [26], the message length is within the range $[n, 2^{39}]$, where n is the block length. In the standard version [20] that employs AES as the underlying block cipher, the message length is the message length is defined as m bits, where m ranges from 128 to 2^{32} and is a multiple of 8. While Attack 1 primarily focuses on an extreme case scenario, it is applicable to all versions of XCB. Moreover, Attack 1 does not rely on any specific properties of the underlying components, it applies to any scheme that follows the XCB structure.

To exploit the method to add the tweak, we restrict the message length as a block in other TEMs. We can categorize the methods into four types: LRW1, LRW2, CLRW2 and TNT in Appendix C. We find that HCI and MXCB [30] are also LRW1-type constructions. Since the minimal message length is set to be two blocks, Attack 1 cannot be applied to HCI and MXCB. However, the following attacks are effective.

5.4 Attack 2: partial plaintext recovery attack with 4 queries

Attack 2 executes the first four queries without message length restrictions. The output is $E \oplus B_3 \oplus E_4$.

The correctness of Attack 2 comes from the following proposition.

Proposition 2. *In XCB, for any different T and T', if h_1, h_2 are separable and*

$$\mathbf{E}^{T}(A \| B) = G \| E,$$
$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}(G \| E) = A_3 \| B_3,$$
$$\mathbf{E}^{T}(A_3 \| B_3) = G_4 \| E_4.$$

Then we have

$$B = E \oplus B_3 \oplus E_4.$$

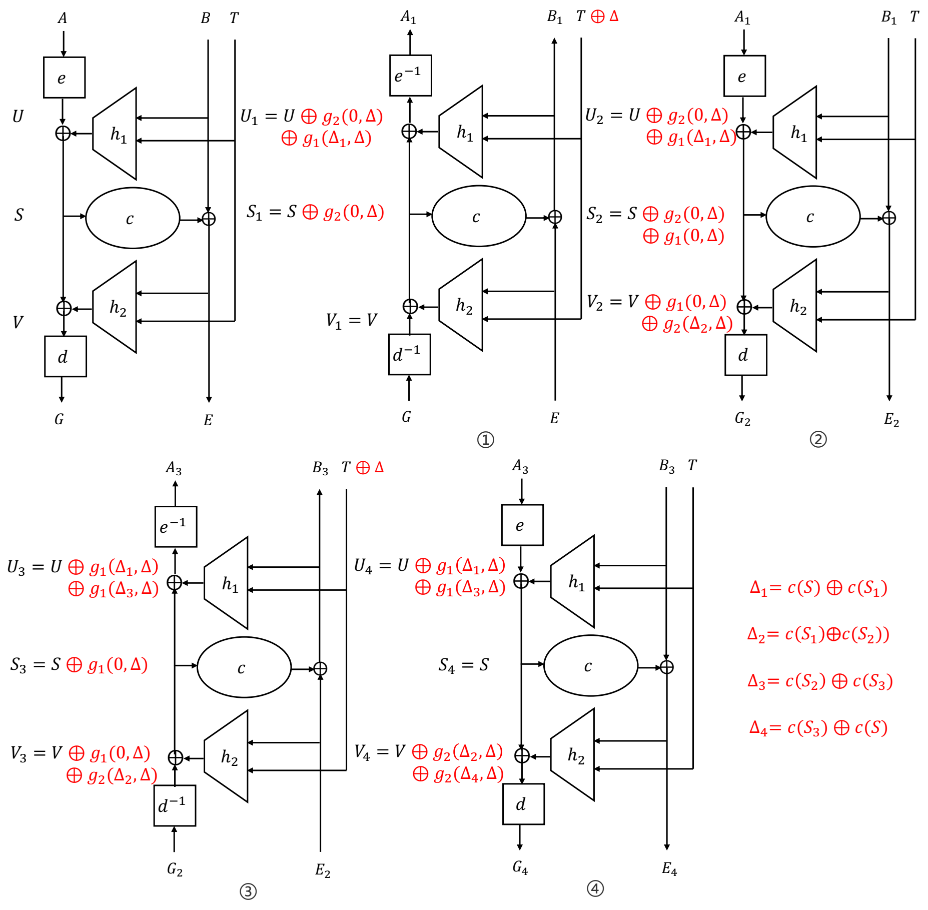

A
B T
e
U
h_1
S
c
V
h_2
d
G E
A_1
B_1 T ⊕ Δ
e^{-1}
U_1 = U ⊕ g_2(0, Δ) ⊕ g_1(Δ_1, Δ)
S_1 = S ⊕ g_2(0, Δ)
V_1 = V
d^{-1}
G E
①
A_1
B_1 T
U_2 = U ⊕ g_2(0, Δ) ⊕ g_1(Δ_1, Δ)
S_2 = S ⊕ g_2(0, Δ) ⊕ g_1(0, Δ)
V_2 = V ⊕ g_1(0, Δ) ⊕ g_2(Δ_2, Δ)
G_2 E_2
②
A_3
B_3 T ⊕ Δ
U_3 = U ⊕ g_1(Δ_1, Δ) ⊕ g_1(Δ_3, Δ)
S_3 = S ⊕ g_1(0, Δ)
V_3 = V ⊕ g_1(0, Δ) ⊕ g_2(Δ_2, Δ)
G_2 E_2
③
A_3
B_3 T
U_4 = U ⊕ g_1(Δ_1, Δ) ⊕ g_1(Δ_3, Δ)
S_4 = S
V_4 = V ⊕ g_2(Δ_2, Δ) ⊕ g_2(Δ_4, Δ)
G_4 E_4
④
Δ_1= c(S) ⊕ c(S_1)
Δ_2= c(S_1)⊕c(S_2))
Δ_3= c(S_2) ⊕ c(S_3)
Δ_4= c(S_3) ⊕ c(S)

Fig. 6. The process of $\mathbf{D}^{T'}, \mathbf{E}^{T}, \mathbf{D}^{T'}, \mathbf{E}^{T}$ (Query 1 to Query 4)

Proof. We define the intermediate values (U, S, V) and (U_i, S_i, V_i) as before. For different values of T and $T' = T \oplus \Delta$, assuming that $\mathbf{E}^T(A\|B) = G\|E$, we compute the intermediate values. These results are illustrated in Figure 6.

Before queries, we have

$$\begin{aligned}
&U = e(A),\\
&S = U \oplus h_1(B, T) = V \oplus h_2(E, T),\\
&V = d^{-1}(G),\\
&c(S) = B \oplus E. \qquad (2)
\end{aligned}$$

For Query 1:

$$V_1 = V,$$

$$\begin{aligned}
S_1 &= S \oplus h_2(E,T) \oplus h_2(E,T\oplus \varDelta) = S \oplus g_2(0,\varDelta),\\
U_1 &= S_1 \oplus h_1(B_1, T\oplus\varDelta) = S \oplus g_2(0,\varDelta) \oplus h_1(B,T) \oplus g_1(\varDelta_1,\varDelta)\\
&= U \oplus g_2(0,\varDelta) \oplus g_1(\varDelta_1,\varDelta),\\
c(S_1) &= B_1 \oplus E,
\end{aligned}$$

where

$$\varDelta_1 = B \oplus B_1 = c(S) \oplus E \oplus c(S_1) \oplus E = c(S) \oplus c(S_1).$$

