Extended Analysis of Key Committing Security of HCTR2

Donghoon Chang[1,2], Yukihito Hiraga[3], Kazuhiko Minematsu[4,5], Nicky Mouha[6], Yusuke Naito[7], Yu Sasaki[1,8], Rentaro Shiba[7,9], and Takeshi Sugawara[3]

[1] Associate of National Institute of Standards and Technology
[2] FWI, Fairfax, USA
[3] The University of Electro-Communications, Tokyo, Japan
[4] NEC Corporation, Kawasaki, Japan
[5] The University of Osaka, Suita, Japan
[6] KeyCryptic, Bethesda, USA
[7] Mitsubishi Electric Corporation, Kanagawa, Japan
[8] NTT Social Informatics Laboratories, Tokyo, Japan
[9] Nagoya University, Nagoya, Japan

Abstract. HCTR2 is a tweakable enciphering mode that has been deployed in real applications, including Android file-based encryption and the Linux kernel. NIST has proposed variants of HCTR2 for its upcoming cryptographic accordion standards, while emphasizing the importance of additional security properties when an accordion is converted into authenticated encryptions (AE) via the encode-then-encipher (EtE) framework, including key-committing security and post-quantum security. Recently, Chen et al. and Chang et al. investigated the key-committing security of EtE-HCTR2, and we extend these results in three ways.
First, we show that the secret mask L in HCTR2, which was previously shown to be unnecessary for AE security, contributes to maintaining a certain level of key-committing security by removing the correlation between the block cipher calls without relying on the hash function used. To show it, we present a constant-time attack on EtE-HCTR2 without L.
Second, we present information-theoretic attacks that tighten the bounds of Chang et al. We reduce the exponent in the number of block-cipher calls by a logarithmic factor by exploiting the block-wise key stream generation of HCTR2. We provide experimental verification and show that the attack also slightly improves the computationally-bounded setting.
Third, we present quantum key-committing attacks. By applying quantum algorithms to the classical attacks of Chang et al., we obtain several quantum attacks that achieve exponential speedups in both qRAM and qRAM-free settings. In particular, for $\mathsf{EtE_A}$-HCTR2 with n-bit block cipher and τ-bit encoding, we obtain a qRAM-based attack with $O(2^{\tau/4})$ time and qRAM for $\tau \leq 4n/3$, and a qRAM-free attack with $O(2^{2\tau/7})$ time and $O(2^{\tau/7})$ classical RAM for $\tau \leq 7n/5$. For larger τ, we also obtain exponential speedups over the classical attacks of Chang et al. by using quantum multiple-collision search techniques.

Keywords: Accordion · HCTR2 · Encode-then-Encipher · Key-committing security · Information-theoretic (IT) attacks · Quantum attacks

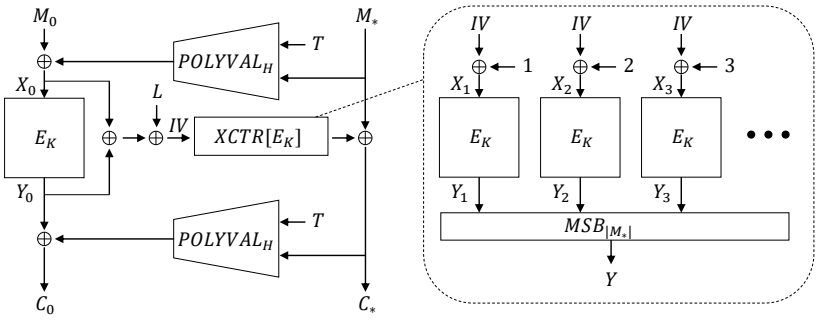

Fig. 1. Encryption of HCTR2; XCTR is shown in the dashed box.

1 Introduction

Cryptographic accordions [10], or simply accordions, are modes of operation for constructing variable-input-length strong tweakable pseudorandom permutations (VIL-STPRPs) from block ciphers (BCs). Since National Institute of Standards and Technology (NIST) launched its development project in 2023, they have attracted increasing attention from the symmetric-key community.

HCTR2 [13] is an accordion proposed by Crowley et al. as an improvement over its predecessor, HCTR [25]. HCTR2 follows the hash-encrypt-hash structure, as shown in Fig. 1, where a message of length $\geq n$ is divided into the first n bits M_0 and the remaining M_*. The construction is composed of a BC in XCTR mode and the polynomial hash function POLYVAL [18, 19]. Instantiated with AES, HCTR2 provides excellent performance thanks to hardware acceleration on modern processors. Developed by Google, HCTR2 is integrated into the Linux kernel [24] and is widely used for Android's file-based encryption [2]. Consequently, NIST recently proposed developing accordions based on HCTR2 [22].

The original designers proved that HCTR2 is secure up to $O(2^{n/2})$ queries when instantiated with an n-bit BC, achieving birthday-bound security. The original proof was in the single-user setting. More recently, Chen et al. [12] investigated the provable security of HCTR2 in the multi-user setting and proved that HCTR2 maintains the same level of security even against stronger adversaries that can send queries to multiple users with distinct keys. Interestingly, they showed that the provable security stays the same even without the secret mask L added to IV (see Fig. 1).

Robust authenticated encryption (RAE) [1,20] is an authenticated encryption (AE) notion that addresses a wide range of misuses by consolidating deterministic authenticated encryption (DAE) [23] and misuse-resistant AE (MRAE) [23]. The simple realization of RAE through the encode-then-encipher framework (EtE) [4] constitutes a key advantage of accordions. EtE works by encoding a message in a special format, typically zero padding, and verifying the encoding at the time of decryption for authenticity.

Another line of research studies the committing security, which was first defined for nonce-based AE [14]. This relatively new security model considers an adversary who is challenged to generate a ciphertext that can be decrypted into multiple messages using different secret keys, which is beyond the scope of conventional AE security. NIST specified the committing security for AE based on an accordion. Let $C = F(K, T, M)$ be the AE function using an accordion, where the ciphertext C is generated by a function F that takes as input a key K, a tweak T, and a message M. Then, the committing security is defined as the difficulty of finding $F(K, T, M)$ and $F(K', T', M')$ satisfying $F(K, T, M) = F(K', T', M')$. Key commitment requires $K \neq K'$, while context commitment requires $(K, T, M) \neq (K', T', M')$, and these align with the definitions for CMT-1 and CMT-4 for nonce-based AE in [3]. Given its importance in many real-world applications, NIST requires that AE applications of the accordion must support key commitment, and therefore, the key committing security of EtE-HCTR2 is crucially important. Moreover, Chang et al. [9] further defined two kinds of key committing security based on the notions proposed by Menda et al. [21]: CMT_k (requiring only $K = K'$) and $\mathsf{CMT}^*_\mathsf{k}$ (additionally requiring $T = T'$).

In EtE-HCTR2, the security level is parameterized by the bit length of the padded zeros, denoted as τ. The committing security also depends on an encoding method, i.e., where to add the 0^τ padding [9, 11]. Let M be a message. In the previous works, the following three encoding methods have been discussed:

EtE_A appends zeros to a message, i.e., the input to HCTR2 is $M\|0^\tau$,
EtE_P prepends zeros to a message, i.e., the input to HCTR2 is $0^\tau\|M$,
EtE_S adds zeros to the beginning of M_*, i.e., the input to HCTR2 is $M_0\|0^\tau\|M_*$.

A trivial attack to break CMT_k and $\mathsf{CMT}^*_\mathsf{k}$ security is a brute-force search, which first fixes C and T, and then searches for two keys satisfying 0^τ in the encoded message, requiring $O(2^\tau)$ decryptions. Then, Chen et al. [11] discovered new attacks that break EtE_P-HCTR2 and EtE_A-HCTR2 in $O(2^{\min\{n/2,\tau\}})$ and $O(2^{\tau/2})$, respectively. More recently, Chang et al. [9] improved the results, showing new attacks that break EtE_P-HCTR2 and EtE_A-HCTR2 in $O(2^{\tau/2})$ and $O(2^{\max\{\tau/3,\tau-n\}})$. Chang et al. [9] also pointed out that EtE_S-HCTR2 can also be attacked in $O(2^{\tau/2})$ and showed proofs that the adversary's advantages for EtE_P-HCTR2 and EtE_S-HCTR2 can be bounded by $\frac{n^2}{\log_2 n} \cdot \frac{p^2}{2^\tau}$ and $\frac{n}{\log_2 n} \cdot \frac{p^2}{2^\tau}$, respectively, where p is the number of BC calls in the ideal cipher model.

1.1 Challenges

The rigorous security evaluation of EtE-HCTR2 has become a significant research challenge. Several open research problems remain, as described below.

Impact of secret mask L on committing security. In proving AE-security of hash-then-hash constructions such as HCTR2, it is important to avoid collisions of the inputs between the BC for the leftmost block and the BC in XCTR. HCTR, the predecessor of HCTR2, did not use L. This design choice was based on

the observation that, when the hash key is secret, the output of the polynomial hash becomes random and unknown. However, the designers of HCTR2 showed that, when all inputs to the polynomial hash are zero, the randomness provided by the hash key cannot be used. With this analysis in mind, L was introduced for HCTR2. On the other hand, Chen et al. [12] showed that, in HCTR2's POLYVAL, the appropriate padding ensures that all inputs cannot be zero, and therefore, a proof of the same level of security is possible even without L. In other words, L does not contribute to improving AE security.

For the commitment, the situation is different because the key is known. Regarding the context-committing security, Chen et al. [11] showed that the attack succeeds trivially by finding a collision in POLYVAL under the same key with different tweaks; thus, L obviously has no impact. For the key-committing security, Chen et al.'s EtE_A attack does not depend on the presence or absence of L [11], whereas Chang et al.'s attack exploits the details of how L is generated [9]. At present, the impact of L on the key-committing security is unclear.

Security against Information-Theoretic (IT) Adversaries. For EtE_P, the attack and the proof [9] match up to a polynomial factor. However, the coefficient of the birthday-bound-security term in the proof is $\frac{n^2}{\log_2 n}$, which is relatively large: 2^{13} when $n = 256$ and $2^{11.2}$ when $n = 128$. Because the attack complexity is $2^{\frac{\tau}{2}+2}$, there remains a gap. Another source of this gap is that the proof considers only the number of BC calls, whereas the attack considers computationally bounded (CB) adversaries. There is room to get closer to the true tightness by considering IT adversaries that take into account only the number of BC calls.

Security against Quantum Adversaries. During the NIST standardization project of the accordion, it was stated explicitly that "*NIST is interested in both analysis and security proofs of the accordion in a quantum setting following the Q1 model [...]* [10, Sect. 2.2.5]." In committing security, adversaries can choose the keys; therefore, all computations can be done offline, which is exactly the setting NIST is interested in. So far, the security of HCTR2 against quantum adversaries has not been considered.

1.2 Our Contributions

Addressing the above challenges, this paper contains the following three results.

***O*(1) Attack on EtE-HCTR2 without *IV* Mask (Section 4).** We show that without L, the key commitment security of EtE_A-HCTR2 and EtE_S-HCTR2 can be broken instantly. This implies that L contributes to maintaining a certain level of security with respect to $\mathsf{CMT}^*_\mathsf{k}$ and CMT_k. Intuitively, to ensure that every BC output behaves randomly, avoiding collisions of the input between the BC for the leftmost block and one in XCTR is also effective for key commitment. Unlike in the secret-key setting, the adversary can freely control POLYVAL without

relying on the all-zero input. Therefore, a structure that weakens block-cipher correlations without relying on POLYVAL is needed, and L plays that role.

IT Attacks on EtE-HCTR2 (Section 5). This section presents IT attacks on EtE-HCTR2, where only the number of BC calls matters for the attack cost, and other costs, such as XOR, comparing two values to search for a match, memory accesses for stored items, etc., are not taken into account. Our main idea for reducing the number of BC calls in $\mathsf{EtE_P}$ is to exploit the fact that the key stream of XCTR is output block by block. This one-block output is commonly used even as the key stream length varies from 1 bit to n bits. In other words, after making a BC call, one can increase the number of collision sources by a factor of n, without making additional BC calls, by varying $|M_*|$ from 1 to n. This reduces the number of BC calls by a factor of $\log_2 n/2$, which tightens the previous security bounds. We show experimental verification with a small τ for the proof of concept. Moreover, with some precomputation, the computational cost of our IT attack for CB adversaries can be lower than that of the previous attack [9], so our attack slightly improves [9] even in the same setting.

We also show that, for $\mathsf{EtE_A}$ with $\tau = 2n$, the number of BC calls can be reduced by a factor of $n/2$ by considering all the positions of the colliding bits.

Quantum Attacks on EtE-HCTR2 (Section 6). We present quantum attacks, which improve the best classical attacks in the relevant parameter regimes.

