Breaking, Repairing and Enhancing XCBv2 into the Tweakable Enciphering Mode GEM

Amit Singh Bhati[1,2], Michiel Verbauwhede[1] and Elena Andreeva[3]

[1] COSIC, KU Leuven, Belgium,
[2] 3MI Labs, Belgium,
[3] TU Wien, Austria
{amitsingh.bhati,michiel.verbauwhede}@esat.kuleuven.be
elena.andreeva@tuwien.ac.at

Abstract. Tweakable enciphering modes (TEMs) provide security in a variety of storage and space-critical applications like disk and file-based encryption, and packet-based communication protocols, among others. XCB-AES (known as XCBv2) is specified in the IEEE 1619.2 standard for encryption of sector-oriented storage media and it comes with a proof of security for block-aligned input messages.

In this work we demonstrate an attack on XCBv2. We show that XCBv2 is *insecure* also for full block messages by presenting a *partial* plaintext recovery attack using *only* two queries. We demonstrate that our attack further applies to the HCI and MXCB TEMs, which follow a similar design approach to XCBv2.

Following the responsible disclosure process, we communicated the attack details to IEEE and the authors[4] of XCB-AES. The authors have confirmed the validity of our attack on 02/09/2024.

In a recent work, Wang et al. also presented stronger attacks on all XCB variants that can recover *full* plaintext and proposed a fixed variant called XCB*. We highlight that our attack is not applicable on XCB*.

Our next contribution is to strengthen the provable security of XCB* (currently $n/3$ bits in queried blocks). We propose a new modular TEM called GEM which can be seen as a generalization of the Hash-CTR-Hash approach as used in XCB-style and HCTR-style TEMs. We are able to prove that GEM achieves full n-bit security using *only* n-bit PRP/PRF. We also give two concrete GEM instantiations: KohiNoor and DaryaiNoor, both of which are based on AES-128 and GHASH-256, and internally use variants of the CTR-based weak pseudorandom functions GCTR-3 and SoCTR, respectively. SoCTR uses AES-128 and GCTR-3 is based on ButterKnife-256. Our security proofs show that both KohiNoor and DaryaiNoor provide full n-bit security. From applications perspective, DaryaiNoor addresses the need for reusing classical components, while KohiNoor enhances performance by leveraging a more modern primitive based on the AES/Deoxys round function.

Our implementation demonstrate competitive performance: For typical 4KiB sector size, KohiNoor's performance is on par with AES_6-CTET+, yet achieving higher standard security guarantees. DaryaiNoor is on par

[4] Authors of both variants XCBv2 [28] and XCBv2fb [10].

with AES-CTET+ performance-wise while also maintaining higher security with standard components. Our GEM instances triple the security margin of XCB* and double that of HCTR2 at the cost of performance loss of only 12% (KohiNoor) and 68% (DaryaiNoor) for 4KiB messages.

Keywords: tweakable enciphering modes, VIL-STPRP, XCBv2, HCI, IEEE 1619.2, disk-sector encryption, GCTR, SoCTR

1 Introduction

An enciphering mode, also known as *length-preserving encryption*, is an encryption method whose ciphertext maintains the length of the plaintext and for a random but fixed key is a deterministic bijective map. Length-preserving encryption is sought for in systems where data storage and space are critical.

Tweakable enciphering modes (TEMs) process an extra public tweak input, often incorporating some public information like a counter or nonce (e.g., disk sectors or packet headers), to achieve flexibility and better security against adaptive chosen plaintext and ciphertext attacks. TEMs aim to provide a strong notion of security as strong tweakable pseudorandom permutations over variable-input-length vil-stprp inputs. TEMs can be seen as a generalization of strong tweakable block ciphers over arbitrary input space, and hence a secure vil-stprp. A vil-stprp secure TEM is also secure as a tweakable (wide) block cipher stprp. vil-stprp means that for an adversary with access to both the enciphering and deciphering and who controls the tweak and input, a TEM design should be indisntinguishable from a family of independent random permutations (indexed by the tweak and input length).