For Query 2:

$$\begin{aligned}
U_2 &= U_1 = U \oplus g_2(0,\varDelta) \oplus g_1(\varDelta_1,\varDelta),\\
S_2 &= S_1 \oplus h_1(B_1,T\oplus\varDelta) \oplus h_1(B_1,T) = S \oplus g_2(0,\varDelta) \oplus g_1(0,\varDelta),\\
V_2 &= S_2 \oplus h_2(E_2,T) = S \oplus g_2(0,\varDelta) \oplus g_1(0,\varDelta) \oplus h_2(E,T) \oplus g_2(\varDelta_2,0)\\
&= V \oplus g_1(0,\varDelta) \oplus g_2(\varDelta_2,\varDelta),\\
c(S_2) &= B_1 \oplus E_2,
\end{aligned}$$

where

$$\varDelta_2 = E \oplus E_2 = c(S_1) \oplus B_1 \oplus c(S_2) \oplus B_1 = c(S_1) \oplus c(S_2).$$

For Query 3:

$$\begin{aligned}
V_3 &= V_2 = V \oplus g_1(0,\varDelta) \oplus g_2(\varDelta_2,\varDelta),\\
S_3 &= S_2 \oplus h_2(E_2,T) \oplus h_2(E_2,T\oplus\varDelta) = S \oplus g_1(0,\varDelta),\\
U_3 &= S_3 \oplus h_1(B_3,T\oplus\varDelta) = S \oplus g_1(0,\varDelta) \oplus h_1(B_1,T) \oplus g_1(\varDelta_3,\varDelta)\\
&= S \oplus g_1(0,\varDelta) \oplus h_1(B,T) \oplus g_1(\varDelta_1,0) \oplus g_1(\varDelta_3,\varDelta)\\
&= U \oplus g_1(\varDelta_1,\varDelta) \oplus g_1(\varDelta_3,\varDelta),\\
c(S_3) &= B_3 \oplus E_2,
\end{aligned}$$

where

$$\varDelta_3 = B_1 \oplus B_3 = c(S_2) \oplus E_2 \oplus c(S_3) \oplus E_2 = c(S_2) \oplus c(S_3).$$

For Query 4:

$$\begin{aligned}
U_4 &= U_3 = U \oplus g_1(\varDelta_1,\varDelta) \oplus g_1(\varDelta_3,\varDelta),\\
S_4 &= S_3 \oplus h_1(B_3,T\oplus\varDelta) \oplus h_1(B_3,T) = S \oplus g_1(0,\varDelta) \oplus g_1(0,\varDelta) = S,\\
V_4 &= S_4 \oplus h_2(E_4,T) = S \oplus h_2(E_2,T) \oplus g_2(\varDelta_4,0)\\
&= S \oplus h_2(E,T) \oplus g_2(\varDelta_2,0) \oplus g_2(\varDelta_4,0) = V \oplus g_2(\varDelta_2,0) \oplus g_2(\varDelta_4,0),\\
c(S) &= B_3 \oplus E_4, \qquad (3)
\end{aligned}$$

where

$$\varDelta_4 = E_2 \oplus E_4 = c(S_3) \oplus B_3 \oplus c(S) \oplus B_3 = c(S_3) \oplus c(S).$$

From (2) and (3) we have:

$$c(S) = B \oplus E = B_3 \oplus E_4.$$

So

$$B = E \oplus B_3 \oplus E_4.$$

□

The universal hash functions h_1, h_2 in XCBv1, XCBv2 and XCBv2fb are separable. By Proposition 1, Attack 2 holds for all these versions.

5.5 Attack 3: full plaintext recovery attack with 7 queries

In Attack 1, we use 3 queries to recover plaintext A; in Attack 2, we use 4 queries to recover plaintext B. So the question arises, can we recover the full plaintext $A\|B$?

The answer is yes. Notice that in the 4th query, the intermediate value S_4 goes back to S. Therefore $\Delta_i = c(S_{i-1}) \oplus c(S_i), i = 1, 2, \cdots$ will also begin a four-step cycle, where $S_0 = S$. This implies that after an additional four queries, the intermediate values will return to the original (U, S, V). In fact, in the 7th query we obtain the plaintext of $A\|B$.

For the completeness of the proof, we calculate the intermediate values as before. The correctness of Attack 3 comes from the following proposition:

Proposition 3. *In XCB, for any distinct T and T', if h_1, h_2 are separable, we have*

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}.$$

Proof. This proposition is an extension of Proposition 2, which adds 3 more queries on top of the 4 queries in Proposition 2, proving that querying XCB 7 times can get the entire plaintext directly.

For distinct T and T', we examine $\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}(G\|E)$. Let $T' = T \oplus \Delta$, the query steps are as follows. Queries 1 to 4 are the same as Proposition 2, and Queries 5 to 7 are shown in Figure 7.

For Query 5,

$$\begin{aligned}
V_5 &= V_4 = V \oplus g_2(\Delta_2, 0) \oplus g_2(\Delta_4, 0),\\
S_5 &= S_4 \oplus h_2(E_4, T) \oplus h_2(E_4, T \oplus \Delta) = S \oplus g_2(0, \Delta) = S_1,\\
U_5 &= S_5 \oplus h_1(B_5, T \oplus \Delta) = S_1 \oplus h_1(B_3, T \oplus \Delta) \oplus g_1(\Delta_5, 0)\\
&= S_1 \oplus U_3 \oplus S_3 \oplus g_1(\Delta_1, 0)\\
&= S \oplus g_2(0, \Delta) \oplus U \oplus S \oplus g_1(\Delta_1, 0) \oplus g_1(\Delta_3, \Delta) \oplus g_1(\Delta_1, 0)\\
&= U \oplus g_2(0, \Delta) \oplus g_1(\Delta_3, \Delta),\\
c(S_1) &= B_5 \oplus E_4 = B_1 \oplus E.
\end{aligned}$$