Quantum random access memory (qRAM), which enables access to stored data in quantum superposition, is often assumed in quantum attacks. Given that the feasibility of large-scale qRAM remains unclear, we analyze quantum attacks under memory assumptions with and without qRAM.

All of our attacks build on Chang et al.'s classical attack and use quantum collision search as a subroutine, employing either the Brassard–Høyer–Tapp (BHT) algorithm [7] with qRAM or the Chailloux–Naya-Plasencia–Schrottenloher (CNS) algorithm [8] with classical RAM. For $\mathsf{EtE_P}$-HCTR2, this gives a direct quantum speedup. For $\mathsf{EtE_A}$-HCTR2, however, a direct quantum acceleration is not optimal. Our main idea is to introduce a tunable collision length into Chang et al.'s divide-and-conquer approach and optimize the resulting trade-off under quantum collision-search costs.

As a result, for $\mathsf{EtE_A}$-HCTR2, we obtain attacks with time complexity $O(2^{\tau/4})$ for $\tau \leq 4n/3$ using qRAM of size $O(2^{\tau/4})$, and time complexity $O(2^{2\tau/7})$ for $\tau \leq 7n/5$ using classical RAM of size $O(2^{\tau/7})$. For $\mathsf{EtE_P}$-HCTR2, by directly applying quantum collision search to the collision-finding step of Chang et al.'s attack, we obtain attacks with time complexity $O(2^{\tau/3})$ using qRAM of size $O(2^{\tau/3})$, and time complexity $O(2^{2\tau/5})$ using classical RAM of size $O(2^{\tau/5})$.

For $\mathsf{EtE_A}$-HCTR2 beyond the above ranges, we further apply multiple-collision search in the quantum setting to Chang et al.'s attack, obtaining exponential speedups over their classical attack in the large-τ regime.

The detailed complexities, memory requirements, and parameter ranges are summarized in Table 1.

Table 1. Summary of our quantum attacks on EtE-HCTR2.

Target	Memory Model	Time	Memory	Range of τ	Ref.
EtE_A-HCTR2	qRAM	$2^{\tau/4}$	$2^{\tau/4}$	$\tau \leq 4n/3$	Sect. 6.3
	classical RAM	$2^{2\tau/7}$	$2^{\tau/7}$	$\tau \leq 7n/5$	Sect. 6.3
	qRAM*	$2^{(2\tau-n)/5}$	$2^{(2\tau-n)/5}$	$4n/3 \leq \tau < 2n$	Sect. 6.4
	classical RAM†	$2^{2(2\tau-n)/9}$	$2^{(2\tau-n)/9}$	$7n/5 \leq \tau < 2n$	Sect. 6.4
EtE_P-HCTR2	qRAM	$2^{\tau/3}$	$2^{\tau/3}$	$\tau \leq n$	Sect. 6.5
	classical RAM	$2^{2\tau/5}$	$2^{\tau/5}$		

*: Valid when $k \geq (\tau + 2n)/5$, where k is the key length of the block cipher. Using Bonnetain et al.'s quantum walk-based algorithm [5], the same complexity can also be obtained in certain shorter-key regimes, namely $(3\tau + n)/10 \leq k \leq n$, at the cost of assuming a stronger qRAM model. The details of qRAM models are given in Sect. 6.1.

†: Valid when $k \geq (\tau + n)/3$.

1.3 Paper Outline

The rest of the paper is organized as follows. The paper begins with basic notations and HCTR2's specification in Sect. 2, and we briefly recall the state-of-the-art attack on EtE-HCTR2 in Sect. 3. Then, Sect. 4 describes a new $O(1)$ attack on HCTR2 without IV mask, followed by our new IT $\mathsf{CMT}^*_\mathsf{k}$ attacks on EtE-HCTR2 in Sect. 5. The quantum attacks on EtE-HCTR2 are presented in Sect. 6, and Sect. 7 concludes the paper.

2 Preliminaries

2.1 Notation

Let ε denote the empty string and $\emptyset$ denote the empty set. For an integer $i \geq 0$, let $\{0,1\}^i$ be the set of all i-bit strings and $\{0,1\}^0 \coloneqq \{\varepsilon\}$. Let $\{0,1\}^{\geq i}$ be the set of all bit strings of length $\geq i$. For a bit string X and an integer i, let $\mathsf{msb}_i(X)$ and $\mathsf{lsb}_i(X)$ be the most and least significant i bits of X, respectively. For integers $0 \leq i \leq j$, let $[i,j] \coloneqq \{i, i+1, \ldots, j\}$, and $[j] \coloneqq [1,j]$. If $j < i$ then $[i,j] \coloneqq \emptyset$. Let ozp (resp. zp) be a padding function that accepts an arbitrary-length bit-string X and returns $X\|10^s$ (resp. $X\|0^s$) where s is the smallest non-negative integer such that $|X\|10^s| \bmod n = 0$ (resp. $|X\|0^s| \bmod n = 0$). For a non-negative integer i, let $\mathtt{bin}_n(i)$ be the little-endian conversion of i to an n-bit string. For a value or set X, the notation $Y \leftarrow X$ means that X is assigned to Y. For a positive integer n and a bit string D of length $\geq n$, $(D_1, D_2) \xleftarrow{n,|D|-n} D$ means that D is partitioned into the n-bit string D_1 and the $(|D|-n)$-bit string D_2 such that $D = D_1\|D_2$, and $(D_1, \ldots, D_\ell) \xleftarrow{n} D$ means that D is partitioned into the ℓ strings $D_1, \ldots, D_\ell$ such that $|D_i| = n$ for $i = 1, \ldots, \ell-1$, $|D_\ell| \leq n$, and $D = D_1\| \cdots \|D_\ell$.

Algorithm 1 Encryption of HCTR2

Main $\mathsf{HCTR2}[E_K](T, M)$

1: $H \leftarrow E_K(\mathtt{bin}_n(0))$; $L \leftarrow E_K(\mathtt{bin}_n(1))$;
2: $(M_0, M_*) \xleftarrow{n,|M|-n} M$
3: $Z_1 \leftarrow \mathsf{POLYVAL}_H(T, M_*)$; $X_0 \leftarrow Z_1 \oplus M_0$; $Y_0 \leftarrow E_K(X_0)$
4: $IV \leftarrow X_0 \oplus Y_0 \oplus L$; $C_* \leftarrow \mathsf{XCTR}[E_K](IV, |M_*|) \oplus M_*$;
5: $Z_2 \leftarrow \mathsf{POLYVAL}_H(T, C_*)$; $C_0 \leftarrow Z_2 \oplus Y_0$
6: $C \leftarrow C_0 \| C_*$; **return** C

Subroutine $\mathsf{XCTR}[E_K](IV, |D|)$

1: **for** $i = 1, \ldots, \lceil |D|/n \rceil$ **do** $Y_i \leftarrow E_K(IV \oplus \mathtt{bin}_n(i))$ **end for**
2: $Y \leftarrow \mathsf{msb}_{|D|}(Y_1 \| \cdots \| Y_{\lceil |D|/n \rceil})$; **return** Y

Subroutine $\mathsf{POLYVAL}_H(T, D)$

1: **if** n **divides** $|D|$ **then** $Z \leftarrow \overline{\mathsf{poly}}_H(\mathtt{bin}_n(2|T|+2) \| \mathtt{zp}(T) \| D)$
else $Z \leftarrow \overline{\mathsf{poly}}_H(\mathtt{bin}_n(2|T|+3) \| \mathtt{zp}(T) \| \mathtt{ozp}(D))$ **end if**
2: **return** Z

Subroutine $\overline{\mathsf{poly}}_H(\tilde{D})$

1: $D_1, \ldots, D_\ell \xleftarrow{n} \tilde{D}$; $\overline{H} \leftarrow H \cdot \mathtt{x}^{-n}$
2: $Z \leftarrow D_1 \cdot \overline{H}^{\ell} \oplus D_2 \cdot \overline{H}^{\ell-1} \oplus \cdots \oplus D_{\ell-1} \cdot \overline{H}^{2} \oplus D_\ell \cdot \overline{H}$; **return** Z

2.2 Specification of HCTR2

The specification of HCTR2 [13] is shown in Alg. 1. HCTR2 is an accordion with a hash-encrypt-hash structure, where the hashing parts use the polynomial hash function POLYVAL, and the encryption part uses XCTR. $\mathsf{POLYVAL}_H$ is a hash function with a key $H \in \{0,1\}^n$ that takes a tweak $T \in \{0,1\}^*$ and a plaintext or ciphertext in $\{0,1\}^*$ and returns an n-bit output. The hash key is defined as $H = E_K(\mathtt{bin}_n(0))$. $\mathsf{XCTR}[E_K]$ is a mode of stream encryption that takes an n-bit IV and a data length and returns a key stream of the specified length.

$\mathsf{HCTR2}[E_K] : \{0,1\}^* \times \{0,1\}^{\geq n} \to \{0,1\}^{\geq n}$ denotes the HCTR2 encryption, which takes a tweak $T \in \{0,1\}^*$ and a plaintext $M \in \{0,1\}^{\geq n}$, and returns a ciphertext C such that $|M| = |C|$. For the encryption operation, a plaintext M is divided into an n-bit left part M_0 and the remaining right part M_*. In the first hashing part, $\mathsf{POLYVAL}_H$ is evaluated on T and M_* to produce the output Z_1. In the encryption part, the value $X_0 = M_0 \oplus Z_1$ is encrypted by the BC E_K to produce Y_0. The value $IV = X_0 \oplus Y_0 \oplus L$ is then used as the IV for XCTR. The right part of the plaintext M_* is encrypted using XCTR with $IV = X_0 \oplus Y_0 \oplus L$, yielding the right part of the ciphertext C_*. In the second hashing part, $\mathsf{POLYVAL}_H$ is evaluated on T and C_* to produce the output Z_2, and the left n-bit part of the ciphertext C_0 is obtained by XORing Z_2 with Y_0. The ciphertext is defined as $C = C_0 \| C_*$. The decryption of HCTR2 can be defined in the natural way, so we omit it.

POLYVAL is a polynomial hash function that accepts $T \in \{0,1\}^*$ and $D \in \{0,1\}^*$ and returns an n-bit value. The multiplications are done over the Galois

field $GF(2^n)$ with an irreducible polynomial $p(\mathtt{x})$. The irreducible polynomial of HCTR2 is defined as $p(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^{127} + \mathtt{x}^{126} + \mathtt{x}^{121} + 1$. In the multiplications, an n-bit string $a = a_0 \| a_1 \| \cdots \| a_{n-2} \| a_{n-1} \in \{0,1\}^n$ is regarded as an element in $GF(2^n)$: $a_{n-1}\mathtt{x}^{n-1} + a_{n-2}\mathtt{x}^{n-2} + \cdots + a_1\mathtt{x} + a_0$, where $\forall i \in [0, n-1] : a_i \in \{0,1\}$.

3 Previous Attacks on EtE-HCTR2

3.1 Inverting POLYVAL

Given a hash key H, input data D, and an output value Z, it is easy to find a tweak T such that $Z = \mathsf{POLYVAL}_H(T, D)$ because of the algebraic structure of polynomial hash functions. The analysis for the case where both $|T|$ and $|D|$ are multiples of n is as follows. The extension to the other case is straightforward.

$$\begin{aligned}
\mathsf{POLYVAL}_H(T, D) = Z &\Leftrightarrow \overline{\mathsf{poly}}_H(\mathtt{bin}_n(2|T|+2) \| T \| D) = Z, \\
&\Leftrightarrow (2|T|+2) \cdot \overline{H}^{\frac{|T|+|D|}{n}+1} \oplus \overline{\mathsf{poly}}_H(T) \cdot \overline{H}^{\frac{|D|}{n}} \oplus \overline{\mathsf{poly}}_H(D) = Z, \\
&\Leftrightarrow \overline{\mathsf{poly}}_H(T) = \left(Z \oplus (2|T|+2) \cdot \overline{H}^{\frac{|T|+|D|}{n}+1} \oplus \overline{\mathsf{poly}}_H(D)\right) \cdot \overline{H}^{-\frac{|D|}{n}},
\end{aligned}$$

where the right-hand side is reduced to a single value. One can set all but one block of T arbitrarily and compute the required value for the remaining block using simple algebra. Moreover, the analysis can be extended to find a common tweak $T = T'$ such that $\mathsf{POLYVAL}_H(T, D) = Z$ and $\mathsf{POLYVAL}_{H'}(T', D') = Z'$ are simultaneously satisfied for given values H, D, Z, H', D', and Z'. For this, the adversary needs to solve a system of two linear equations, and a solution can be found if at least two blocks of $T (= T')$ can be controlled.