TEM Applications. TEMs are used in disk-sector encryption, key-wrap, swapfile and filebased encryption (FBE), packet-based communication and network protocols like TLS/SSL and IPsec that prioritize bandwidth efficiency. TEMs with fixed but large input length (256 bits or more) are known as wide (tweakable) block ciphers. Their efficient performance in processing large blocks of fixed-length data makes them particularly suitable for full disk encryption. If the tweak input of a TEM is ensured to be unique for every message, e.g. associated data, packet headers or packet numbers along with a counter or nonce, then a vil-stprp secure TEM is also an ind-cca (in the classical sense for randomized encryption) secure encryption scheme. In [22], it has been shown that the Encode-then-Encipher [4] scheme based on a vil-stprp scheme offers also *in-depth* or *robust* [6] security for authenticated encryption schemes. A few TEMs are also standardized and these include the IEEE 1619.2 (2010 and 2021) standard for encryption of sector-oriented storage media XCBv2 [28] (a.k.a. XCB-AES) and XTS [15] for disk-encryption. Still, these standards come with *brittle* security guarantees: XCBv2 was proven secure only for block-aligned messages and XTS is not proven secure even as an stprp enciphering scheme.
The importance of novel, candidate TEM designs was emphasized by the NIST (National Institute of Standards and Technology of US). In 2023 NIST held a

workshop whose objective was stated [29] as: NIST is "*particularly interested in discussing the possibility of standardizing a tweakable wide block encryption technique that could support a large range of input length*" with applications "*such as storage*" and "*security and efficiency of tweakable wide encryption techniques*".

TEM Requirements. A TEM aims to achieve the security notion of vil-stprp via a sound proof of security. A *beyond birthday bound* (BBB) security guarantees that even after $q \approx 2^{n/2}$ message blocks have been enciphered under the same key, where n is the size of the underlying primitive, the security of the TEM as vil-stprp is still attained. BBB security is an important requirement for present TEM applications as a mean of avoiding frequent rekeying (particularly relevant for large-data and long-term storage applications), and when used with standardized ciphers like AES, implying security beyond 2^{64} cipher evaluations for the same key.

An additional important consideration is the security model. TEMs like CTET+ manage to achieve BBB security ($2n/3$ bits), yet they do so *only* in the ideal cipher model and not in the standard one, which illustrates the non-triviality of the task. Yet, security in the *standard model* gives overall more realistic security guarantees and hence is a desirable goal for TEMs.

A device implementing a TEM benefits in performance, smaller hardware or software footprint from *optimal use* (as few as possible) of *inverse-free* underlying primitives. The latter means that the TEM makes only forward calls to its underlying primitive(s) for both enciphering and deciphering. If further the TEM enciphering and deciphering algorithms are *almost identical* (modifying the enciphering by a few instructions gives the deciphering), the device minimizes footprint and implementation size by avoiding storing the whole deciphering. A highly *parallelizable* TEM can significantly speed-up the performance, particularly on platforms with intrinsic hardware accelerators. AES or AES-based designs like Deoxys [24] are often the primitives of choice [11, 15, 23] due to the available, in-built AES-NI acceleration support on most platforms that perform length-preserving encryption.

TEM Designs. The design of well-optimized, secure and efficient TEMs has been a challenging problem, that has received a lot of attention in the last two decades [8, 9, 11, 12, 19–21, 23, 27, 28, 31]. Existing dedicated TEMs follow design strategies like the Feistel structure, Encrypt-Mix-Encrypt, Hash-ECB-Hash, and Hash-CTR-Hash to achieve various optimizations and trade-offs between security and performance. For example, the Encrypt-Mix-Encrypt approach was introduced and used by the EME design [21], and the AEZ-core [23] builds on the Feistel structure, and they both go into the direction of performance optimizations at birthday bound security. Hash-ECB-Hash has been used in the TET [20], HEH [31] and CTET+ [11] designs. CTET+ was proven $2n/3$-bit BBB secure in the ideal cipher model at the cost of some performance reduction. Note that for performance oprimization reasons AES_6-CTET+ reduces the AES block cipher use to only six out of 10 rounds to gain in performance, yet weakening the argument for instantiating CTET+ with an ideal primitive. XCB [27, 28], together with ZCZ [8], FAST [9], HCTR2 [12] and MXCB [30] follow the Hash-CTR-