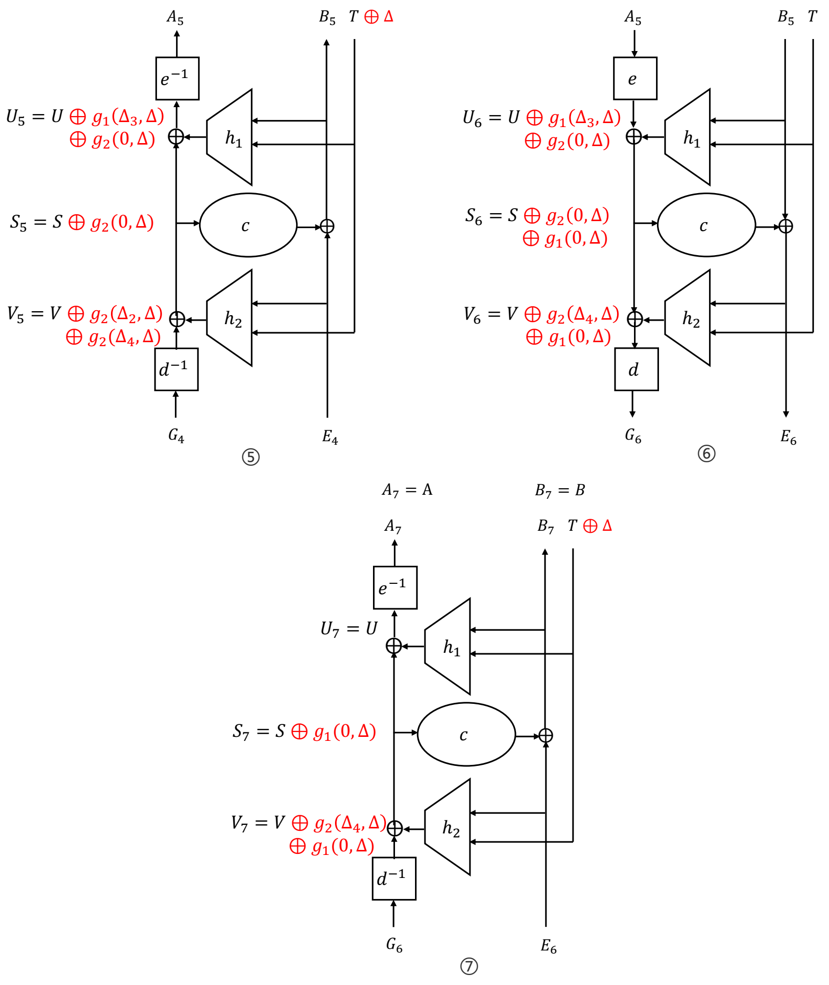

A_5
B_5 $T \oplus \Delta$
e^{-1}
$U_5 = U \oplus g_1(\Delta_3, \Delta) \oplus g_2(0, \Delta)$
h_1
$S_5 = S \oplus g_2(0, \Delta)$
c
$V_5 = V \oplus g_2(\Delta_2, \Delta) \oplus g_2(\Delta_4, \Delta)$
h_2
d^{-1}
G_4
E_4
⑤
A_5
B_5 T
e
$U_6 = U \oplus g_1(\Delta_3, \Delta) \oplus g_2(0, \Delta)$
h_1
$S_6 = S \oplus g_2(0, \Delta) \oplus g_1(0, \Delta)$
c
$V_6 = V \oplus g_2(\Delta_4, \Delta) \oplus g_1(0, \Delta)$
h_2
d
G_6
E_6
⑥
$A_7 = A$
$B_7 = B$
A_7
B_7 $T \oplus \Delta$
e^{-1}
$U_7 = U$
h_1
$S_7 = S \oplus g_1(0, \Delta)$
c
$V_7 = V \oplus g_2(\Delta_4, \Delta) \oplus g_1(0, \Delta)$
h_2
d^{-1}
G_6
E_6
⑦

Fig. 7. The process of $\mathbf{D}^{T'}, \mathbf{E}^{T}, \mathbf{D}^{T'}$ (Query 5 to Query 7)

where

$$\Delta_5 = B_3 \oplus B_5 = c(S) \oplus E_4 \oplus c(S_1) \oplus E_4 = c(S) \oplus c(S_1) = \Delta_1.$$

For Query 6,

$$\begin{aligned}
U_6 &= U_5 = U \oplus g_2(0, \Delta) \oplus g_1(\Delta_3, \Delta), \\
S_6 &= S_5 \oplus h_1(B_5, T \oplus \Delta) \oplus h_1(B_5, T) = S \oplus g_2(0, \Delta) \oplus g_1(0, \Delta) = S_2, \\
V_6 &= S_6 \oplus h_2(E_6, T) = S_2 \oplus h_2(E_4, T) \oplus g_2(\Delta_6, 0) \\
&= S_2 \oplus V_4 \oplus S_4 \oplus g_2(\Delta_2, 0) \\
&= S \oplus g_2(0, \Delta) \oplus g_1(0, \Delta) \oplus V \oplus S \oplus g_2(\Delta_2, 0) \oplus g_2(\Delta_4, 0) \oplus g_2(\Delta_2, 0)
\end{aligned}$$

$$\begin{aligned} &= V \oplus g_1(0, \varDelta) \oplus g_2(\varDelta_4, \varDelta), \\ c(S_2) &= B_5 \oplus E_6 = B_1 \oplus E_2, \end{aligned}$$

where

$$\varDelta_6 = E_4 \oplus E_6 = c(S_1) \oplus B_5 \oplus c(S_2) \oplus B_5 = c(S_1) \oplus c(S_2) = \varDelta_2.$$

For Query 7,

$$\begin{aligned} V_7 &= V_6 = V \oplus g_1(0, \varDelta) \oplus g_2(\varDelta_4, \varDelta), \\ S_7 &= S_6 \oplus h_2(E_6, T) \oplus h_2(E_6, T \oplus \varDelta) = S \oplus g_1(0, \varDelta) = S_3, \\ U_7 &= S_7 \oplus h_1(B_7, T \oplus \varDelta) = S_3 \oplus h_1(B_5, T \oplus \varDelta) \oplus g_1(\varDelta_7, 0) \\ &= S_3 \oplus U_5 \oplus S_5 \oplus g_1(\varDelta_3, 0) \\ &= S \oplus g_1(0, \varDelta) \oplus U \oplus S \oplus g_1(\varDelta_3, \varDelta) \oplus g_1(\varDelta_3, 0) = U, \\ c(S_3) &= B_7 \oplus E_6 = B_3 \oplus E_2, \end{aligned}$$

where

$$\varDelta_7 = B_5 \oplus B_7 = c(S_2) \oplus E_6 \oplus c(S_3) \oplus E_6 = c(S_2) \oplus c(S_3) = \varDelta_3.$$

Then we have

$$A_7 = e^{-1}(U_7) = e^{-1}(U) = e^{-1}(e(A)) = A.$$

and

$$\begin{aligned} B_7 =& c(S_3) \oplus E_6 = B_3 \oplus E_2 \oplus E_6 = B_3 \oplus E_2 \oplus B_5 \oplus B_1 \oplus E_2 \\ =& B_1 \oplus B_3 \oplus B_5 = B_1 \oplus B_3 \oplus E_4 \oplus B_1 \oplus E = E \oplus B_3 \oplus E_4 \\ =& B. \end{aligned}$$

So we have

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}(G \| E) = A \| B = \mathbf{D}^{T}(G \| E).$$

And

$$\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'}\mathbf{E}^{T}\mathbf{D}^{T'} = \mathbf{D}^{T}.$$

□

In Appendix B, we provide experimental validation of the three plaintext recovery attacks on XCBv2, the standard version which uses the underlying block cipher AES-128.

6 Modification

All versions of XCB have security vulnerabilities. How can these be addressed? If we want to handle one-block-length messages, we should not use the XCB

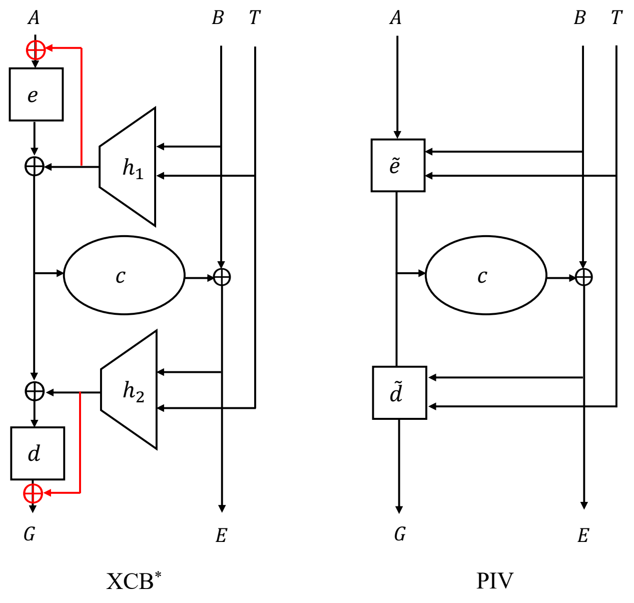


Fig. 8. XCB* and PIV

structure. Attack 1 shows the weakness of the structure. Even if we use stronger components, such as making h_1 and h_2 pseudorandom functions, XCB still fails to achieve CCA security.