3.2 Previous Attacks on EtE$_\mathsf{A}$-HCTR2

The core part of the attack involves generating a collision on the least significant τ bits of the key stream by choosing two distinct values X_0 and X_0'. The attack by Chen et al. [11], depicted in Fig. 2, follows this approach. It first fixes K and K', and then searches for X_0 and X_0' that collide on the last τ bits of the key stream, i.e., $\mathsf{lsb}_\tau(Y) = \mathsf{lsb}_\tau(Y')$, by the birthday paradox, with a cost of $O(2^{\tau/2})$ computations. Once such a collision is found, the rest of the attack is straightforward. The last τ bits of M_* are fixed to 0; hence, the last τ bits of C_* are $0 \oplus \mathsf{lsb}_\tau(Y) = \mathsf{lsb}_\tau(Y)$. The remaining bits of the ciphertext are set arbitrarily. The attack computes values of T that satisfy both $E_K(X_0) \oplus C_0 = \mathsf{POLYVAL}_H(T, C_*)$ and $E_{K'}(X_0') \oplus C_0 = \mathsf{POLYVAL}_{H'}(T, C_*)$ by inverting the second hash layer. Finally, M and M' are generated by decrypting C and C' under K and K', respectively.

Chang et al. [9] generate a key collision with a complexity of $O(2^{\tau/3})$ for $\tau \leq 3n/2$ using the divide-and-conquer approach. The key observation of their attack is that "When the output value of the BC on the left-hand side is $L \oplus \mathtt{bin}_n(i)$, the key-stream value of the i-th block of the XCTR mode is $L \oplus \mathtt{bin}_n(i)$."

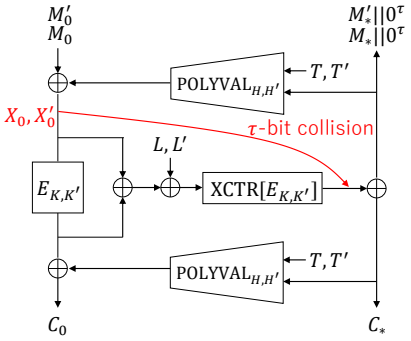

Fig. 2. Previous attacks on $\mathsf{EtE_A}$-HCTR2 [9, Fig. 5].

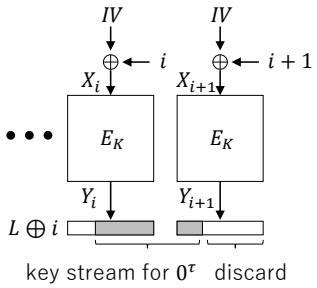

Fig. 3. Setup for divide-and-conquer.

Chang et al. choose the message length so that, among τ bits of zeros for the zero-padding, $2\tau/3$ bits are positioned in block i and $\tau/3$ bits are positioned in block $i+1$ as illustrated in Fig. 3. The attack first finds L and L' satisfying $\mathsf{lsb}_{2\tau/3}(L) = \mathsf{lsb}_{2\tau/3}(L')$. This ensures the collision in the least significant $2\tau/3$ bits of the i-th key-stream block, i.e., $\mathsf{lsb}_{2\tau/3}(L \oplus \mathtt{bin}_n(i)) = \mathsf{lsb}_{2\tau/3}(L' \oplus \mathtt{bin}_n(i))$ holds for any i. Therefore, it remains to find i such that the most significant $\tau/3$ bits of the $(i+1)$-th key-stream block also collide. Once such (L, L') and i are found, the remaining is the same as that of Chang et al. [9]. In summary, the attack consists of three phases, each requiring $O(2^{\tau/3})$ complexity.

Phase 1. Searching for a pair (K, K') colliding on the last $2\tau/3$ bits of (L, L').
Phase 2. Searching for i that derives a collision on $\mathsf{msb}_{\tau/3}(Y_{i+1})$.
Phase 3. Generating an input pair with a length of $O(2^{\tau/3})$ blocks.

The maximum collision size in Phase 1 is n bits. When $\tau > 3n/2$, $2\tau/3$ exceeds n; thus, the $O(2^{\tau/3})$ attack no longer works. The attacker then searches for an n-bit collision in Phase 1, and searches for an i that results in a $(\tau - n)$-bit collision in the $(i+1)$-th block in Phase 2, with an attack cost of $O(2^{\tau-n})$.

3.3 Previous Attacks on $\mathsf{EtE_P}$-HCTR2

The attacker can trivially ensure that C_* and C'_* collide by properly choosing M_* and M'_*. Hence, the most challenging part is to generate a collision between C_0 and C'_0 while preserving the encoded 0^τ in both M_0 and M'_0.

The attack by Chang et al. [9] is depicted in Fig. 4. They observe that the essence of the attack is to find a collision on a function that maps X_0 to a τ-bit value under two distinct keys. Specifically, they define the following function.

$$\mathsf{msb}_\tau\Big(X_0 \oplus E_K(X_0) \oplus \overline{\mathsf{poly}}_H\big(E_K(X_0 \oplus E_K(X_0) \oplus L \oplus \mathtt{bin}_n(1))\big)\Big) \qquad (1)$$

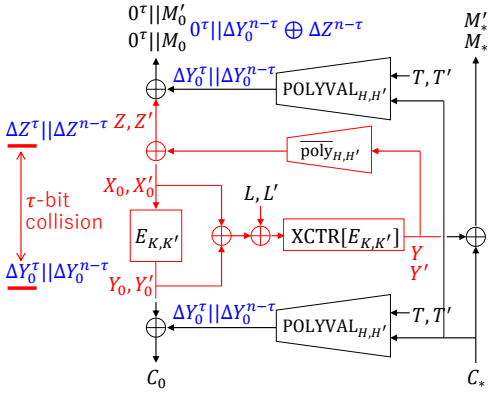

Fig. 4. Previous attack for $\mathsf{EtE_P}$-HCTR2 [9, Fig. 9].

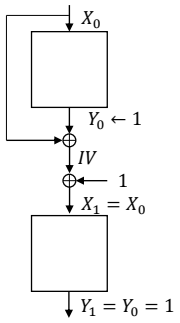

Fig. 5. Fixing the key stream for HCTR2 without L.

Let z denote the value of $X_0 \oplus \overline{\mathsf{poly}}_H\big(E_K(X_0 \oplus E_K(X_0) \oplus L \oplus \mathtt{bin}_n(1))\big)$, and recall that $Y_0 = E_K(X_0)$. Equation (1) is the function mapping (X_0, K) to $\mathsf{msb}_\tau(z \oplus Y_0)$. If (X_0, K) and (X_0', K') collide on this function, then $\mathsf{msb}_\tau(z \oplus Y_0) = \mathsf{msb}_\tau(z' \oplus Y_0')$; hence, $\mathsf{msb}_\tau(z \oplus z') = \mathsf{msb}_\tau(Y_0 \oplus Y_0')$, i.e., $\mathsf{msb}_\tau(\Delta z) = \mathsf{msb}_\tau(\Delta Y_0)$. After that, $\mathsf{msb}_\tau(M_0)$ and $\mathsf{msb}_\tau(M_0')$ are fixed to 0^τ, $\mathsf{lsb}_{n-\tau}(M_0)$ is set arbitrarily, and $\mathsf{lsb}_{n-\tau}(M_0')$ is computed as $M_0 \oplus \mathsf{lsb}_{n-\tau}(\Delta z \oplus \Delta Y_0)$. Then T is computed by inverting the first hash layer. This ensures that the output difference between $\mathsf{POLYVAL}_H(T, C_*)$ and $\mathsf{POLYVAL}_{H'}(T, C_*)$ is $\mathsf{msb}_\tau(\Delta z) \| \mathsf{lsb}_{n-\tau}(\Delta Y_0)$, which equals $\mathsf{msb}_\tau(\Delta Y_0) \| \mathsf{lsb}_{n-\tau}(\Delta Y_0)$ because of the collision. Hence, the differences cancel in the second hash layer, ensuring $\Delta C_0 = 0$.

4 $O(1)$ Attack on EtE-HCTR2 without IV Mask

In this section, we show a constant-time attack that breaks $\mathsf{CMT}_\mathsf{k}^*$ and CMT_k security when L is removed from EtE-HCTR2. This implies that L contributes to maintaining a certain level of security with respect to $\mathsf{CMT}_\mathsf{k}^*$ and CMT_k.

Core Idea. The attack breaks the $\mathsf{CMT}_\mathsf{k}^*$ security of both $\mathsf{EtE_A}$-HCTR2 and $\mathsf{EtE_S}$-HCTR2 for any target zero-padding length τ satisfying $\tau \le n$. The attacker chooses the message length $n \le |M| \le 2n$ so that $|M| + \tau = 2n$. Namely, M_0 satisfies HCTR2's minimum message-length requirement, and M_* is a one-block string consisting of $|M| - n$ message bits concatenated with τ zero-padding bits. In this setting, $\mathsf{EtE_A}$-HCTR2 and $\mathsf{EtE_S}$-HCTR2 differ only slightly: 0^τ is placed in the lsb of M_* in $\mathsf{EtE_A}$-HCTR2, while it is placed in the msb of M_* in $\mathsf{EtE_S}$-HCTR2. In what follows, the attack is described in a way that applies to both settings.

Suppose that the key is fixed to some value K. The attacker first fixes the value of Y_0 to $\mathtt{bin}_n(1)$, and computes the corresponding X_0 by $X_0 \leftarrow E_K^{-1}(Y_0)$. Because L has been removed, the IV for XCTR is computed as $X_0 \oplus Y_0$, which

Algorithm 2 $O(1)$ $\mathsf{CMT}^*_{\mathsf{k}}$ Attack against $\mathsf{EtE_A}$-HCTR2 without L

Require: τ satisfying $\tau \leq n$
Ensure: K, K', T, M_0, M_0' s.t. $|M| = 2n$, $|T| = 2n$ and the ciphertexts for $(K, T, M_0 \| 0^n)$ and $(K', T, M_0' \| 0^n)$ are colliding
1: Fix $K, K' \in \{0,1\}^k$ to any distinct values and compute the corresponding H, H'
2: $X_0 \leftarrow E_K^{-1}(\mathtt{bin}_n(1))$; $X_0' \leftarrow E_{K'}^{-1}(\mathtt{bin}_n(1))$, which ensures $Y_1 = Y_1' = \mathtt{bin}_n(1)$
3: Fix $C_0 \in \{0,1\}^n$ to arbitrary values and set $C_* = \mathtt{bin}_n(1)$, which ensures $M_* = M_*' = Y_1 \oplus C_* = \mathtt{bin}_n(1) \oplus \mathtt{bin}_n(1) = 0^n$
4: Find T s.t. $\mathsf{POLYVAL}_H(T, C_*) = \mathsf{POLYVAL}_{H'}(T, C_*) = C_0 \oplus \mathtt{bin}_n(1)$ by inverting POLYVAL.
5: $M_0 \leftarrow \mathsf{POLYVAL}_H(T, M_*) \oplus X_0$ and $M_0' \leftarrow \mathsf{POLYVAL}_{H'}(T, M_*) \oplus X_0'$
6: **return** K, K', T, M_0, M_0'

is $X_0 \oplus \mathtt{bin}_n(1)$. Because $|M_*| = n$, the XCTR mode generates only a single block of key stream. Inside XCTR, the counter value $\mathtt{bin}_n(1)$ is XORed with the IV, so the BC input becomes X_0. Because $E_K(X_0) = Y_0 = \mathtt{bin}_n(1)$, we have $Y_1 = \mathtt{bin}_n(1)$. A schematic diagram is depicted in Fig. 5.

The event $Y_1 = \mathtt{bin}_n(1)$ is key-independent. Hence, the attacker chooses another key value K' and fixes Y_0 to $\mathtt{bin}_n(1)$, which results in $Y_1 = \mathtt{bin}_n(1)$. This implies that the attacker finds a pair (K, Y_0) and (K', Y_0') that generate colliding key streams. Hence, this can easily be extended to $(K, T, M_0 \| M_*)$ and $(K', T', M_0' \| M_*')$ with τ bits of zeros in M_*, M_*' by following the previous attack framework for $\mathsf{EtE_A}$-HCTR2 in Sect. 3.2, which is detailed below.

Remaining Procedure and Pseudocode. First, the $n-\tau$ message bits in M_* and M_*' are fixed to zeros so that all n bits of M_* and M_*' are zero. Since $C_* = Y_1 \oplus M_*$ and Y_1 is fixed to $\mathtt{bin}_n(1)$, we set $C_* = \mathtt{bin}_n(1)$. C_0 can be fixed arbitrarily. Then, the tweaks T and T' are computed by inverting POLYVAL in the second hash layer, i.e., they are chosen to satisfy $(C_0 \oplus Y_0) = \mathsf{POLYVAL}_H(T, C_*) = \mathsf{POLYVAL}_{H'}(T', C_*)$. For $\mathsf{CMT_k}$ security, $T \neq T'$ is allowed; thus, both T and T' can be one block long. For $\mathsf{CMT}^*_{\mathsf{k}}$ security, $T = T'$ must be satisfied; thus, both T and T' must be at least two blocks long. Finally, M_0 and M_0' are computed by $M_0 = \mathsf{POLYVAL}_H(T, M_*) \oplus X_0$ and $M_0' = \mathsf{POLYVAL}_{H'}(T', M_*) \oplus X_0'$. Algorithm 2 gives pseudocode for the $\mathsf{CMT}^*_{\mathsf{k}}$ attack on $\mathsf{EtE_A}$-HCTR2 without L.