Hash approach and aim at efficiency optimizations. These are generally possible through the application of efficient XOR-universal hash function and the parallelization support of counter mode CTR-based encryption. XCBv2 (a.k.a. XCB-AES in IEEE 1619.2) [28] and HCTR2 achieve some of the best performance results but require an inverse primitive call in deciphering, leading to distinct enciphering and deciphering cost and to the impossibility for instantiation with primitives that lack an inverse (e.g. a pseudorandom function). A notable exception in this group is the FAST TEM design which supports inverse-free deciphering and at the same time offers good performance. All of these Hash-CTR-Hash designs achieve up to $n/2$-bit security, except XCBv2 [28] which claims only $n/3$-bit security but is highly performant, and ZCZ* [8], which achieves full n-bit security but at the cost of highly reduced performance competitiveness (see ZCZ in [8, Table 2] and its main component ZHash under ZMAC in [14, Table 2] for details).

In Table 1, we summarize a selected number of TEMs with their design, security and efficiency properties. We additionally provide TEM performance comparisons in Table 2.

While the Hash-CTR-Hash TEM paradigm has enabled the design of efficient TEMs, combining the efficiency with well beyond birthday bound or full n-bit vil-stprp security, inverse-freeness, and almost identical enciphering and deciphering has remained a challenge that has not been successful so far.

Mode	Key Size	Primitive Assumption	Mode Security	Security Size	Inverse Free	Almost Identical Enc/Dec	Partial Block Support	Input Length Support
XTS	$2k$	sprp	✗	✗	✗	✗	✓	$\geq n$
EME*	$3k$	sprp	vil-stprp	$n/2$	✗	✗	✓	$\geq n$
AEZ-core	$3k$	prf	vil-stprp	$n/2$	✓	✓	✓	$\geq 2n$
TET	$2k$	sprp	stprp	$n/2$	✗	✗	✓	$\geq n$
HEH	$3n+k$	sprp	vil-stprp	$n/2$	✗	✗	✓	$\geq n$
CTET+	$5n+k$	IC	stprp	$2n/3$	✗	✗	✗	$\geq 2n$
HCH	$2n+k$	sprp	vil-stprp	$n/2$	✗	✗	✓	$\geq n$
ZCZ*	k	stprp	vil-sprp	n	✗	✗	✓	$\geq 2n$
FAST	$n+k$	prf	vil-stprp	$n/2$	✓	✓	✓	$\geq 2n$
HCTR2	$n+k$	sprp	vil-stprp	$n/2$	✗	✗	✓	$\geq n$
XCB*	$2n+3k$	sprp	vil-stprp	$n/3$	✗	✓	✗	$\geq n$
KohiNoor	$2n+3k$	prp, prf	vil-stprp	n	✓	✓	✓	$\geq 4n$
DaryaiNoor	$2n+4k$	prp	vil-stprp	n	✓	✓	✓	$\geq 4n$

Table 1: Comparison of Popular TEMs with KohiNoor and DaryaiNoor. Here n is the block size of the underlying primitive and IC denotes an ideal cipher. All variables here are in bits. XCB* [34] is a secure variant of XCB. For XCB*, *Poly* [33] is used here as the hash function.

Our Contribution. First, in our study of the IEEE 1619.2 standard XCBv2, which comes with two proofs of security [10, 28] for messages of block-aligned length, we unveil a surprisingly simple plaintext recovery attack. Our XCBv2 attack requires only two adversarial queries; one forward and one inverse, to break the currently claimed sprp $n/3$-bit security. We are able to also identify the weak spot, that is an implicit false assumption in both existing proofs of XCBv2. More specifically, it is assumed (implicitly) in both proofs that the sum of two linear XOR-universal hashes is a universal hash when the keys to both of the hashes are identical, a fact we show is untrue. Even further, we demonstrate that our attack also applies to an XCBv2 variant called MXCB [30] which is based on a vil-tprs (short for variable-input-length tweakable pseudorandom self-inverse permutation) HCI [30].

Attack Overview: Our proposed attack is a plaintext recovery attack under a chosen-ciphertext attack (CCA) model. For an m-bit ciphertext, the attack recovers the first $m-n$ bits of the corresponding plaintext. Here, m can be quite large, such as sector sizes typically used in disk encryption (e.g., 512B or 4KiB), and n represents the block size of the encryption primitive (16B for AES-128 in this example). Consequently, the attack is capable of recovering almost the entire sector.