Our suggestion is to XOR the output of $h_i, i = 1, 2$ into both before and after the block cipher as shown on the left of Figure 8. The modification only adds two XOR operations. We call the new structure XCB*. If we look at the UHF and its left block cipher as a whole, it is exactly an LRW2 construction. The same method was used in [1] to strengthen the security of GCM. Consequently, the whole structure changes to the PIV structure proposed by Shrimpton et al. [34]. We prove that when h_1 and h_2 are AXU hash functions, e and d are SPRPs, and c is a secure IV-based stream cipher, then XCB* is an STPRP.

The following is the pseudocode for XCB* and PIV. We note that the components in XCB* and PIV, such as $\tilde{e}, \tilde{d}, e, d, h_1, h_2, c$, are all independent.

LRW2 is an STPRP if the underlying block cipher is an SPRP and the UHFs are AXU, as proved by the Theorem 2 in [24].

Lemma 2 (Theorem 2 in [24]). *Let $\tilde{e}[e, h]^T(P) = e(P \oplus h(T)) \oplus h(T)$, where e is a block cipher, and h is ϵ-AXU. Then $\tilde{e}$ is an STPRP. Specifically, for any adversary $\mathcal{A}$ making q queries, there exists an adversary $\mathcal{B}$*

$$\mathbf{Adv}^{\mathrm{stprp}}_{\tilde{e}[e,h]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{sprp}}_{e}(\mathcal{B}) + 3\epsilon q^2.$$

Algorithm 1 The XCB* encryption and decryption

```
XCB*.$\mathbf{E}^T(A\|B)$
    $\varDelta_1 \leftarrow h_1(B,T)$
    $S \leftarrow e(A \oplus \varDelta_1) \oplus \varDelta_1$
    $E \leftarrow B \oplus c(S)$
    $\varDelta_2 \leftarrow h_2(E,T)$
    $G \leftarrow d(V \oplus \varDelta_2) \oplus \varDelta_2$
    return $G\|E$

XCB*.$\mathbf{D}^T(G\|E)$
    $\varDelta_2 \leftarrow h_2(E,T)$
    $S \leftarrow d^{-1}(G \oplus \varDelta_2) \oplus \varDelta_2$
    $B \leftarrow E \oplus c(S)$
    $\varDelta_1 \leftarrow h_1(B,T)$
    $A \leftarrow e^{-1}(S \oplus \varDelta_1) \oplus \varDelta_1$
    return $A\|B$
```

Algorithm 2 The PIV encryption and decryption

```
PIV.$\mathbf{E}^T(A\|B)$
    $S \leftarrow \tilde{e}^{(B,T)}(A)$
    $E \leftarrow B \oplus c(S)$
    $G \leftarrow \tilde{d}^{(E,T)}(S)$
    return $G\|E$

PIV.$\mathbf{D}^T(G\|E)$
    $S \leftarrow \tilde{d}^{-1(E,T)}(G)$
    $B \leftarrow E \oplus c(S)$
    $A \leftarrow \tilde{e}^{-1(B,T)}(B)$
    return $A\|B$
```

Lemma 3. *For PIV$[\tilde{e}, \tilde{d}, c]$, let $\mathcal{A}$ be an adversary making q queries. Then there exist adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, each making q queries, such that*

$$\mathbf{Adv}^{\mathrm{stprp}}_{\mathrm{PIV}[\tilde{e},\tilde{d},c]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{stprp}}_{\tilde{e}}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{stprp}}_{\tilde{d}}(\mathcal{C}) + \mathbf{Adv}^{\mathrm{ivrnd}}_{c}(\mathcal{D}) + \frac{3q^2}{2^{n+1}}.$$

Lemma 3 is similar to Theorem 1 in [34]. The difference is the security notion of the stream cipher. In Lemma 3 $S\|c(S)$ is indistinguishable from a random string for a random IV S, whereas in Theorem 1 in [34], $c(S)$ is indistinguishable from a random string for a nonce S which never repeats. The security notions based on random IV and nonce are referred to as ivRND and nRND, respectively. They are similar to IV-based encryption (ivE) and nonce-based encryption (nE) in [29], or weak pseudorandom function (wPRF) and pseudorandom function (PRF) in [32,28]. We choose the weaker security notion due to the fact that the stream cipher in XCBv1 and XCBv2 is ivRND secure but not nRND secure. For the sake of completeness, we provide the proof in Appendix D.

In PIV$[\tilde{e}, \tilde{d}, c]$, let $\tilde{e}^T(P) = e(P \oplus h_1(T)) \oplus h_1(T)$ and $\tilde{d}^T(P) = d(P \oplus h_2(T)) \oplus h_2(T)$, then PIV$[\tilde{e}, \tilde{d}, c]$ becomes XCB*$[e, h_1, d, h_2, c]$ as shown in Figure 8. Combine Lemma 2 and 3, we have the following theorem.

Theorem 1. *Assume that h_1 and h_2 are both ϵ-AXU. For XCB*$[e, h_1, d, h_2, c]$, let $\mathcal{A}$ be an adversary making q queries. Then there exist adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, making q queries, respectively, such that*

$$\mathbf{Adv}^{\mathrm{stprp}}_{\mathrm{XCB*}[e,d,h_1,h_2,c]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{sprp}}_{e}(\mathcal{B}) + \mathbf{Adv}^{\mathrm{sprp}}_{d}(\mathcal{C}) + \mathbf{Adv}^{\mathrm{ivrnd}}_{c}(\mathcal{D}) + \frac{3q^2}{2^{n+1}} + 6\epsilon q^2.$$

Note that the fixing method works for XCBv1 but is not applicable for XCBv2 due to its UHF h_1 is not an AXU hash function. For the add-tweak method, if we restrict the message length to one block, XCB* becomes a tweakable block cipher with the Chained LRW2 (CLRW2) construction, which is CCA secure beyond birthday bound [22].

7 Discussions

Requirement of CCA security. STPRP is an essential security requirement for TEM. As IEEE 1619.2 [20] states the security goal as "Interacting with the encryption and decryption routines, it should be infeasible to distinguish them from a random permutation and its inverse. Moreover, varying the plaintext length and/or the associated data should look like using a different and independent key."

In NIST's draft for the requirements for Accordion mode [8], it also sets the security goal as STPRP. The document describes three categories of applications for an accordion mode: authenticated encryption with associated data (AEAD), tweakable encryption for storage devices, and deterministic authenticated encryption. All applications require the underlying TEM to be STPRP-secure.

In full disk encryption, the plaintext P in the sector indexed by T is encrypted into $C = \mathbf{E}^T(P)$. Given (T, C), assume we can access the encryption oracle $\mathbf{E}^T$ and the decryption oracle of another sector $\mathbf{D}^{T'}$. Then, with 4 queries, Attack 2 can recover the plaintext that excludes one block. With 7 queries, Attack 3 can recover the full plaintext.

STPRP implies security against plaintext recovery attacks. Our findings reveal that XCB does not meet the STPRP criteria, indicating previous security proofs for XCBv2 [26], XCBv2fb and XCBv1 [5] are all flawed.

Flaws in previous proofs. The security of STPRP is derived from the indistinguishability between the results of encryption or decryption queries and random strings, as discussed in a series of papers [18,19,16,5]. We assume that the target plaintext is connected by the ciphertext through the 0-th query, and (U_i, S_i, V_i) are the intermediate values for i-th query as shown in Figure 6 and 7, where $i = 0, 1, 2, \cdots$, and $(U_0, S_0, V_0) = (U, S, V)$.