Remarks. The essence of the attack is that the adversary can create correlated inputs to the BC on the first block and a BC within XCTR. In the secret-key setting, such an event could occur only by the all-zero input to POLYVAL. In the commitment setting, however, the adversary can easily control the POLYVAL output. Therefore, the $O(1)$ attack is invalidated by having any construction, not only one using L, that eliminates BCs' correlation without relying on POLYVAL.

5 Information-Theoretic $\mathsf{CMT}^*_{\mathsf{k}}$ Attacks on EtE-HCTR2

This section presents IT attacks on EtE-HCTR2, where only the number of BC calls matters for the attack cost, and other costs, such as XOR operations, comparisons between two values when searching for a match, memory accesses to stored items, etc., are not taken into account. This setting matches the one analyzed in the proof by Chang et al. [9], and reducing the number of BC calls required for the attack narrows the gap between the bounds and improves our understanding of the construction's security. In Sect. 5.1, we describe an IT attack on EtE_P-HCTR2 and provide experimental verification for a small τ for the proof of concept. This attack also reduces the cost for CB adversaries, depending on the cost-counting model. In Sect. 5.2, we describe an IT attack on EtE_A-HCTR2 with $\tau = 2n$.

5.1 Attacks on EtE_P-HCTR2

The overall attack framework is the same as that of Chang et al. [9] in Sect. 3.3, which searches for (X_0, K) and (X_0', K') colliding on the τ-bit function specified by Eq. (1). In their framework, K and K' are fixed at the beginning. Then,

1. For $2^{\tau/2}$ choices of X_0, Eq. (1) is computed, and the results are stored.
2. For $2^{\tau/2}$ choices of X_0', Eq. (1) is computed, and the results are stored.

Finally, the colliding pair is selected. Equation (1) requires two BC calls per execution: one to compute $E_K(X_0)$ and the other to compute $E_K(X_0 \oplus E_K(X_0) \oplus L \oplus \mathtt{bin}_n(1))$. Hence, the total number of BC calls in this attack is $2^{\frac{\tau}{2}+2}$, because each of the above two steps requires $2 \times 2^{\tau/2}$ BC calls.

The core idea of this IT attack is that, for the one-block key stream generated with two BC calls, i.e., $Y_1 = E_K\big(X_0 \oplus E_K(X_0) \oplus L \oplus \mathtt{bin}_n(1)\big)$, the attacker can consider n options by varying $|M_*|$ from 1 to n without further increasing the number of BC calls. For each of the n choices of $|M_*|$, $\overline{\mathsf{poly}}_H$ is computed, and the birthday attack is used to find a match. The idea is depicted in Fig. 6.

Note that the $\overline{\mathsf{poly}}_H$ function accepts only inputs that are a multiple of n in length. Hence, if a one-block key stream is truncated, it must be properly padded to be processed by $\overline{\mathsf{poly}}_H$. Remember that the first hash layer is $\mathsf{POLYVAL}_H(T, C_* \oplus Y_1)$, where neither $|C_*|$ nor $|Y_1|$ is a multiple of n. Then $\mathsf{POLYVAL}_H(T, C_* \oplus Y_1)$ is transformed as

$$\begin{aligned}
\mathsf{POLYVAL}_H(T, C_* \oplus Y_1) &= \overline{\mathsf{poly}}_H(\mathtt{bin}_n(2|T|+3) \| \mathsf{zp}(T) \| \mathtt{ozp}(C_* \oplus Y_1)), \\
&= \overline{\mathsf{poly}}_H(\mathtt{bin}_n(2|T|+3) \| \mathsf{zp}(T) \| \mathtt{ozp}(C_*) \oplus \mathsf{zp}(Y_1)), \\
&= \mathsf{POLYVAL}_H(T, C_*) \oplus \overline{\mathsf{poly}}_H(\mathsf{zp}(Y_1)).
\end{aligned}$$

Hence, we apply zp when the one-block key stream Y_1 is truncated.

More specifically, we replace the two steps in the previous attack with the following procedures. Here, we set the loop number to $2^{(\tau - \log_2 n)/2}$ for simplicity, but we will later modify it to $2^{(\tau - \log_2 n + 1)/2}$ to consider the average number of distinct values of $\mathsf{zp}(\mathsf{msb}_\ell(Y_1))$ for $\ell = 1, \ldots, n$.

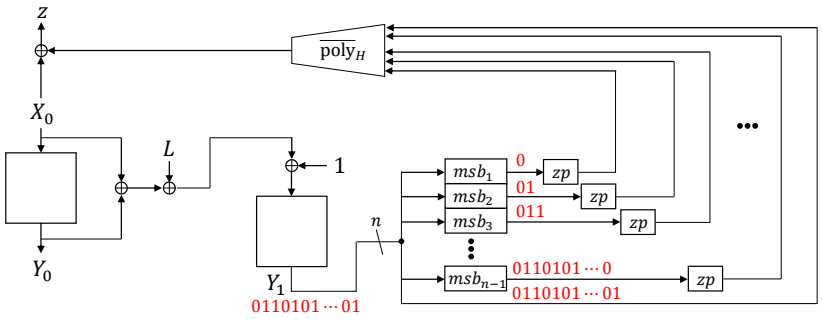

Fig. 6. Increasing Collision Sources without Additional BC Calls.

1. For $2^{(\tau-\log_2 n)/2}$ choices of X_0, $Y_1 = E_K\big(X_0 \oplus E_K(X_0) \oplus L \oplus \mathtt{bin}_n(1)\big)$ is computed, and for $\ell = 1, \ldots, n$, $|M_*|$ is set to ℓ, and $\mathsf{msb}_\tau\Big(X_0 \oplus Y_0 \oplus \overline{\mathsf{poly}}_H\big(\mathsf{zp}(\mathsf{msb}_\ell(Y_1))\big)\Big)$ is computed, and the results are stored.
2. For $2^{(\tau-\log_2 n)/2}$ choices of X_0', $Y_1' = E_{K'}\big(X_0' \oplus E_{K'}(X_0') \oplus L' \oplus \mathtt{bin}_n(1)\big)$ is computed, and for $\ell = 1, \ldots, n$, $|M_*'|$ is set to ℓ, and $\mathsf{msb}_\tau\Big(X_0' \oplus Y_0' \oplus \overline{\mathsf{poly}}_{H'}\big(\mathsf{zp}(\mathsf{msb}_\ell(Y_1'))\big)\Big)$ is computed, and the results are stored.

In key-committing attacks, two ciphertexts must satisfy $|C_*| = |C_*'|$, so the collision can be searched for within the same message length. For each $|M_*|$, we have $2^{(\tau-\log_2 n)} (= 2^{(\tau-\log_2 n)/2} \times 2^{(\tau-\log_2 n)/2})$ pairs to match. So for all the n choices of $|M_*|$, we have $n \cdot 2^{(\tau-\log_2 n)} = 2^\tau$ pairs to match. Algorithm 3 gives pseudocode of the entire attack procedure.

Strictly speaking, $\mathsf{zp}(\mathsf{msb}_\ell(Y_1)) \neq \mathsf{zp}(\mathsf{msb}_{\ell+1}(Y_1))$ does not always hold. This holds only when the $(\ell+1)$-th bit of Y_1 is one. Thus, the number of distinct values of $\mathsf{zp}(\mathsf{msb}_\ell(Y_1))$ is equal to the number of ones in its binary representation. In this attack, the above two steps are repeatedly performed for various choices of X_0. The average number of ones in the n-bit value of Y_1 is $n/2$. Hence, the number of iterations of the `for` loops in the above two steps must be $2^{(\tau-\log_2 n+1)/2}$ so that 2^τ pairs are collected after examining n choices of ℓ.

Both `for` loops starting from lines 4 and 12 each require two BC calls in each loop; hence, the total number of BC calls in this attack is $2^{\frac{\tau-\log_2 n}{2}+\frac{5}{2}}$. This reduces the number of BC calls of the attack in [9] by a factor of $2^{\frac{\log_2 n}{2}-\frac{1}{2}}$.

Note that this attack is an extension of the previous attack in [9]. In fact, by choosing $2^{\tau/2}$ values in lines 4 and 12, and by fixing $\ell = n$ for the loops from lines 6 – 10 and 14 – 18 in Alg. 3, the attack becomes the same as that in [9].

Numerical Example. Let $n = 256$ and $\tau = 64$. The number of BC calls in the previous attack [9] is $2^{\frac{\tau}{2}+2}$, which is 2^{34}. In contrast, the number of BC calls in

Algorithm 3 $O(2^{(\tau-\log_2 n)/2})$ IT attack against $\mathsf{EtE_P}$-HCTR2 with $\tau \leq n$

Require: τ satisfying $\tau \leq n$
Ensure: (K, T, M), (K', T, M')
s.t. $\mathsf{HCTR2}[E_K](T, 0^\tau \| M) = \mathsf{HCTR2}[E_{K'}](T, 0^\tau \| M')$

Set $|T| = 2n$
Fix $K, K' \in \{0,1\}^k$, $M_0 \in \{0,1\}^{n-\tau}$ arbitrarily
$H \leftarrow E_K(\mathtt{bin}_n(0))$; $L \leftarrow E_K(\mathtt{bin}_n(1))$; $H' \leftarrow E_{K'}(\mathtt{bin}_n(0))$; $L' \leftarrow E_{K'}(\mathtt{bin}_n(1))$
for $i = 1, \ldots, 2^{(\tau-\log_2 n+1)/2}$ **do**
    $X_0^{(i)} \xleftarrow{\$} \{0,1\}^n$; $Y_0^{(i)} \leftarrow E_K(X_0^{(i)})$; $Y_1^{(i)} \leftarrow E_K(X_0^{(i)} \oplus Y_0^{(i)} \oplus L \oplus \mathtt{bin}_n(1))$
    for $\ell = 1, \ldots, n$ **do**
        $Y_1^{(i),\ell} \leftarrow \mathsf{msb}_\ell(Y_1^{(i)})$
        $Z^{(i),\ell} \leftarrow X_0^{(i)} \oplus \overline{\mathsf{poly}}_H(\mathsf{zp}(Y_1^{(i),\ell}))$
        Compute $\mathsf{msb}_\tau(Y_0^{(i)} \oplus Z^{(i),\ell})$ and store it along with $Y_0^{(i)}$, $Z^{(i),\ell}$, $Y_1^{(i),\ell}$
    end for
end for
for $j = 1, \ldots, 2^{(\tau-\log_2 n+1)/2}$ **do**
    $X_0'^{(j)} \xleftarrow{\$} \{0,1\}^n$; $Y_0'^{(j)} \leftarrow E_{K'}(X_0'^{(j)})$; $Y_1'^{(j)} \leftarrow E_{K'}(X_0'^{(j)} \oplus Y_0'^{(j)} \oplus L' \oplus \mathtt{bin}_n(1))$
    for $\ell = 1, \ldots, n$ **do**
        $Y_1'^{(j),\ell} \leftarrow \mathsf{msb}_\ell(Y_1'^{(j)})$
        $Z'^{(j),\ell} \leftarrow X_0'^{(j)} \oplus \overline{\mathsf{poly}}_{H'}(\mathsf{zp}(Y_1'^{(j),\ell}))$
        Compute $\mathsf{msb}_\tau(Y_0'^{(j)} \oplus Z'^{(j),\ell})$ and store it along with $Y_0'^{(j)}$, $Z'^{(j),\ell}$, $Y_1'^{(j),\ell}$
    end for
end for
Find $i*$, $j*$, and $\ell*$ such that $\mathsf{msb}_\tau(Y_0^{(i*)} \oplus Z^{(i*),\ell*}) = \mathsf{msb}_\tau(Y_0'^{(j*)} \oplus Z'^{(j*),\ell*})$
$(Y_0, Z, Y_1) \leftarrow (Y_0^{(i*)}, Z^{(i*),\ell*}, Y_1^{(i*),\ell*})$, $(Y_0', Z', Y_1') \leftarrow (Y_0^{(j*)}, Z^{(j*),\ell*}, Y_1^{(j*),\ell*})$
Fix $C_* \in \{0,1\}^{\ell*}$ arbitrarily
$M_* \leftarrow Y_1 \oplus C_*$; $M_*' \leftarrow Y_1' \oplus C_*$
$M_0' \leftarrow M_0 \oplus \mathsf{lsb}_{n-\tau}(\Delta Y_0 \oplus \Delta Z)$
Find T such that both $Z \oplus 0^\tau \| M_0 = \mathsf{POLYVAL}_H(T, C_*)$ and $Z' \oplus 0^\tau \| M_0' = \mathsf{POLYVAL}_{H'}(T, C_*)$ are satisfied by inverting $\mathsf{POLYVAL}$
$C_0 \leftarrow Y_0 \oplus \mathsf{POLYVAL}_H(T, C_*)$
return $(K, T, M_0 \| M_*)$, $(K', T, M_0' \| M_*')$

this attack is $2^{\frac{\tau-\log_2 n}{2}+\frac{5}{2}}$, which is $2^{30.5}$, and this breaks the birthday bound for the encoded bit length τ. The number of BC calls is reduced by a factor of $2^{3.5}$, which is $2^{\frac{\log_2 n}{2}-\frac{1}{2}}$.