Informally, the attack proceeds as follows: Given a target ciphertext C, the attacker performs two steps: 1. The attacker decrypts a related ciphertext $C' = C \oplus X$, where X is an arbitrary binary string of length $|C|$, with the last n bits set to 0. This results in a new message, M'. 2. The attacker then encrypts $M'' = M' \oplus X$, producing the ciphertext C''. Finally, the plaintext is recovered using the formula $M = C \oplus M'' \oplus C''$, where the first $m - n$ bits of M match the first $m - n$ bits of the target plaintext.

We note that a recent related work by Wang et al. [34] also provides plaintext recovery attacks on XCB variants breaking their STPRP security with three or more queries.[5] Building on their findings, we also generalize their Attack 2 (currently breaking the STPRP claim) to even break the basic SPRP claim of XCB variants and apply it to XCBv3 (our previous fix for XCBv2fb) to point out the exact flaw in its security proof in Appendix E.

In [34], the authors also proposed a fixed XCB variant called XCB* to restore usability through applying easy and least amount of changes i.e., adding two XORs and replacing the current hash functions with two independent XOR-universal hash functions. We highlight that our attack is not applicable on XCB*.

XCB* provably provides $\log(\epsilon)/2$ bits of security when used with ϵ-XOR-universal hash functions. For standard polynomial hashes like GHASH [26] that follow the structure of *Poly* [33] and have $\epsilon = \ell \cdot 2^{-n}$ (with ℓ being the input length in n-bit blocks), the concrete security of XCB* becomes $n/3$ bits.

The use of $\ell \cdot 2^{-n}$-XOR-universal hash function to generate the initialization value (IV) for the counter (CTR) mode introduces an extra variable ℓ in their security bounds and that reduces the security to a cubic bound. The ℓ variable cannot be trivially avoided to achieve BB or BBB security without making more substantial changes to the algorithms, such as post-processing the hash outputs with some weak pseudorandom function PRF before using them to generate the IV for the CTR mode.

[5] As evidenced by our responsible disclosure dates and earlier submissions of our work, our attack precedes the results in [34].

In order to increase the concrete security margin of XCB* to BBB, one would need to instantiate it with at least $2n$-bit sprp primitives. We show how the cost of securely designing and efficiently instantiating such large primitives can be circumvented. As our second main contribution, we relax the need of large-state and large-block primitive and propose a new generic n-bit secure vil-stprp TEM, which we call GEM. GEM improves further XCB* in terms of: 1) security margin – for an internal primitive with block size n, XCB* provides security *only* up to $n/3$ bits, whereas GEM provides full n-bit security; 2) primitive support – XCB* works only with efficiently invertible SPRPs, whereas GEM allows all PRPs (including the ones with inefficient inverse) and PRFs, hence a larger set of permissible instances; 3) small price in performance – GEM instances can incur a performance loss as little as 12% compared to XCB-AES.

GEM is a generalization of the Hash-CTR-Hash approach as used in XCB-style and HCTR-style TEMs. GEM relaxes the use of $2n$-bit sprp primitive to n-bit prp or prf primitive (in a two-round Feistel). GEM has three main components – a variable-input-length pseudorandom function vilF, a variable-output-length pseudorandom function volF and an n-to-n-bit block cipher or PRF E (see Figure 1). The two-round Feistel enables inverse-freeness and supports almost identical enciphering and deciphering. We also provide a solution for how to construct the variable-input-length vilF and variable-output-length volF pseudorandom functions components from any suitable weak pseudorandom function wPRF and an XOR-universal hash function. We formally prove the full n-bit vil-stprp security of GEM in the standard model and provide a detailed proof.

As our next contribution, we propose two GEM instances: KohiNoor and DaryaiNoor (see Table 1 for comparison with existing TEMs). KohiNoor and DaryaiNoor are based on AES-128 (128-bit secure prp) and GHASH-256 (XOR-universal hash) and use CTR-based weak pseudorandom functions: GCTR-3 [2] and Sum of CTR (SoCTR), respectively. SoCTR internally uses AES-128 and GCTR-3 is based on ButterKnife-256 [3]. We independently prove GCTR-3 and SoCTR secure as weak PRFs and then use these results to prove that KohiNoor and DaryaiNoor are each n-bit secure in the standard model. From applications perspective, DaryaiNoor answers the need of (re)use of classical components, while KohiNoor further pushes the performance based on the use of a more recent, yet AES/Deoxys round-function-based primitive.