In XCB, if the underlying block cipher is a random permutation, the output randomness for each query depends on the following multi sets: $\mathcal{U}$ consisting of U_i obtained from decryption queries, $\mathcal{U}$ consisting V_i obtained from encryption queries and $\mathcal{S}$ consisting the inputs to block cipher in the component c, which are determined by S_i obtained from encryption or decryption queries.

The multi sets of $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{S}$ correspond to $\mathcal{R}_1$ (Rng_1), $\mathcal{R}_3$ (Rng_3) and $\mathcal{D}_2$ (Dom_2) respectively in the security proof of XCBv2fb (XCBv2fb) in [5].

The negligible collision probability in $\mathcal{U}$ ($\mathcal{V}$) induces the left-output randomness of each decryption (encryption) query, and $\mathcal{S}$ the right-output randomness. The event Ω defined in the proof of XCBv2 in [26] corresponds to the the absence of collisions in $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{S}$.

For Attack 1, in the third query, $U_3 = U$. For Attacks 2 and 3, in the forth query, $S_4 = 4$ and in the seventh query, $U_7 = U$.

As a result, all collision probabilities converge to 1, thereby validating the proofs in [26] and [5].

Structure weakness of XCB. The underlying issue with XCB's security lies primarily in its method of integrating tweaks. Though XCB and HCTR are both categorized within the Hash-CTR-Hash framework—leveraging UHFs in the first and third layers, with the CTR mode in the middle—their tweak handling is fundamentally different. XCB adopts the LRW1-type method, while HCTR employs the more secure LRW2-type method. See Appendix C for more detail. The LRW1 construction inherently makes XCB vulnerable to CCA attacks, as demonstrated by Attack 1.

Another critical vulnerability is the separability of UHFs, which allows for attacks on schemes supporting arbitrary message lengths. While one might hope that selecting an appropriate UHF could mitigate these attacks, this is unlikely. Separability is a nearly unavoidable characteristic, as most practical UHFs rely on basic algebraic operations. For instance, GHASH in GCM [27], POLYVAL in AES-GCM-SIV [14], Poly1305 in ChaCha20-Poly1305 [31], and BPE in TET [17] etc. are all separable for some operation.

8 Conclusions

In this paper, we present three plaintext recovery attacks that are applicable to all versions of XCB. When $m = n$, 3 queries are sufficient to recover the plaintext A of (T, G). The attacks are applicable to any scheme utilizing the XCB structure. When $m > n$, 4 queries are enough to recover the *partial* plaintext B of $(T, G\|E)$, and 7 queries are enough to recover the *full* plaintext $A\|B$ of $(T, G\|E)$. These attacks apply to any scheme that uses the XCB structure with separable UHFs.

Hence, in the case of XCBv1, we need 7 queries to recover the full plaintext. In contrast, for XCBv2 (XCBv2fb), our warm-up method uses only 3 queries. We leave it as an open problem to determine whether there are full plaintext recovery attacks against any version of XCB that require fewer queries.

As a final recommendation, we suggest reconsidering the continued use of XCB. Its inherent *structural weaknesses*, particularly in how it manages tweaks and the separability of its UHFs, make it unsuitable for secure applications. A more robust alternative would be to modify the XCB structure by incorporating two additional XOR operations, effectively transforming it into the PIV construction [34]. This adjustment provides a pathway to stronger security guarantees.

Acknowledgments

The authors would like to thank Yaobin Shen for the insightful discussions, which have significantly contributed to the improvement of this paper. The work is supported by the National Key Research and Development Program of China (No. 2023YFB3105802).

References

1. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Advances in Cryptology - CRYPTO 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 3–33. Springer (2017). https://doi.org/10.1007/978-3-319-63697-9_1 19
2. Bao, Z., Guo, C., Guo, J., Song, L.: TNT: how to tweak a block cipher. In: Advances in Cryptology - EUROCRYPT 2020. vol. 12106, pp. 641–673. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_22 2
3. Bhati, A.S., Verbauwhede, M., Andreeva, E.: Breaking, repairing and enhancing XCBv2 into the tweakable enciphering mode GEM. Cryptology ePrint Archive, Paper 2024/1554 (2024), https://eprint.iacr.org/2024/1554 1, 2, 4, 8, 10
4. Bhaumik, R., Nandi, M.: An inverse-free single-keyed tweakable enciphering scheme. In: Advances in Cryptology - ASIACRYPT 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 159–180. Springer (2015). https://doi.org/10.1007/978-3-662-48800-3_7 2
5. Chakraborty, D., Hernandez-Jimenez, V., Sarkar, P.: Another look at XCB. Cryptogr. Commun. **7**(4), 439–468 (2015). https://doi.org/10.1007/S12095-015-0127-8 2, 3, 4, 5, 8, 21, 22
6. Chakraborty, D., Sarkar, P.: A new mode of encryption providing a tweakable strong pseudo-random permutation. In: Fast Software Encryption, 13th International Workshop, FSE 2006, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4047, pp. 293–309. Springer (2006). https://doi.org/10.1007/11799313_19 2, 29
7. Chakraborty, D., Sarkar, P.: HCH: A new tweakable enciphering scheme using the hash-counter-hash approach. IEEE Trans. Inf. Theory **54**(4), 1683–1699 (2008). https://doi.org/10.1109/TIT.2008.917623 2, 29
8. Chen, Y.L., Davidson, M., Dworkin, M., Kang, J., Kelsey, J., Sasaki, Y., Chang, D., Mouha, N., Thompson, A.: Proposal of requirements for an Accordion mode: Discussion draft for the NIST Accordion mode workshop 2024 (2024) 2, 21
9. Crowley, P., Biggers, E.: Adiantum: length-preserving encryption for entry-level processors. IACR Trans. Symmetric Cryptol. **2018**(4), 39–61 (2018). https://doi.org/10.13154/TOSC.V2018.I4.39-61 2, 29
10. Crowley, P., Huckleberry, N., Biggers, E.: Length-preserving encryption with HCTR2. IACR Cryptol. ePrint Arch. p. 1441 (2021), https://eprint.iacr.org/2021/1441 2, 29
11. Datta, N., Dutta, A., Ghosh, S., Nandi, H.: Optimally secure TBC based accordion mode. IACR Cryptol. ePrint Arch. p. 2053 (2024), https://eprint.iacr.org/2024/2053 2
12. Dobraunig, C., Matusiewicz, K., Mennink, B., Tereschenko, A.: Efficient instances of docked double decker with AES. IACR Cryptol. ePrint Arch. p. 84 (2024), https://eprint.iacr.org/2024/084 2
13. Fleischmann, E., Forler, C., Lucks, S., Wenzel, J.: McOE: A family of almost foolproof on-line authenticated encryption schemes. Cryptology ePrint Archive, Paper 2011/644 (2011), https://eprint.iacr.org/2011/644 11
14. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: specification and analysis. IACR Cryptol. ePrint Arch. p. 168 (2017), http://eprint.iacr.org/2017/168 22