Experimental Verification. The attack assumes that the outputs from BC and $\mathsf{POLYVAL}$ are uniformly distributed, and we empirically verify it through an experiment. We can safely assume that the BC outputs for distinct inputs are uniformly distributed. However, it is not strictly true for $\mathsf{POLYVAL}$ because a specific relationship exists between $\overline{\mathsf{poly}}_H\big(\mathsf{zp}(\mathsf{msb}_i(Y))\big)$ and $\overline{\mathsf{poly}}_H\big(\mathsf{zp}(\mathsf{msb}_j(Y))\big)$ for $i \neq j$. By considering all choices of $|M_*|$, we cannot claim that there are n independent sets of $2^\tau/n$ pairs. Addressing the concern, we experimentally verify that these 2^τ distinct pairs are sufficient to generate a τ-bit collision. The exper-

iment is conducted on $\mathsf{EtE_P}$-HCTR2 instantiated with AES-128 for $\tau = 32$ bits. Following [9], our implementation of Algorithm 3 optimizes the basic HCTR2 components using AVX instructions, including AES-NI [16] and CLMUL [17]. For each run of Algorithm 3, we record the value of $2(i * + j*)$ as a metric of attack complexity. The entire experiment of running Algorithm 3 $1,000$ times finishes in $1,070$-second wall-clock time on a single core of the Intel Xeon w5-3423 processor. As a result, the average value of $2(i * + j*)$ is $40,282 \approx 2^{15.3}$, which is close to the theoretical value of $2^{15} = 2^{\frac{\tau - \log_2 n}{2} + \frac{5}{2}}$ in Algorithm 3, improving upon the original attack that requires $2^{18} = 2^{\frac{\tau}{2}+2}$ [9]. Table 2 in supplementary material also shows a concrete attack vector for $\ell = 104$.

Impact on Computationally-Bounded (CB) Attacks. Algorithm 3 is designed for IT adversaries, but we show that with some precomputation, its computational cost for CB adversaries is lower than that of the previous attack [9], so our attack improves upon that of [9] in the same setting.

Because the previous attack [9] considers a practical scenario, we assume that the BC is AES and $n = 128$. The cost of the previous attack is $2^{\frac{\tau}{2}+2}$ AES and $2^{\frac{\tau}{2}+1}$ $\overline{\mathsf{poly}}$ for random input of one block in length. The cost of one AES and one $\overline{\mathsf{poly}}$ are similar when using the AES new instructions (AES-NI) [16] and carry-less multiplication (CLMUL) [17] instructions, which are present in modern x86_64-based CPUs. For example, our benchmark on an Intel Xeon w5-3423 processor with the Sapphire Rapids architecture shows that the performance of AES-XCTR and POLYVAL is 0.318 cycles per byte (cpb) and 0.314 cpb, respectively. By assuming that the cost of one AES is equal to one $\overline{\mathsf{poly}}$, the previous attack cost is $2^{\frac{\tau}{2}+2} + 2^{\frac{\tau}{2}+1} = 6 \cdot 2^{\frac{\tau}{2}}$ AES.

First, we observe that the computational cost of $\overline{\mathsf{poly}}_H(\mathsf{zp}(\mathsf{msb}_\ell(Y_1)))$ for $\ell = 1, \ldots, n$ is much smaller than that of evaluating $\overline{\mathsf{poly}}$ on n random one-block inputs, because the length of $\mathsf{zp}(\mathsf{msb}_\ell(Y_1))$ of our attack fits in one block and the n inputs to $\overline{\mathsf{poly}}_H$ have a structure. Let $e[i]$ denote the n-bit unit vector, where the i-th bit is one, and the other bits are zero. Then, the multiplication of two n-bit values $Y_1 \otimes H$ can be represented by a linear combination of n terms $\bigoplus_{i=0}^{n-1}(Y_1 \cdot e[i]) \otimes H$. In Alg. 3, H is fixed at the beginning. Hence, when computing $\overline{\mathsf{poly}}_H(\mathsf{zp}(\mathsf{msb}_\ell(Y_1)))$, H is a known constant. This enables us to precompute $e[i] \otimes H$ for $i = 0, \ldots, n-1$, and the cost of $\overline{\mathsf{poly}}_H(\mathsf{zp}(\mathsf{msb}_\ell(Y_1)))$ for $\ell = 1, \ldots, n$ becomes at most n XORs with the precomputed values, and $\frac{n}{2}$ XORs on average, since an XOR is computed only when the ℓth bit of Y_1 is one.

We now count the computational cost of Alg. 3. As mentioned above, the BC must be computed $2^{\frac{\tau - \log_2 n}{2} + \frac{5}{2}}$ times in total. The number of iterations of each `for` loop with respect to X_0 is $2^{\frac{\tau - \log_2 n + 1}{2}}$, and we spend $\frac{n}{2}$ XORs in each loop. Moreover, there are two such `for` loops; hence, we need $2 \cdot \frac{n}{2} \cdot 2^{\frac{\tau - \log_2 n + 1}{2}}$ XORs, which is $2^{\frac{\tau + \log_2 n + 1}{2}}$ XORs. There is no generic way to convert the number of XORs to the BC cost. For AES, considering that AES consists of ten rounds and AES-NI computes one round per cycle, it is reasonable to assume that one AES is about ten XORs. Hence, we can estimate that the computational cost of

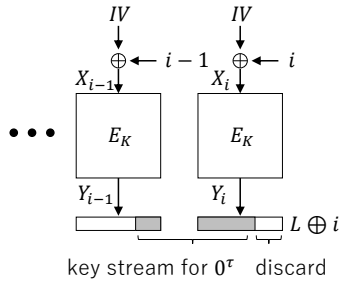

Fig. 7. New setup for divide-and-conquer.

Alg. 3 is $2^{\frac{\tau-\log_2 n}{2}+\frac{5}{2}} + 2^{\frac{\tau+\log_2 n+1}{2}}/10$ AES computations. With $n = 128$, this is $2^{\frac{\tau}{2}-1} + 2^{\frac{\tau}{2}+0.68}$, which is approximately $2.1 \cdot 2^{\tau/2}$ AES. Compared to $6 \cdot 2^{\tau/2}$ of the previous attack, Alg. 3 is more than twice as fast even for CB adversaries.

5.2 Attacks on $\mathsf{EtE_A}$-HCTR2

Halving the Number of BC Calls in Phase 1 for $\tau \leq \frac{3n}{2}$. We begin with a simple observation that halves the number of BC calls in Phase 1 when $\tau \leq \frac{3n}{2}$. In the previous attack explained in Sect. 3.2 [9], the encoded τ bits are positioned across the boundary between the i-th and $(i+1)$-th blocks as shown in Fig. 3, and the attacker generates a collision between L and L' in the least significant $2\tau/3$ bits.

We observe that the same attack works by positioning the encoded τ bits across the boundary between the i-th and $(i-1)$-th blocks. Specifically, $2\tau/3$ bits are positioned in block i, and $\tau/3$ bits are positioned in block $i-1$ as shown in Fig. 7. This changes Phase 1 of the attack so that it finds L and L' satisfying $\mathsf{msb}_{2\tau/3}(L) = \mathsf{msb}_{2\tau/3}(L')$ rather than the LSB as in the previous attack. Then, Phase 2 finds an i such that the least significant $\tau/3$ bits of the $(i-1)$-th block collide rather than the most significant $\tau/3$ bits of the $(i+1)$-th block. By combining the previous and current approaches, the attacker can test collisions in two positions, the LSB and the MSB, reducing the number of BC calls by half.

A $\log_2(n-1)$-bit Improvement for $\tau = 2n$. The above idea for halving the number of BC calls in Phase 1 can be used more efficiently for improving Phase 2 with $\tau = 2n$. Note that, when $\tau > \frac{3n}{2}$, the cost of Phase 1 is small and the cost of Phase 2 is the bottleneck.

The previous attack [9] generates an n-bit collision for the i-th block in Phase 1, and then finds an n-bit collision for the $(i+1)$-th block in Phase 2 (Fig. 8, left), resulting in an attack complexity $2^{\tau-n} = 2^n$. However, the n-bit collision in Phase 2 can occur in any consecutive $2n$-bit segment spanning from the $(i-1)$-th block to the $(i+1)$-th block as shown in the right-hand side of Fig. 8.

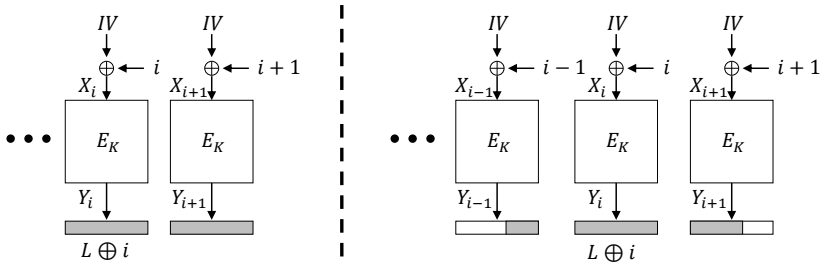

Fig. 8. Previous (left) and new (right) setup for divide-and-conquer with $\tau = 2n$.

Lemma 1. *By looking for a collision on any consecutive* $2n$ *bits spanning* Y_{i-1}, Y_i, *and* Y_{i+1}, *the probability of finding a collision increases by a factor of* $\frac{n+2}{2}$ *compared with the case of finding a collision in* Y_i *and* Y_{i+1}.

Proof. For every i, IV and IV', computed as $E_K^{-1}(L \oplus \mathtt{bin}_n(i)) \oplus \mathtt{bin}_n(i)$ and $E_{K'}^{-1}(L' \oplus \mathtt{bin}_n(i)) \oplus \mathtt{bin}_n(i)$, respectively, can be regarded as random n-bit values. Then, the corresponding two n-bit differences ΔY_{i-1} and ΔY_{i+1} are also uniformly and independently chosen from $\{0,1\}^n$. Phase 1 ensures that $\Delta Y_i = 0$. We then evaluate the probability that $\Delta Y_{i-1} \| \Delta Y_i \| \Delta Y_{i+1}$ contains $2n$ consecutive zero bits.

First, we count the number of sequences (favorable sequences) that contain $2n$ consecutive bits of zeros. Let j be the position at which $2n$ consecutive zeros occur for the first time. Here, we number the position such that the MSB of ΔY_{i-1} is at position 1 and the LSB of ΔY_{i+1} is at position $3n$. Then, the possible values of j are $2n, 2n+1, \ldots, 3n$. Put $m = j - 2n$, so $m = 0, 1, \ldots, n$.

Case 1: $m = 0$**.** The first $2n$ bits are 0, i.e., $\Delta Y_{i-1} = 0$. ΔY_{i+1} can take any value; thus, the number of favorable sequences is 2^n.

Case 2: $m \geq 1$**.** Position $m-1$ must contain a non-zero difference; otherwise, this would contradict the assumption that $2n$ consecutive zeros appear for the first time at position j. Hence, the $m-1$ positions before the start of the run of $2n$ consecutive zeros can take any difference, and the $n-m$ bits after the end of the run can take any difference. Thus, each $m = 1, \ldots, n$ gives $2^{m-1} \cdot 2^{n-m} = 2^{n-1}$ favorable sequences.

Hence, the total number of favorable sequences is $2^n + n \cdot 2^{n-1} = (n+2) \cdot 2^{n-1}$. Because the total number of sequences is 2^{2n}, the exact probability is therefore $\frac{(n+2) \cdot 2^{n-1}}{2^{2n}}$, which is $\frac{n+2}{2^{n+1}}$.

In the previous attack, the probability of generating a $2n$-bit collision in $Y_i \| Y_{i+1}$ is 2^{-n}. Hence, the ratio is $\frac{n+2}{2^{n+1}} / \frac{1}{2^n} = \frac{n+2}{2}$. □

In summary, by searching for a pair that contains a $2n$-bit consecutive zero-difference across Y_{i-1}, Y_i, Y_{i+1}, the number of i that must be tested in Phase 2 is reduced by a factor of $\frac{n+2}{2}$, which eventually reduces the number of BC calls

by the same factor. In addition, since this attack succeeds for a smaller value of i, it can shorten the message length required for the attack.