	XCB-AES	HCTR2	AES_6-CTET+	AES-CTET+	KohiNoor	DaryaiNoor
512B	1.02	1.07	1.07	1.49	1.61	1.99
4 KiB	0.95	1.00	1.04	1.53	1.12	1.68

Table 2: Performance (in cycles/byte) comparison of XCB-AES, HCTR2, CTET+, KohiNoor and DaryaiNoor for 512B and 4096B sizes on the gracemont cove microarchitecture. XCB-AES, HCTR2, CTET+ and DaryaiNoor use AES-128.

Finally, we provide an implementation of our GEM instantiations and compare them against a representative number of TEMs following popular design strategies – XCB-AES, HCTR2 [12] (Hash-CTR-Hash), and CTET+ [11] (Hash-ECB-hash). On gracemont cove, for 4KiB messages, KohiNoor, which relies on the prf security of ButterKnife-256, outperforms AES_6-CTET+, which relies on the ideal cipher assumption of 6-round AES, while improving the security bounds from 86 to 128 bits in the standard versus ideal model. For 512B message lengths, KohiNoor is slower than AES_6-CTET+, but is still on par with the AES-CTET+. DaryaiNoor is slower than KohiNoor, but it only relies on the prp security of AES to achieve 128 bits of security. Performance-wise DaryaiNoor compares to AES-CTET+ for 4KiB messages whereas for 512B messages, AES-CTET+ is around 25% faster than DaryaiNoor. Compared to the lower security category of 43-bit secure XCB-AES and 64-bit secure HCTR2 TEMs, KohiNoor provides 128 bits of security with less than 18% performance loss at just 4 KiB length. At 4KiB length, DaryaiNoor shows the possibility to achieve 128 bit security with standard components without significantly hindering the performance. The details on our implementation and extended benchmarks are given in Appendix C.

Paper Organization. Preliminaries are covered in Section 2, followed by TEM security notions in Section 3. We present our attack on XCB-AES in Section 4. In Section 5, we introduce GEM mode as a modular TEM that fixes and improves XCB-AES. We then state its formal security, and defer the proof to Section 7. Section 6 presents two secure GEM instantiations, KohiNoor and DaryaiNoor. Full implementation details and benchmarking results for both GEM instantiations are provided in Appendix C.

2 Preliminaries

2.1 Notation

All strings used here are binary strings. Strings of length $n > 0$ are referred to as n-bit strings, and the set of all n-bit strings is denoted by $\{0,1\}^n$. The set of strings of any possible length is denoted as $\{0,1\}^*$. The set of all permutations of $\{0,1\}^n$ is denoted as $\mathrm{Perm}(n)$ and the set of all self-inverse permutations or involutions of $\{0,1\}^n$, is denoted by $\mathrm{SIPerm}(n)$, where $\mathrm{SIPerm}(n) \subset \mathrm{Perm}(n)$. The set of all functions/maps from $\{0,1\}^m$ (respectively, $\{0,1\}^*$) to $\{0,1\}^n$ is denoted by $\mathrm{Func}(m,n)$ (respectively, $\mathrm{Func}(*,n)$). For any string A, $|A|$ is the length of A in bits. For two strings $A, B \in \{0,1\}^*$ with (w.l.o.g.) $|A| \leq |B|$, we let $A \oplus B$ denote the bitwise XOR of $A \| 0^{|B|-|A|}$ and B, and define $A \oplus_a B = (A \oplus B)[0 \ldots a-1]$. We use $\mathsf{msb}_a(A)$ to denote the prefix string $A[0 \ldots a-1]$. For two a-bit strings X and Y, we denote their field multiplication in $\mathrm{GF}(2^a)$ (using a suitable, fixed irreducible polynomial) as $X \cdot Y$.