15. Gunsing, A., Daemen, J., Mennink, B.: Deck-based wide block cipher modes and an exposition of the blinded keyed hashing model. IACR Trans. Symmetric Cryptol. **2019**(4), 1–22 (2019). https://doi.org/10.13154/TOSC.V2019.I4.1-22 1
16. Halevi, S.: EME*: Extending EME to handle arbitrary-length messages with associated data. In: Progress in Cryptology - INDOCRYPT 2004, Proceedings. Lecture Notes in Computer Science, vol. 3348, pp. 315–327. Springer (2004). https://doi.org/10.1007/978-3-540-30556-9_25 1, 21, 29
17. Halevi, S.: Invertible universal hashing and the TET encryption mode. In: Advances in Cryptology - CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622, pp. 412–429. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_23 2, 22, 29
18. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Advances in Cryptology - CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 482–499. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_28 1, 21, 29, 31
19. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Topics in Cryptology - CT-RSA 2004. Lecture Notes in Computer Science, vol. 2964, pp. 292–304. Springer (2004). https://doi.org/10.1007/978-3-540-24660-2_23 1, 21, 29
20. IEEE 1619.2: IEEE standard for wide-block encryption for shared storage media (2011) 2, 3, 13, 21
21. Jha, A., Khairallah, M., Nandi, M., Saha, A.: Tight security of TNT and beyond - attacks, proofs and possibilities for the cascaded LRW paradigm. In: Advances in Cryptology - EUROCRYPT 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 249–279. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_9 3, 11
22. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Advances in Cryptology - CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 14–30. Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_2 2, 21
23. Lee, B.: A BBB secure accordion mode from hctr. NIST Workshop on the Requirements for an Accordion Cipher Mode (2024) 2, 29
24. Liskov, M.D., Rivest, R.L., Wagner, D.A.: Tweakable block ciphers. In: Advances in Cryptology - CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002). https://doi.org/10.1007/3-540-45708-9_3 2, 19
25. McGrew, D.A., Fluhrer, S.R.: The extended codebook (XCB) mode of operation. IACR Cryptol. ePrint Arch. p. 278 (2004), http://eprint.iacr.org/2004/278 2, 3, 4, 5, 13, 29
26. McGrew, D.A., Fluhrer, S.R.: The security of the extended codebook (XCB) mode of operation. In: Selected Areas in Cryptography, 14th International Workshop, SAC 2007. Lecture Notes in Computer Science, vol. 4876, pp. 311–327. Springer (2007). https://doi.org/10.1007/978-3-540-77360-3_20 2, 3, 4, 5, 7, 13, 21, 22, 29
27. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode (GCM) of operation. In: Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer (2004). https://doi.org/10.1007/978-3-540-30556-9_27 22
28. Minematsu, K., Matsushima, T.: Tweakable enciphering schemes from hash-sum-expansion. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) Progress in Cryptology

- INDOCRYPT 2007. Lecture Notes in Computer Science, vol. 4859, pp. 252–267. Springer (2007). https://doi.org/10.1007/978-3-540-77026-8_19 20

29. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Advances in Cryptology - EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 257–274. Springer (2014). https://doi.org/10.1007/978-3-642-55220-5_15 20
30. Nandi, M.: An efficient SPRP-secure construction based on pseudo random involution. IACR Cryptol. ePrint Arch. p. 92 (2008), http://eprint.iacr.org/2008/092 2, 3, 4, 13, 29
31. Nir, Y., Langley, A.: Chacha20 and poly1305 for IETF protocols. RFC **7539**, 1–45 (2015). https://doi.org/10.17487/RFC7539 22
32. Pietrzak, K., Sjödin, J.: Range extension for weak PRFs; the good, the bad, and the ugly. In: Advances in Cryptology - EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 517–533. Springer (2007). https://doi.org/10.1007/978-3-540-72540-4_30 20
33. Sarkar, P.: Improving upon the TET mode of operation. In: Information Security and Cryptology - ICISC 2007. Lecture Notes in Computer Science, vol. 4817, pp. 180–192. Springer (2007). https://doi.org/10.1007/978-3-540-76788-6_15 2, 29
34. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Advances in Cryptology - ASIACRYPT 2013, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 405–423. Springer (2013). https://doi.org/10.1007/978-3-642-42033-7_21 2, 19, 20, 22, 29
35. Wang, P., Feng, D., Wu, W.: HCTR: A variable-input-length enciphering mode. In: Information Security and Cryptology, First SKLOIS Conference, CISC 2005. Lecture Notes in Computer Science, vol. 3822, pp. 175–188. Springer (2005). https://doi.org/10.1007/11599548_15 2, 29

A The Description of XCBv1 and XCBv2

The encryption of XCBv1 is described in Algorithm 3, the encryption of XCBv2 is described in Algorithm 4.

B Experimental data of Attacks

We conduct experimental verifications for the attacks described in the paper. Our target is XCBv2, which is part of the IEEE 1619.2 standard, and the underlying block cipher used is AES-128.

We assume that $C = \mathbf{E}_K^T(P)$, and set the key

$$K = \texttt{000102030405060708090a0b0c0d0e0f},$$

and two different tweaks

$$T = \texttt{80000000000000000000000000000000},$$

$$T' = \texttt{81010101010101010101010101010101}.$$

Algorithm 3 The XCBv1 encryption operation. Given a key $K \in \{0,1\}^n$, a plaintext $P \in \{0,1\}^m$, where $m \in [n, 2^{39}]$, and a tweak $T \in \{0,1\}^t$, where $t \in [0, 2^{39}]$, this operation returns a ciphertext $C \in \{0,1\}^m$.

```
$H_1 \leftarrow e_K(0^{n-3}\|001)$
$H_2 \leftarrow e_K(0^{n-3}\|011)$
$K_e \leftarrow e_K(0^n)$
$K_d \leftarrow e_K(0^{n-3}\|100)$
$K_c \leftarrow e_K(0^{n-3}\|010)$
$A \leftarrow P[0, n-1]$
$B \leftarrow P[n, m-1]$
$U \leftarrow e_{K_e}(A)$
$S \leftarrow U \oplus h_{1H_1}(B, T)$
$E \leftarrow B \oplus c_{K_c}(S)$
$V \leftarrow S \oplus h_{2H_2}(E, T)$
$G \leftarrow d_{K_d}(V)$
$C \leftarrow G\|E$
return $C$
```

Algorithm 4 The XCBv2 encryption operation. Given a key $K \in \{0,1\}^k$, a plaintext $P \in \{0,1\}^m$, where $m \in [n, 2^{39}]$, and a tweak $T \in \{0,1\}^t$, where $t \in [0, 2^{39}]$, this operation returns a ciphertext $C \in \{0,1\}^m$.

```
$H \leftarrow e_K(0^n)$
$K_e \leftarrow \mathrm{msb}_k(e_K(0^{n-3}\|001)\|e_K(0^{n-3}\|010))$
$K_d \leftarrow \mathrm{msb}_k(e_K(0^{n-3}\|011)\|e_K(0^{n-3}\|100))$
$K_c \leftarrow \mathrm{msb}_k(e_K(0^{n-3}\|101)\|e_K(0^{n-3}\|110))$
$A \leftarrow P[m-n, m-1]$
$B \leftarrow P[0, m-n-1]$
$U \leftarrow e_{K_e}(A)$
$S \leftarrow U \oplus h_{1H}(B, T)$
$E \leftarrow U \oplus c_{K_c}(S)$
$V \leftarrow S \oplus h_{2H}(E, T)$
$G \leftarrow d_{K_d}(V)$
$C \leftarrow E\|G$
return $C$
```

Give (T, C), the following tree attacks try to recover the plaintext P.
We assume that alternate queries to $\mathbf{D}_K^{T'}$ and $\mathbf{E}_K^T$ get the following data:

$$P_1, C_2, P_3, C_4, \cdots$$

In the following, all data is represented as hexadecimal strings.