6 Quantum Attacks on EtE-HCTR2

In this section, we present quantum attacks on EtE-HCTR2. Although we assume basic knowledge of quantum computation, first we briefly review the quantum memory models, the quantum search algorithm, and the quantum collision search algorithms used in this paper. We then present quantum attacks on EtE_A-HCTR2 and EtE_P-HCTR2 under several memory assumptions. For both EtE_A-HCTR2 and EtE_P-HCTR2, our attacks are obtained by quantumly accelerating Chang et al.'s classical attack shown in Sect. 3.2 and 3.3. For EtE_A-HCTR2, we further develop this approach by introducing an extension based on multiple-collision search. Throughout this section, we count the time for one BC call, together with a constant number of simple operations such as XORs and truncations, as $O(1)$, regardless of whether they are implemented classically or quantumly.

6.1 Quantum Memory Models and Algorithmic Tools

Quantum Random Access Memory (qRAM). qRAM is a memory model in which stored data can be accessed coherently, including access in quantum superposition. Let $\mathcal{L} \coloneqq \{y_1, \ldots, y_M\}$ be a data structure consisting of M elements. Then, the qRAM gate $\mathsf{qRAM}_\mathsf{read}$ is defined as

$$\mathsf{qRAM}_\mathsf{read} : |i\rangle\,|x\rangle \mapsto |i\rangle\,|x \oplus y_i\rangle\,.$$

Generally, the data structure $\mathcal{L}$ does not need to be stored as a quantum state. It suffices that there exists a quantum circuit that, given an address i, coherently computes the corresponding value y_i. The address register i may be given in superposition. In particular, the most commonly considered qRAM is *Quantum-Accessible Classical Memory* (QACM), in which $\mathcal{L}$ is stored in classical memory but can be read in superposition. By contrast, quantum search algorithms based on the quantum walk often use *Quantum-Accessible Quantum Memory* (QAQM) where the data structure $\mathcal{L}$ is maintained on qubits as $|y_1, \ldots, y_M\rangle$. Therefore, unlike QACM, QAQM supports not only the read operation $\mathsf{qRAM}_\mathsf{read}$ but also write operations, which can be realized by swapping quantum registers.

For consistency of notation, we refer to a fully classical RAM as *Classically Accessible Classical Memory* (CACM) in the rest of the paper.

Quantum Search (Quantum Amplitude Amplification) [6]. Let $\mathcal{X}$ be a finite set and let $f : \mathcal{X} \to \{0, 1\}$ be a Boolean function. Suppose that a quantum algorithm SETUP prepares a state whose amplitude on the good subspace $\mathcal{G} \coloneqq \{x \in \mathcal{X} \mid f(x) = 1\}$ is $|\alpha|$. Then, quantum amplitude amplification (QAA) finds a good element with constant probability using $O(1/|\alpha|)$ iterations. Let $\mathcal{T}_\mathsf{SETUP}$ and $\mathcal{T}_\mathsf{FLIP}$ denote the costs of the state preparation and the phase flip for f, respectively. Then, the total time complexity is $\frac{1}{|\alpha|}\left(\mathcal{T}_\mathsf{SETUP} + \mathcal{T}_\mathsf{FLIP}\right)$.

Grover's algorithm. Grover's algorithm [15] is the special case of QAA where SETUP prepares the uniform superposition over $\mathcal{X}$ using Hadamard gates. In the following, we also refer to this procedure simply as Grover's search.

Quantum Collision Search. A representative quantum collision search algorithm was proposed by Brassard, Høyer, and Tapp [7], known as the BHT algorithm. Let $f : \{0,1\}^n \to \{0,1\}^m$ be a target function. The BHT algorithm consists of a precomputation phase and a quantum search, where SETUP creates a uniform superposition over the input space $\{0,1\}^n$, corresponding to Grover's search. More precisely, it proceeds as follows.

1. **Precomputation.** Choose a uniformly random subset $\mathcal{S} \subseteq \{0,1\}^n$ of size 2^t. Compute $f(x)$ for all $x \in \mathcal{S}$, and store the list $\mathcal{L} \coloneqq \{(x, f(x))\}_{x\in\mathcal{S}}$ in QACM.
2. **Quantum search.** Define the following Boolean function

$$F_{\mathcal{L}}^{\mathsf{BHT}}(x) \coloneqq \begin{cases} 1 \text{ if } \exists (x', f(x')) \in \mathcal{L} \text{ s.t. } x' \neq x \wedge f(x') = f(x), \\ 0 \text{ otherwise.} \end{cases}$$

   and run Grover's search with the following FLIP.
   - FLIP is the phase flip operation that maps $|x\rangle$ to $-|x\rangle$ iff $F_{\mathcal{L}}^{\mathsf{BHT}}(x) = 1$ for all $x \in \{0,1\}^n$.

   Finally, with high probability, it outputs distinct inputs $x, x' \in \{0,1\}^n$ such that $f(x) = f(x')$.

In the following, for a function $f : \{0,1\}^n \to \{0,1\}^m$ and an integer $t \geq 0$, we write $\mathsf{BHT}(f,t)$ for the above BHT algorithm applied to f, where the precomputation phase uses a uniformly random subset of $\{0,1\}^n$ of size 2^t.

The Complexity. Here, we consider the computational complexity of $\mathsf{BHT}(f,t)$. The precomputation takes 2^t times, and uses QACM of size 2^t to store the list. For a random input $x \in \{0,1\}^n$, the probability that $F_{\mathcal{L}}^{\mathsf{BHT}}(x) = 1$ is approximately 2^{t-m}. Hence, the quantum search takes $2^{(m-t)/2}$ time. Therefore, the total time complexity of $\mathsf{BHT}(f,t)$ is approximately $2^t + 2^{(m-t)/2}$.

The Optimal Complexity. The BHT algorithm is optimal when $t = m/3$, in which case we obtain the time and QACM complexity $O(2^{m/3})$.

Quantum Collision Search without (Exponential) qRAM. The precomputation phase in the BHT algorithm requires qRAM (QACM) of exponential size. Since large-scale qRAM is widely regarded as difficult to implement in practice, the BHT algorithm remains far from practical deployment. Motivated by this limitation, Chailloux, Naya-Plasencia, and Schrottenloher [8] proposed an efficient quantum algorithm for collision finding that does not rely on exponential qRAM. This algorithm is commonly referred to as the CNS algorithm. Like the BHT algorithm, the CNS algorithm consists of a precomputation and a

quantum search. The main differences are that the precomputed data are stored in CACM rather than QACM, and that the search phase is formulated in the more general framework of QAA, rather than as a plain Grover's search. More precisely, it proceeds as follows.

1. **Precomputation.** Define $S_r^f \coloneqq \{(x, f(x)) : \exists z \in \{0,1\}^{m-r}, f(x) = 0^r \| z\}$. Choose a uniformly random subset $\mathcal{S} \subseteq S_r^f$ of size 2^{t-r} where $t \leq m$. Compute $f(x)$ for all $x \in \mathcal{S}$, and store the list $\mathcal{L} \coloneqq \{(x, f(x))\}_{x \in \mathcal{S}}$.
2. **Quantum search.** Define the following Boolean function

$$F_{\mathcal{L}}^{\mathsf{CNS}}(x) \coloneqq \begin{cases} 1 \text{ if } \exists (x', f(x')) \in \mathcal{L} \text{ s.t. } f(x) = f(x'), \\ 0 \text{ otherwise.} \end{cases}$$

   and run QAA instantiated with the following SETUP and FLIP.
   - SETUP is a quantum algorithm for preparing $\frac{1}{\sqrt{|S_r^f|}} \sum_{x \in S_r^f} |x, f(x)\rangle$.
   - FLIP is the phase flip operation that maps $|x\rangle$ to $-|x\rangle$ iff $F_{\mathcal{L}}^{\mathsf{CNS}}(x) = 1$ for all $x \in S_r^f$.

   Finally, with high probability, it outputs distinct inputs $x, x' \in \{0,1\}^n$ such that $f(x) = f(x')$.

Similarly to the BHT algorithm, for a function $f : \{0,1\}^n \to \{0,1\}^m$ and integers $t, r \geq 0$ with $r \leq t \leq m$, we write $\mathsf{CNS}(f, t, r)$ for the above CNS collision-finding algorithm applied to f.

The Complexity. Here, we consider the computational complexity of $\mathsf{CNS}(f, t, r)$. In the precomputation phase, it must collect 2^{t-r} inputs x such that the first r bits of $f(x)$ are all zero. Each such input can be obtained by running Grover's search and then measuring the output state, which costs $O(2^{r/2})$ time. Therefore, the total time complexity of the precomputation phase is $2^{t-r} \cdot 2^{r/2} = 2^{t-r/2}$. To store 2^{t-r} input and output pairs, we use CACM with a size of 2^{t-r}. In the QAA, SETUP algorithm requires $2^{r/2}$ to create the uniform superposition over S_r^f. On the other hand, FLIP takes 2^{t-r}, since evaluating $F_{\mathcal{L}}^{\mathsf{CNS}}$ requires scanning all 2^{t-r} entries $\mathcal{L}$ stored in CACM. The fraction of marked items is approximately $\frac{2 \cdot 2^{t-r}}{2^{m-r}} = 2^{t-m+1}$, and hence the number of QAA iterations is approximately $2^{(m-t-1)/2}$. Thus, the total time complexity is $2^{t-r/2} + 2^{(m-t-1)/2} \left(2^{r/2} + 2^{t-r}\right)$.

The Optimal Complexity. The CNS algorithm is optimal when $t = 3m/5$ and $r = 2m/5$, in which case the time complexity becomes $O(2^{2m/5})$ and the CACM complexity becomes $O(2^{m/5})$.

6.2 Trivial Quantum Attacks on EtE-HCTR2

An approach for quantum attacks on EtE-HCTR2 is to use Grover's search to find two distinct keys that, for a fixed ciphertext (and a fixed tweak), yield valid decryptions satisfying the EtE redundancy condition. This results in a quantum attack with time complexity $O(2^{\tau/2})$ and constant memory complexity. However,

this trivial quantum speedup does not outperform the classical attacks of Chang et al. For $\mathsf{EtE_A}$-HCTR2, it is slower than their classical attack. For $\mathsf{EtE_P}$-HCTR2, it matches the complexity of their classical attack. This motivates us to consider further accelerating Chang et al.'s already fast classical attacks by means of quantum algorithms.

Complexity notation. In the rest of this section, we present efficient quantum attacks based on Chang et al.'s classical attacks and analyze their complexities. For this purpose, we write $\mathcal{T}$ for the total time complexity of an attack. Similarly, we write $\mathcal{M}_{\mathsf{QACM}}$, $\mathcal{M}_{\mathsf{CACM}}$, and $\mathcal{M}_{\mathsf{QAQM}}$ for the memory complexities in the QACM, CACM, and QAQM models, respectively.

6.3 Quantum Attacks on $\mathsf{EtE_A}$-HCTR2

As mentioned above, our quantum attacks are based on the classical attack by Chang et al. We first present our quantum attack framework and its instantiations in the QACM and CACM models.

Framework for Our Quantum Attack. Here, we consider how to accelerate Chang et al.'s classical attack shown in Sect. 3.2, in the quantum setting. **Phase 1** of the classical attack involves searching for a colliding key pair (K, K') such that the last $2\tau/3$ bits of the corresponding values L and L' coincide, and this step admits a quantum speedup using quantum collision search algorithms. **Phase 2** searches for a counter value i such that the $\mathsf{msb}_{\tau/3}$ values of the $(i+1)$st XCTR blocks coincide, and this step can likewise be accelerated quantumly. By contrast, **Phase 3** requires generating a sequence of message blocks whose length is determined by the counter value found in **Phase 2**. Since the maximum counter value is exponential in τ, the cost of this phase is not negligible. Moreover, **Phase 3** is essentially an explicit output-generation procedure for a definite plaintext, and thus does not admit a quantum speedup.

Therefore, a straightforward quantum acceleration of the classical attack is not necessarily optimal. Even if we accelerate **Phase 1** and **Phase 2** using quantum computation, **Phase 3** remains the bottleneck for the overall computational complexity under the current parameters.

This leads us to consider a different trade-off between **Phase 1** and **Phase 2**: increasing the number of collision bits handled in **Phase 1** reduces the number of bits that remain to be matched in **Phase 2**. Consequently, the maximum possible counter value found in **Phase 2** becomes smaller, and hence, the maximum number of message blocks that must be generated in **Phase 3** also decreases.