Given a string A and an integer n such that $|A| = cn+d$, where c is a positive integer and $0 < d \leq n$, the notation $A_1, A_2, \ldots, A_{c+1} \xleftarrow{n} A$ is used to indicate the partitioning of A into a maximum number of n-bit blocks. Each block A_i has a length of n for $1 \leq i \leq c$, and the last block A_{c+1} has a length of d. For a given n (as used in partitioning), we define an injective padding function for A as $\mathsf{pad}_n(A) = A_1 \| A_2 \| \ldots \| A_{c+1} \| 0^{n-d} \| \mathsf{bin}_n(|A|)$ where $\mathsf{bin}_n(|A|)$ denotes the n-bit binary representation of $|A|$. The notation $r \xleftarrow{\$} \mathcal{R}$ indicates the random sampling of an element r from a finite set $\mathcal{R}$ with a uniform distribution. We use lexicographic comparison for integer tuples (to exemplify, $(i', j') < (i, j)$ iff $i' < i$ or $i' = i$ and $j' < j$). The symbol $\perp$ is used to represent an undefined value or an error.

2.2 Definitions

In this section, we provide the syntax of a TEM followed by the standard definitions of relevant cryptographic primitives in this work. The security definitions for TEMs are provided in Section 3.

Definition 1 (Tweakable Enciphering Mode (TEM)). *A TEM scheme is defined as $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ which consists of three components: a key distribution $\mathcal{K}$, a deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{C}$ that preserves the length of the message, and a deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{T} \times \mathcal{C} \rightarrow \mathcal{M}$. The encryption algorithm takes a key K, a tweak T, and a message M from the sets $(K, T, M) \in \mathcal{K} \times \mathcal{T} \times \mathcal{M}$, and returns a ciphertext $C = \mathcal{E}(K, T, M) = \mathcal{E}_K(T, M)$. The decryption algorithm maps the key, tweak, and ciphertext to the original message using $M = \mathcal{D}(K, T, C) = \mathcal{D}_K(T, C)$. Here being length-preserving means that for any given key K, tweak T, and message M, $|\mathcal{E}_K(T, M)| = |M|$.*

Definition 2 (Universal Hashing). *Let $H : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^m$ for some non-negative k and m be a set of keyed hash functions. Let H_K denote H with a k-bit key K. H is called ϵ-XOR-universal if for any given two distinct inputs $X, X' \in \{0,1\}^*$ and any output $Y \in \{0,1\}^m$,*

$$\Pr[H_K(X) \oplus H_K(X') = Y] \leq \epsilon,$$

where the probability is computed over K chosen uniformly at random from $\{0,1\}^k$. Further, when Y is fixed to 0^m, H is called ϵ-universal.

Definition 3 (Polynomial Hashing). *A polynomial hash $H : \{0,1\}^m \times \{0,1\}^* \rightarrow \{0,1\}^m$ for some non-negative m is a set of keyed hash functions. For any given inputs $(K, X) \in \{0,1\}^m \times \{0,1\}^*$ with $P_1, P_2, \ldots, P_c \xleftarrow{m} \mathsf{pad}_m(X)$ for some positive c, we define two polynomial hashes $\mathsf{H1}$ and $\mathsf{H2}$ as follows*

$$\mathsf{H1}(K, X) = \mathsf{H1}_K(X) = K^{c-1} \cdot P_1 \oplus K^{c-2} \cdot P_2 \oplus \ldots \oplus K \cdot P_{c-1} \oplus P_c\,,$$
$$\mathsf{H2}(K, X) = \mathsf{H2}_K(X) = K \cdot \mathsf{H1}_K(X)\,.$$

$\mathsf{H1}$ and $\mathsf{H2}$ are $\ell \cdot 2^{-m}$-universal and $\ell \cdot 2^{-m}$-XOR-universal hash functions when restricted to messages with post-padding length $m \cdot \ell$, i.e. setting $c = \ell$ in Definition 3 (for detailed discussion on this, see [33]).

Definition 4 (Block Cipher). *A block cipher* $\mathsf{E} : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is a function that is a permutation for every* $K \in \{0,1\}^k$. *We define the advantage of a distinguisher* $\mathcal{D}$ *in distinguishing* E *from a random permutation* $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ *as*

$$\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\mathcal{D}) = \left|\Pr\left[K \xleftarrow{\$} \{0,1\}^k : \mathcal{D}^{\mathsf{E}_K(\cdot)} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}(n) : \mathcal{D}^{\pi(\cdot)} \Rightarrow 1\right]\right|.$$

Note that for prp (pseudorandom permutation) security advantage, $\mathcal{D}$ only gets forward access to E.