B.1 Attack 1 with 3 queries

We set $|P| = 128$ bits in Attack 1. We choose

$$P = \texttt{000102030405060708090a0b0c0d0e0f},$$

then the corresponding ciphertext

$$C = \texttt{1f20682d644ac931b4188e66714615d2}.$$

Three queries get the following data:

$$\begin{aligned} P_1 &= \texttt{1e73ed22a51ca40b55c320785e59b109}, \\ C_2 &= \texttt{7be9349f89dfce3fe07508e8fc2a741c}, \\ P_3 &= \texttt{000102030405060708090a0b0c0d0e0f}. \end{aligned}$$

Through 3 queries, we successfully recover P, which is equal to P_3.

B.2 Attack 2 and 3 with 4 and 7 queries respectively

We set $|P| = 384$ bits, three blocks, and choose

$$\begin{aligned} P =& \texttt{000102030405060708090a0b0c0d0e0f} \\ & \texttt{101112131415161718191a1b1c1d1e1f} \\ & \texttt{202122232425262728292a2b2c2d2e2f}. \end{aligned}$$

The corresponding ciphertext is:

$$\begin{aligned} C =& \texttt{f950309c130bf8a7a939ea6aedac65f4} \\ & \texttt{c59bc42086ac51106ce49731f244d233} \\ & \texttt{f624d22b23ec8e403f6df24f93b02691}. \end{aligned}$$

We get the following results:

$$\begin{aligned} P_1 =& \texttt{d6d60805cb1050fdf04ecf3e9b8428ed} \\ & \texttt{595802e0b1ed5e228b8962f6d7176dfe} \\ & \texttt{0a6188cfeeca35eaf35cfc02f875524c}, \end{aligned}$$

$$\begin{aligned} C_2 =& \texttt{959d3145ab745d50439a233f351045c2} \\ & \texttt{3b8c18f9e00536912ed5ee6e7faddfaa} \\ & \texttt{9f535a1eed7e72f81779a175333c95bc}, \end{aligned}$$

$$\begin{aligned} P_3 =& \texttt{4d7ff229f936a913b45fd260cb0a781e} \\ & \texttt{40bc56e4f6df1834020b0bc7a4daf981} \\ & \texttt{cc6ecea9e048ee9578679b5e8424ec7e}, \end{aligned}$$

$$\begin{aligned} C_4 = &\texttt{b42ec0b6ee3857b3156f32012aab13e5} \\ &\texttt{953680d764665f3376f686ed4a8335ad} \\ &\texttt{7b6be4e56282fe1520ae35e131a451d8}, \end{aligned}$$

$$\begin{aligned} P_5 = &\texttt{9ba8f82f3623ffe94c1817555c835efc} \\ &\texttt{09f5461753275001919b732a6fd08a60} \\ &\texttt{7a0da4a2463d96bdf0fcd2e9abe549bd}, \end{aligned}$$

$$\begin{aligned} C_6 = &\texttt{d8e3c16f5647f244ffccfb54f21733d3} \\ &\texttt{6b215c0e02cf38b234c7ffb2c76a3834} \\ &\texttt{82452f7ad79acad6ab9aee5bb8e55f03}, \end{aligned}$$

$$\begin{aligned} P_7 = &\texttt{000102030405060708090a0b0c0d0e0f} \\ &\texttt{101112131415161718191a1b1c1d1e1f} \\ &\texttt{202122232425262728292a2b2c2d2e2f}. \end{aligned}$$

If we partition the data into two parts: $P = B \| A$, $C = E \| G$, $P_i = B_i \| A_i$, $C_j = E_j \| G_j$, $i = 1, 3, 5, 7$, $i = 2, 4, 6$, where the left is 2-block, the right is 1-block. We can verify that $B = E \oplus B_3 \oplus E_4$. Therefore through 4 queries, we successfully recover the partial plaintext B.

Through 7 queries, we recover the plaintext P, which is equal to P_7.

B.3 Full plaintext recovery attack on XCBv2fb with 3 queries

We choose the same plaintxt P as in B.2 and set

$$\begin{aligned} \varDelta = &\texttt{000102030405060708090a0b0c0d0e0f} \\ &\texttt{101112131415161718191a1b1c1d1e1f} \\ &\texttt{00000000000000000000000000000000}. \end{aligned}$$

The first query gets the result $P_1 = \mathbf{D}_K^T(C \oplus \varDelta)$ and

$$\begin{aligned} P_1 = &\texttt{4ba865ceb32003a81181ebded20676a5} \\ &\texttt{8bb80bf65ec51bf79ccd34a6d39b14db} \\ &\texttt{f47742c7c7057c8126f5ed3a9f4816a3}. \end{aligned}$$

The second query gets the result $C_2 = \mathbf{E}_K^T(P_1 \oplus \Delta)$ and

$$\begin{aligned} C_2 = & \texttt{b2f85552a02bfb0fb8b801b43faa1351} \\ & \texttt{4e23cfd6d8694ae7f029a39721dfc6e8} \\ & \texttt{1177e0cb539d660d7be9264a5a0fff9d}. \end{aligned}$$

The third query gets the result $P_3 = \mathbf{D}_K^T(C_2 \oplus \Delta)$ and

$$\begin{aligned} P_3 = & \texttt{00000000000000000000000000000000} \\ & \texttt{00000000000000000000000000000000} \\ & \texttt{202122232425262728292a2b2c2d2e2f}. \end{aligned}$$

We partition the plaintext or ciphertext data into a 2-block and a 1-block strings as: $P = B\|A$, $C = E\|G$, $P_1 = B_1\|A_1$, $C_2 = E_2\|G_2$ and $P_3 = B_3\|A_3$.

We calculate the following expression:

$$\begin{aligned} ((E \oplus B_1 \oplus E_2)\|A_3) \oplus \Delta = & \texttt{000102030405060708090a0b0c0d0e0f} \\ & \texttt{101112131415161718191a1b1c1d1e1f} \\ & \texttt{202122232425262728292a2b2c2d2e2f}. \end{aligned}$$

This result corresponds exactly to the full plaintext P.

C Methods to Add Tweaks

In Section 5.1, we see that XCB becomes an LRW1-type tweakable block cipher in the case of $m = n$ (Figure 3), leading to a CCA attack.

By limiting the length of the message to one block, we transform TEM into a tweakable block cipher. This alteration enables us to concentrate exclusively on strategies for incorporating tweaks.

For another example, if we do it in HCTR [35] as in Figure 9, we can see that HCTR follows the LRW2-type method to add tweak.

We investigate TEM designs based on block ciphers in the current literature and find that there are mainly four methods to add tweaks, including LRW1-type, LRW2-type, CLRW2-type and TNT type. As shown in Figure 10, the LRW1-type method XORs the tweak between two block ciphers; the LRW2-type method XORs the UHF value of tweak before and after a block cipher; the CLRW2-type method is a chain of two LRW2; the TNT-type method XORs the tweak between two block ciphers twice.

We summarize block-cipher-based TEMs in the following list.

- LRW1-type: XCB [25,26], HCI [30], MXCB [30];
- LRW2-type: HCTR [35], HCTR2 [10], CMC [18], PEP [6], TET [17], HEH [33], HCH [7], Adiantum [9], DbHCTR [23];
- CLRW2-type: XCB*, TCT$_1$ [34];
- TNT-type: EME [19], EME* [16].