This observation suggests that the classical partition is no longer optimal in the quantum setting. Instead of fixing the split to $2\tau/3$ and $\tau/3$ as in the classical attack, we consider a more general partition: **Phase 1** handles s bits, while **Phase 2** handles the remaining $(\tau - s)$ bits. Increasing s makes **Phase 1** more expensive, but this increase is mitigated by the quantum speedup. At the same time, it decreases the number of bits to be matched in **Phase 2**, thereby

reducing the maximum possible counter value returned there and, in turn, the cost of **Phase 3**. Our quantum attack is obtained by optimizing this trade-off.

We now describe our quantum attack on $\mathsf{EtE}_{\mathsf{A}}$-HCTR2 based on the above idea. Our quantum attack is obtained by generalizing the parameters of the classical attack and replacing **Phase 1** and **Phase 2** with their quantum counterparts, while keeping **Phase 3** classical. The resulting attack proceeds as follows.

Phase 1. Searching for a pair (K, K') that collides on the last s bits of (L, L').
Phase 2. Searching for i that derives a collision on $\mathsf{msb}_{\tau-s}(Y_{i+1})$.
Phase 3. Generating an input pair with a length of $O(2^{\tau-s})$ blocks.

The complexity for **Phase 1** depends on whether the BHT or CNS algorithm is used for the collision search. The complexity for **Phase 2** is $O(2^{(\tau-s)/2})$, since we use Grover's search on $\{0,1\}^{\tau-s}$. The complexity of **Phase 3** is bounded by the maximum possible counter value returned in **Phase 2**. Let $\mathcal{T}_1$ be the complexity for **Phase 1**, which is determined by whether the BHT or CNS algorithm is used. Then, the time complexity $\mathcal{T}$ of our quantum attack is as

$$\mathcal{T} = \mathcal{T}_1 + 2^{\frac{\tau-s}{2}} + 2^{\tau-s}.$$

The memory complexity depends on how **Phase 1** is instantiated. In the following, we analyze the BHT and CNS algorithm-based instantiations.

Instantiation with the BHT Algorithm. We instantiate **Phase 1** with the BHT algorithm. More precisely, define $f_s : \{0,1\}^k \to \{0,1\}^s$ such that, for all $K \in \{0,1\}^k$, $f_s(K) \coloneqq \mathsf{lsb}_s(L)$ where $L = E_K(\mathtt{bin}_n(1))$. Then, **Phase 1** consists of finding a collision in f_s, that is, a pair (K, K') satisfying $f_s(K) = f_s(K')$. We implement this phase using the BHT algorithm $\mathsf{BHT}(f_s, t)$ for some t with $0 \le t \le s$. Let $\mathcal{M}_{\mathsf{QACM}}$ be the QACM complexity. Then, $\mathcal{M}_{\mathsf{QACM}} = 2^t$ is required to store the precomputed list. $\mathcal{T}_1$ corresponds to the time complexity of the BHT algorithm when the target function has s output bits, and it is $2^t + 2^{(s-t)/2}$. Thus, the attack complexity $\mathcal{T}$ for this case is as follows.

$$\mathcal{T} = 2^t + 2^{\frac{s-t}{2}} + 2^{\frac{\tau-s}{2}} + 2^{\tau-s}.$$

The complexities are balanced and optimal when $t = \tau/4$, $s = 3\tau/4$, and we have

$$\mathcal{T} = \mathcal{M}_{\mathsf{QACM}} = O(2^{\tau/4}).$$

The above optimization assumes that the choice $s = 3\tau/4$ is feasible. Since s denotes the number of collided bits of the n-bit value L, we must have $s \le n$. Hence, the above attack with the optimal complexity $O(2^{\tau/4})$ is valid only for $s = 3\tau/4 \le n$, that is, $\tau \le 4n/3$. If $\tau > 4n/3$, then we can no longer assume that $s = 3\tau/4$, and thus the resulting attack complexity becomes larger than $O(2^{\tau/4})$ in this range.

Instantiation with the CNS Algorithm. We instantiate **Phase 1** with the CNS algorithm. As for the BHT-algorithm-based instantiation, our aim is to find an s-bit collision of f_s defined in our BHT algorithm-based instantiation. Therefore, the CNS instance in this case can be denoted as $\mathsf{CNS}(f_s, t, r)$ for some t, r with $0 \leq r \leq t \leq s$. Let $\mathcal{M}_{\mathsf{CACM}}$ be the CACM complexity. Then, $\mathcal{M}_{\mathsf{CACM}} = 2^{t-r}$ is required to store the precomputed list. $\mathcal{T}_1$ corresponds to the time complexity of the CNS algorithm when the target function has s output bits, and it is $2^{t-r/2} + 2^{(s-t-1)/2}(2^{r/2} + 2^{t-r})$. Thus, the time complexity is

$$\mathcal{T} = 2^{t-\frac{r}{2}} + 2^{\frac{s-t-1}{2}}(2^{\frac{r}{2}} + 2^{t-r}) + 2^{\frac{\tau-s}{2}} + 2^{\tau-s}.$$

The complexities are optimal when $s = 5\tau/7, t = 3\tau/7, r = 2\tau/7$, and we have

$$\mathcal{T} = O(2^{2\tau/7}),\ \mathcal{M}_{\mathsf{CACM}} = O(2^{\tau/7}).$$

Similarly to the BHT algorithm-based instantiation, the above attack with the optimal complexity $O(2^{2\tau/7})$ is valid only when $\tau \leq 7n/5$. If $\tau > 7n/5$, then we can no longer assume the optimal choice $s = 5\tau/7$, and thus the resulting attack complexity becomes larger than $O(2^{2\tau/7})$ in this range.

Comparison with Quantum Speedups of Chen et al.'s Attack. For $\mathsf{EtE_A}$-HCTR2, one may also consider a direct quantum speedup of Chen et al.'s attack. Although Chen et al.'s attack is slower than Chang et al.'s attack, it has a simpler structure: it does not involve the generation of message blocks, and all of its dominant steps admit direct quantum speedup. Since the dominant step of their attack is a τ-bit collision search, instantiating this step with the BHT algorithm yields $O(2^{\tau/3})$ time and QACM complexities, while instantiating it with the CNS algorithm yields $O(2^{2\tau/5})$ time complexity and $O(2^{\tau/5})$ CACM complexity. However, these complexities are still larger than those of our quantum attacks based on Chang et al.'s attack derived above. The same holds for the range of τ considered in the next subsection.

We note in passing that Chen et al.'s attack is still relevant to $\mathsf{EtE_S}$-HCTR2. Indeed, Chang et al.'s attack is not applicable to $\mathsf{EtE_S}$-HCTR2, whereas Chen et al.'s attack trivially applies. Thus, the quantum speedup of Chen et al.'s attack is currently considered the best quantum attack against $\mathsf{EtE_S}$-HCTR2.

6.4 Quantum Attacks on $\mathsf{EtE_A}$-HCTR2 using Multiple Collisions Search

The attacks proposed in Sect. 6.3 are faster than generic quantum collision search only in restricted parameter regimes. More precisely, the BHT-based instantiation achieves an improvement only when $\tau \leq 4n/3$, whereas the CNS-based instantiation does so only when $\tau \leq 7n/5$.

Chang et al. encountered a similar difficulty in the classical setting when $\tau > 3n/2$. To improve the attack in this case, they use a multi-collision strategy: they first find, for some integer $u \geq 2$, a set of secret keys $\{K^{(1)}, \ldots, K^{(u)}\}$ such

that $E_{K^{(1)}}(\mathtt{bin}_n(1)) = \cdots = E_{K^{(u)}}(\mathtt{bin}_n(1))$, namely, a u-way collision on L, in **Phase 1** of the classical attack. In **Phase 2**, we search for a counter value using the u keys obtained in **Phase 1**. Since it suffices that some pair among the u values $\mathsf{msb}_{\tau-n}(Y_{i+1})$ collides, the success probability in this phase is amplified by a factor of u. As a result, the complexity of **Phase 2** is reduced by the same factor, and this also lowers the complexity of **Phase 3**.

Framework for the Quantum Attack using Multiple Collisions. For values of τ outside the range of quantum attacks in Sect. 6.3, our quantum attack achieves an exponential speedup by collecting multiple collisions in **Phase 1**. More precisely, we instantiate **Phase 1** with a quantum algorithm depending on the underlying RAM model. We stress that a multiple collision is different from a multi-collision: the former means a collection of several colliding pairs (or, more generally, tuples of size at least two), while the latter means a set of multiple inputs that all collide on a single shared image.

The procedure for our quantum attack with the multiple collision search is as follows:

Phase 1. Searching for multiple key pairs $(K^{(1)}, K'^{(1)}), \ldots, (K^{(2^v)}, K'^{(2^v)})$ that collide, such that $E_{K^{(a)}}(\mathtt{bin}_n(1)) = E_{K'^{(a)}}(\mathtt{bin}_n(1))$ for all $a \in [2^v]$, and store them in a list $\mathcal{L}_K$.
Phase 2. Searching for i that derives a collision on $\mathsf{msb}_{\tau-n}(Y_{i+1})$.
Phase 3. Generating an input pair with a length of $O(2^{\tau-n})$.

Let ℓ denote the counter value returned in **Phase 2**. Suppose that **Phase 1** outputs 2^v colliding key pairs. For any fixed counter value i and any fixed key pair, the probability that the corresponding values $\mathsf{msb}_{\tau-n}(Y_{i+1})$ collide is $2^{-(\tau-n)}$. Hence, the probability p_{hit} that at least one collision occurs among the 2^v key pairs is $p_{\mathsf{hit}} = 2^{v-(\tau-n)}$. For any integer $j \geq 1$ and any constant $c > 0$, we have $\Pr[\ell > cj] = (1 - p_{\mathsf{hit}})^{cj} \leq e^{-p_{\mathsf{hit}}cj}$. Now we set $j = 2^{\tau-n-v}$. Since $p_{\mathsf{hit}}j \approx 1$, it follows that $\Pr[\ell \leq c \cdot 2^{\tau-n-v}] \approx 1 - e^{-c}$. When $c = 4$, this probability is approximately 0.98. That is, with probability approximately 98%, the space $\{0,1\}^{\tau-n-v+2}$ contains a counter value that yields a collision on $\mathsf{msb}_{\tau-n}(Y_{i+1})$. Hence, it suffices to perform Grover's search over this space in **Phase 2**. As the FLIP operator, we use the quantum oracle of the following function:

$$F^{\mathsf{cnt}}_{\mathcal{L}}(i) := \begin{cases} 1 \text{ if } \exists (K^{(a)}, K'^{(a)}) \in \mathcal{L}_K \text{ s.t. } \mathsf{msb}_{\tau-n}(Y^{(a)}_{i+1}) = \mathsf{msb}_{\tau-n}(Y'^{(a)}_{i+1}), \\ 0 \text{ otherwise,} \end{cases}$$

where $Y^{(a)}_{i+1}$ and $Y'^{(a)}_{i+1}$ denote the $(i+1)$st XCTR key-stream blocks generated under the keys $K^{(a)}$ and $K'^{(a)}$, respectively. The FLIP based on the above function must verify, for each candidate counter value, whether the counter condition is satisfied for at least one of the 2^v key pairs stored in $\mathcal{L}_K$. This requires searching over the stored key pairs, and it depends on whether qRAM or CACM is used for storing them. Let $\mathcal{T}_{\mathsf{check}}$ be the cost to search the list in the FLIP operator. Then, the complexity of **Phase 2** is $2^{(\tau-n-v+2)/2} \times \mathcal{T}_{\mathsf{check}}$. Recall that the complexity

of **Phase 3** can be upper-bounded by the maximum possible counter value returned in **Phase 2**. Since the maximum counter value is at most $2^{\tau-n-v+2}$, it follows that the complexity of **Phase 3** is approximately $2^{\tau-n-v}$.

As per Sect. 6.3, we let $\mathcal{T}_1$ be the time complexity for **Phase 1**. Then, the time complexity for our attack is

$$\mathcal{T} = \mathcal{T}_1 + 2^{\frac{\tau-n-v+2}{2}} \times \mathcal{T}_{\mathsf{check}} + 2^{\tau-n-v}.$$

The memory complexity depends on whether **Phase 1** is instantiated with qRAM or CACM. In both cases, memory of size 2^v is required to store the colliding key pairs. In addition, the memory required by the multiple-collision search used in **Phase 1** must be taken into account.

In the following, we present three instantiations of the above framework. We first present instantiations in the QACM and CACM models. Two instantiations apply when the key length is sufficiently large relative to the block length, and in particular cover most common settings such as $k = n$ and $k = 2n$. We then present a QAQM-based instantiation. Although QAQM is generally more costly than QACM, this instantiation achieves the same complexity as the QACM-based attack while also applying in certain parameter ranges with $k < n$.

Quantum Attack in the QACM Model. In this model, the attack proposed in Sect. 6.3 is valid only for $\tau \leq 4n/3$, and we consider the regime $4n/3 \leq \tau < 2n$. In **Phase 1**, to collect 2^v colliding key pairs using a BHT algorithm-based procedure, we first construct a single precomputed list of size 2^t ($t > v$) and reuse this list throughout the procedure. We then run Grover's search repeatedly, each time finding a new key that collides with one of the elements in the list. After i colliding pairs have already been found, about $2^t - i$ values remain available to produce a new pair. Hence, the total number of Grover iterations required to collect 2^v colliding key pairs is $\sum_{i=0}^{2^v-1} \sqrt{2^n/(2^t-i)} \approx 2^{v+\frac{n-t}{2}}$. By considering the cost for the precomputation 2^t, the time complexity for **Phase 1** is $\mathcal{T}_1 = 2^t + 2^{v+(n-t)/2}$. In this setting, the colliding key pairs found in **Phase 1** are stored in QACM. Therefore, $\mathcal{T}_{\mathsf{check}} = 2^{v/2}$. Thus, the time and QACM complexities for the quantum attack in this model are

$$\mathcal{T} = 2^t + 2^{v+\frac{n-t-1}{2}} + 2^{\frac{\tau-n+2}{2}} + 2^{\tau-n-v},\ \mathcal{M}_{\mathsf{QACM}} = 2^t + 2^v,$$

respectively. The complexities are balanced and optimal when $t = (2\tau - n)/5$ and $v = (3\tau - 4n)/5$, and then we have

$$\mathcal{T} = \mathcal{M}_{\mathsf{QACM}} = O(2^{\frac{2\tau-n}{5}}).$$

This instantiation is valid only for $v \leq 3k - 2n$.[10] Combining the optimal choice $v = (3\tau - 4n)/5$, the condition on the block and key lengths for the success of

[10] This is the usual validity condition for the BHT algorithm repeating: assume that the precomputed list has size 2^t, then the expected number of further colliding inputs is about 2^{k+t-n}, so extracting 2^v colliding pairs requires $v \leq k + t - n$. Combining this with the optimal choice $t = (n + 2v)/3$ gives $v \leq 3k - 2n$.

this procedure is $k \geq (\tau + 2n)/5$. Since we assume $4n/3 \leq \tau < 2n$, this covers the common settings $k = n$ and $k = 2n$, as well as all settings with $k \geq n$.

Quantum Attack in the CACM Model. In this model, the attack proposed in Sect. 6.3 is valid up to $\tau \leq 7n/5$. Here, we consider the regime $7n/5 \leq \tau < 2n$.

In **Phase 1**, to collect 2^v colliding key pairs using a CNS algorithm-based procedure, we first construct a single precomputed list of size 2^{t-r} $(t - r > v)$ elements and reuse this list throughout the procedure.

We then run QAA iterations, repeatedly, each time finding a new key that collides with one of the elements in the list. Similarly to the BHT algorithm-based procedure explained above, after i colliding pairs have already been found, about $2^{t-r+1} - i$ values remain available to produce a new pair. Hence, the total number of QAA iterations required to collect 2^v colliding pairs is $\sum_{i=0}^{2^v-1} \sqrt{2^{n-r}/(2^{t-r+1} - i)} \approx 2^{v+\frac{n-t-1}{2}}$. Considering the complexity of the list generation, SETUP and FLIP, the time complexity of **Phase 1** is

$$\mathcal{T}_1 = 2^{v+\frac{n-t-1}{2}} \left(2^{\frac{r}{2}} + 2^{t-r}\right) + 2^{t-\frac{r}{2}}.$$

In this setting, the colliding key pairs found in **Phase 1** are stored in CACM. Therefore, $\mathcal{T}_{\mathsf{check}} = 2^v$. Thus, the time and CACM complexities are

$$\mathcal{T} = 2^{v+\frac{n-t-1}{2}} \left(2^{\frac{r}{2}} + 2^{t-r}\right) + 2^{t-\frac{r}{2}} + 2^{\frac{\tau-n+v+2}{2}} + 2^{\tau-n-v},$$
$$\mathcal{M}_{\mathsf{CACM}} = 2^{t-r} + 2^v,$$

respectively. When $v = (5\tau - 7n)/9$, $t = (2\tau - n)/3$, $r = 2(2\tau - n)/9$, the complexities are optimal, and then we have

$$\mathcal{T} = O(2^{\frac{2(2\tau-n)}{9}}),\ \mathcal{M}_{\mathsf{CACM}} = O(2^{\frac{2\tau-n}{9}}).$$

Similarly to the repeated BHT algorithm in the QACM instantiation, this instantiation is valid for $k \geq (\tau + n)/3$. Since we assume $7n/5 \leq \tau < 2n$, this covers the common settings $k = n$ and $k = 2n$, as well as all settings with $k \geq n$.

Quantum Attack in the QAQM Model. As in the QACM instantiation, the attack proposed in Sect. 6.3 is valid only up to $\tau \leq 4n/3$ in this model. A limitation of the QACM-based instantiation is that its validity requires a sufficiently large key length relative to the block length. We can partially overcome this limitation using Bonnetain et al.'s algorithm [5], which was recently proposed. This requires the QAQM, which is generally more costly than the QACM. However, it also applies in certain parameter regimes with $k < n$, more precisely when $k \leq n \leq 2k$ and $v \leq 2k - n$.

The optimal complexity of Bonnetain et al.'s algorithm is $2^{n/3+2v/3}$ time and $2^{n/3+2v/3}$ QAQM complexity. Although the memory model is QAQM rather than QACM, the analysis is the same as in the repeated-BHT instantiation in the QACM model. The same also holds for the overall complexity, and we have

$$\mathcal{T} = \mathcal{M}_{\mathsf{QAQM}} = O(2^{\frac{2\tau-n}{5}}).$$

Table 2. Concrete vector for $\mathsf{EtE_P}$-HCTR2 with $\tau = 32$ bits.

Parameter	Value
ℓ	104
K	d72b0553676190918b3143f7ec7141d9
K'	2c7883476436268ec208cd2d80a5d93f
T	b4eebd474812f401e04e427fe2b5ae4523e8de2fdd3ed36b71147b6b6cf7c3fd
$0^\tau \| M$	000000003ae25ebd45cef1542944d4f08b8bff46117f8e35f05d35d8a8
$0^\tau \| M'$	00000000f8e328d47671aabd7db4a1327b21ecea2fb2b2d298bd9f4ef8
C	a5a789227c191733fce95eb60743c5cacf58bc45d7c4240d3ddb1d9829

Since the above complexity can be obtained with $v = (3\tau - 4n)/5$ as in the QACM instantiation, and since $k \leq n \leq 2k$ and $v \leq 2k - n$, this instantiation is valid for $(3\tau + n)/10 \leq k \leq n$.

6.5 Quantum Attacks on $\mathsf{EtE_P}$-HCTR2

The quantum attacks on $\mathsf{EtE_P}$-HCTR2 follow directly from the classical attack by Chang et al.; they are obtained by replacing its collision-finding part with either the BHT or CNS algorithm. If we use the BHT algorithm, both the time and QACM complexity are $O(2^{\tau/3})$. On the other hand, if we use the CNS algorithm, the time complexity is $O(2^{2\tau/5})$ and the CACM complexity is $O(2^{\tau/5})$.

7 Concluding Remarks

This paper analyzed the key-committing security of EtE-HCTR2. First, we analyzed the effect of the IV mask L for $\mathsf{CMT_k}$ and $\mathsf{CMT_k^*}$ security. Without L, $O(1)$ attacks are possible, implying that L provides a certain level of security. Next, focusing on the polynomial gap between the previous bounds, we developed IT attacks that reduced the number of BC calls, making this gap closer. This attack, even as a CB attack, is more than twice as fast as the existing one. Finally, we proposed quantum attacks both with and without qRAM. The results on $\mathsf{EtE_P}$-HCTR2 could be obtained by simple applications of the quantum collision search, while the results on $\mathsf{EtE_A}$-HCTR2 required a careful analysis of optimizing parameter trade-offs, involving multiple-collision search for large τ.

A Concrete vector for $\mathsf{EtE_P}$-HCTR2 with $\tau = 32$

Table 2 shows the concrete vectors generated in the experiment in Sect. 5.1 for breaking $\mathsf{CMT_k^*}$ security of $\mathsf{EtE_P}$-HCTR2 instantiated with AES-128 for $\tau = 32$.

References

1. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to Securely Release Unverified Plaintext in Authenticated Encryption. In: ASIACRYPT 2014, Part I. pp. 105–125. LNCS, Springer (2014)

2. Android Open Source Project: File-based encryption. https://source.android.com/docs/security/features/encryption/file-based (2023)
3. Bellare, M., Hoang, V.T.: Efficient Schemes for Committing Authenticated Encryption. In: EUROCRYPT 2022, Part II. pp. 845–875. LNCS, Springer (2022)
4. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: ASIACRYPT 2000. pp. 317–330. LNCS, Springer (2000)
5. Bonnetain, X., Loyer, J., Schrottenloher, A., Shen, Y.: A tight quantum algorithm for multiple collision search. Cryptology ePrint Archive, Paper 2025/1690 (2025), https://eprint.iacr.org/2025/1690
6. Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics **305**, 53–74 (2002)
7. Brassard, G., Høyer, P., Tapp, A.: Quantum Cryptanalysis of Hash and Claw-Free Functions. In: LATIN '98. pp. 163–169. LNCS, Springer (1998)
8. Chailloux, A., Naya-Plasencia, M., Schrottenloher, A.: An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography. In: ASIACRYPT 2017, Part II. pp. 211–240. LNCS, Springer (2017)
9. Chang, D., Chen, Y.L., Hiraga, Y., Minematsu, K., Mouha, N., Naito, Y., Sasaki, Y., Sugawara, T.: Key committing security of HCTR2, revisited. Cryptology ePrint Archive, Paper 2026/254 (2026), https://eprint.iacr.org/2026/254
10. Chen, Y.L., Davidson, M., Dworkin, M., Kelsey, J., Sasaki, Y., Turan, M.S., Chang, D., Mouha, N., Thompson, A.: Requirements for cryptographic accordions. NIST Interagency Report (NIST IR) 8552 (2025). https://doi.org/10.6028/NIST.IR.8552
11. Chen, Y.L., Flórez-Gutiérrez, A., Inoue, A., Ito, R., Iwata, T., Minematsu, K., Mouha, N., Naito, Y., Sibleyras, F., Todo, Y.: Key committing security of AEZ and more. IACR Trans. Symm. Cryptol. **2023**(4), 452–488 (2023). https://doi.org/10.46586/tosc.v2023.i4.452-488
12. Chen, Y.L., Hiraga, Y., Mouha, N., Naito, Y., Sasaki, Y., Sugawara, T.: Beyond-birthday-bound security with HCTR2: cascaded construction and tweak-based key derivation. In: Hanaoka, G., Yang, B. (eds.) ASIACRYPT 2025, Part I. LNCS, vol. 16245, pp. 3–34. Springer (2025)
13. Crowley, P., Huckleberry, N., Biggers, E.: Length-preserving encryption with HCTR2. Cryptology ePrint Archive, Report 2021/1441 (2021), https://eprint.iacr.org/2021/1441
14. Farshim, P., Orlandi, C., Roşie, R.: Security of symmetric primitives under incorrect usage of keys. IACR Trans. Symm. Cryptol. **2017**(1), 449–473 (2017). https://doi.org/10.13154/tosc.v2017.i1.449-473
15. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC. pp. 212–219. ACM (1996)
16. Gueron, S.: Intel advanced encryption standard (AES) new instructions set (2010)
17. Gueron, S., Kounavis, M.E.: Intel carry-less multiplication instruction and its usage for computing the GCM mode (2014)
18. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: Specification and analysis. Cryptology ePrint Archive, Paper 2017/168 (2017), https://eprint.iacr.org/2017/168
19. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: nonce misuse-resistant authenticated encryption. RFC **8452**, 1–42 (2019), https://doi.org/10.17487/RFC8452
20. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust Authenticated-Encryption AEZ and the Problem That It Solves. In: EUROCRYPT 2015, Part I. pp. 15–44. LNCS, Springer (2015)

21. Menda, S., Len, J., Grubbs, P., Ristenpart, T.: Context Discovery and Commitment Attacks - How to Break CCM, EAX, SIV, and More. In: EUROCRYPT 2023, Part IV. pp. 379–407. LNCS, Springer (2023)
22. National Institute of Standards and Technology: NIST launches development of cryptographic accordions. https://csrc.nist.gov/News/2025/nist-launches-development-cryptographic-accordions (2025)
23. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. pp. 373–390. LNCS, Springer (2006)
24. The Linux Kernel: hctr2.c. https://github.com/torvalds/linux/blob/master/crypto/hctr2.c, accessed: 2025-05-15
25. Wang, P., Feng, D., Wu, W.: HCTR: A variable-input-length enciphering mode. In: Information Security and Cryptology, First SKLOIS Conference, CISC 2005. LNCS, vol. 3822, pp. 175–188. Springer (2005)