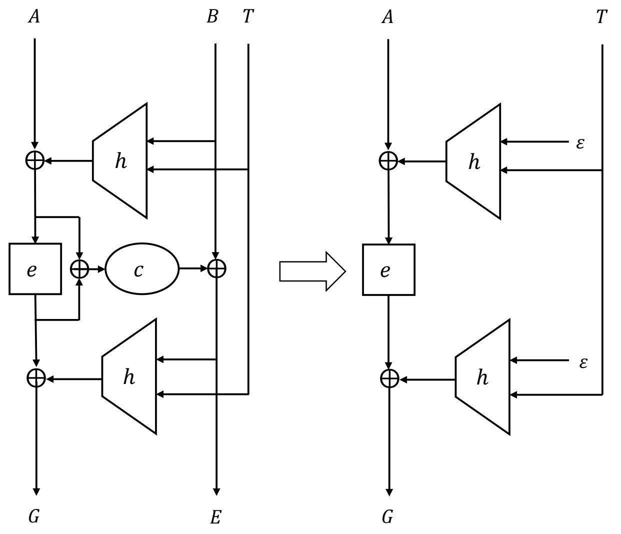


Fig. 9. HCTR and LRW2

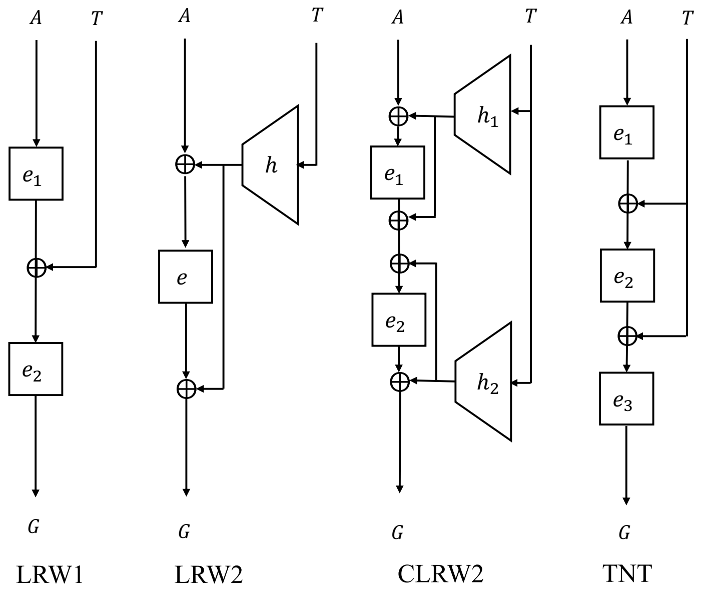


Fig. 10. LRW1, LRW2, CLRW2 and TNT

D Proof of Lemma 3

ivRND and nRND. Stream cipher c is a keyed variable-output-legth function. We define two security notions for stream ciphers, one is ivRND and the other is nRND. The ivRND security is defined through two games ivReal and ivRandom. When adversary $\mathcal{A}$ queries l, the game ivReal returns $S \| \text{msb}_l(c(S))$, where S is an n-bit random string, the game ivRandom returns an $(n+l)$-bit random string R. The ivRand advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_c^{\text{ivrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{ivReal} \Rightarrow 1] - \Pr[\mathcal{A}^{ivRandom} \Rightarrow 1]$.

The nRND security is defined through two games nReal and nRandom. When adversary $\mathcal{A}$ queries (S, l), the game nReal returns $\text{msb}_l(c(S))$, the game nRandom returns an l-bit random string R. The adversary $\mathcal{A}$ never repeats S. The nRand advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_c^{\text{nrnd}}(\mathcal{A}) = \Pr[\mathcal{A}^{nReal} \Rightarrow 1] - \Pr[\mathcal{A}^{nRandom} \Rightarrow 1]$.

In XCBv1 and XCBv2, $c(S) = e(S) \| e(incr(S)) \| e(incr^2(S)) \| \cdots$, where e is a block cipher and $incr(S) = S[0, n-33] \| (S[n-32, n-1] + 1 \mod 2^{32})$. It is easy to verify that c is ivRND secure but not nRND secure.

STRND. Let $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ be a tweakable enciphering mode, and let $\mathbf{D}$ be its inverse. The advantage of distinguishing $\mathbf{E}$ from random bits is

$$\mathbf{Adv}_{\mathbf{E}}^{\text{strnd}}(\mathcal{A}) = \Pr_{K \xleftarrow{\$} \mathcal{K}}[\mathcal{A}^{\mathbf{E}_K(\cdot,\cdot), \mathbf{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot), \$(\cdot,\cdot)} \Rightarrow 1]$$

where $\$(T, P)$ returns a random string of length $|P|$. We insist that $\mathcal{A}$ makes no pointless queries, regardless of oracle responses, and $\mathcal{A}$ asks no query (T, P) outside of $\mathcal{T} \times \mathcal{X}$. We extend the definition above in the usual way to its resource-bounded versions. We have the following.

Lemma 4. *[STPRP-security ≈ STRND-security [18]] Let $\mathbf{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ be a tweakable enciphering mode and let $q \geq 1$. Then for adversary $\mathcal{A}$ making q queries*

$$|\mathbf{Adv}_{\mathbf{E}}^{\text{stprp}}(\mathcal{A}) - \mathbf{Adv}_{\mathbf{E}}^{\text{strnd}}(\mathcal{A})| \leq \frac{q(q-1)}{2^{n+1}}$$

where n is the length of the shortest string in $\mathcal{X}$.

Proof of Lemma 3.

Proof. . The adversary $\mathcal{A}$ queries the encryption and decryption oracles of $\text{PIV}[\tilde{e}, \tilde{d}, c]$. In the following, we will replace the components in PIV with ideal components one-by-one, so that $\mathcal{A}$ interacts with two oracles in the subsequent games.

- Game G1: the encryption and decryption of $\text{PIV}[\tilde{e}, \tilde{d}, c]$.
- Game G2: the encryption and decryption of $\text{PIV}[\$, \tilde{d}, c]$.
- Game G3: the encryption and decryption of $\text{PIV}[\$, \$, c]$.
- Game G4: the encryption and decryption of $\text{PIV}[\$, \$, \$]$.
- Game G5: \$ and \$.

By regular reductions, there exist adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$, each making q queries, such that

$$\Pr[\mathcal{A}^{G1} \Rightarrow 1] - \Pr[\mathcal{A}^{G2} \Rightarrow 1] \leq \mathbf{Adv}_{\tilde{e}}^{\text{strnd}}(B),$$

$$\Pr[\mathcal{A}^{G2} \Rightarrow 1] - \Pr[\mathcal{A}^{G3} \Rightarrow 1] \leq \mathbf{Adv}_{\tilde{d}}^{\text{strnd}}(C),$$

$$\Pr[\mathcal{A}^{G3} \Rightarrow 1] - \Pr[\mathcal{A}^{G4} \Rightarrow 1] \leq \mathbf{Adv}_{c}^{\text{ivrnd}}(D).$$

G4 and G5 are identical, so we have

$$\mathbf{Adv}_{\text{PIV}[\tilde{e},\tilde{d},c]}^{\text{strnd}}(\mathcal{A}) \leq \mathbf{Adv}_{\tilde{e}}^{\text{strnd}}(\mathcal{B}) + \mathbf{Adv}_{\tilde{d}}^{\text{strnd}}(\mathcal{C}) + \mathbf{Adv}_{c}^{\text{ivrnd}}(\mathcal{D}).$$

By Lemma 4, we have

$$\mathbf{Adv}_{\text{PIV}[\tilde{e},\tilde{d},c]}^{\text{stprp}}(\mathcal{A}) \leq \mathbf{Adv}_{\tilde{e}}^{\text{stprp}}(\mathcal{B}) + \mathbf{Adv}_{\tilde{d}}^{\text{stprp}}(\mathcal{C}) + \mathbf{Adv}_{c}^{\text{ivrnd}}(\mathcal{D}) + \frac{3q^2}{2^{n+1}}.$$

□