Definition 5 (Pseudorandom Function (PRF)). *A* PRF *function* $\mathsf{F} : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ *is a function that takes a key* $K \in \{0,1\}^k$ *and a message* $X \in \{0,1\}^m$ *as input and returns an output* $Y \in \{0,1\}^n$ *for some positive integers* m *and* n. *We define the advantage of a distinguisher* $\mathcal{D}$ *in distinguishing* F *from a random function* $f \xleftarrow{\$} \mathrm{Func}(m,n)$ *as*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{D}) = \left|\Pr\left[K \xleftarrow{\$} \{0,1\}^k : \mathcal{D}^{\mathsf{F}_K(\cdot)} \Rightarrow 1\right] - \Pr\left[f \xleftarrow{\$} \mathrm{Func}(m,n) : \mathcal{D}^{f(\cdot)} \Rightarrow 1\right]\right|.$$

If m is not fixed i.e., when $X \in \{0,1\}^*$, we call F a variable input length pseudorandom function ($\mathsf{vil\text{-}PRF}$). Similarly, if n is not fixed i.e., when $Y \in \{0,1\}^*$, we call F a variable-output-length PRF. We note that uniform sampling cannot be defined on $\mathrm{Func}(m,*)$, therefore, we consider $f' \xleftarrow{\$} \mathrm{Func}(m, n_{\mathsf{max}})$ for some large fixed n_{max} such that for all variable ns, $n_{\mathsf{max}} \geq n$. We then define $f(\cdot) = f'(\cdot)[0 \ldots n-1]$. We highlight that for such f, any two queries with same input but different ns will have one output as prefix of the other. We also note that there are other ways to sample random fs that can avoid this property. However, in this work, we consider this sampling as the ideal case.

Definition 6 (Weak Pseudorandom Function (wPRF)). *A* wPRF *function* $\mathsf{F} : \{0,1\}^{k_\mathsf{F}} \times \{0,1\}^m \rightarrow \{0,1\}^n$ *is a function that takes a key* $K \in \{0,1\}^{k_\mathsf{F}}$ *and a message* $X \in \{0,1\}^m$ *as input and returns an output* $Y \in \{0,1\}^n$ *for some positive integers* m *and* n. *We define the advantage of a distinguisher* $\mathcal{D}$ *in distinguishing* F *from a random function* $f \xleftarrow{\$} \mathrm{Func}(m,n)$ *under transformed inputs using a keyed XOR-universal hash function* $\mathsf{H} : \{0,1\}^{k_\mathsf{H}} \times \{0,1\}^* \rightarrow \{0,1\}^m$ *as*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{F},\mathsf{H}}(\mathcal{D}) = \Big| &\Pr\left[K_1 \xleftarrow{\$} \{0,1\}^{k_\mathsf{F}}, K_2 \xleftarrow{\$} \{0,1\}^{k_\mathsf{H}} : \mathcal{D}^{\mathsf{F}_{K_1}(\mathsf{H}_{K_2}(\cdot))} \Rightarrow 1\right] \\ &- \Pr\left[f \xleftarrow{\$} \mathrm{Func}(m,n), K_2 \xleftarrow{\$} \{0,1\}^{k_\mathsf{H}} : \mathcal{D}^{f(\mathsf{H}_{K_2}(\cdot))} \Rightarrow 1\right]\Big|.\end{aligned}$$

For simplicity, we keep H implicit and drop the subscript H, denoting the advantage by $\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{F}}(\mathcal{D})$. Further, when n is not fixed, we sample f in a similar manner to the variable-output-length pseudorandom function discussed earlier.

3 Security Notions for Enciphering Schemes

In this section, we introduce a unified notion for enciphering schemes which incorporates popular TEM notions as its special cases.

Let $\Pi = (\mathcal{K}, \mathsf{TEM}, \mathsf{TEM}^{-1})$ be a TEM scheme with key space $\mathcal{K}$, tweak space $\mathcal{T}$ and message space $\mathcal{M} = \cup_{i \in \mathcal{L}} \{0,1\}^i$ for some set $\mathcal{L}$ of all allowed message lengths. Let $\mathcal{T}^\star = \mathcal{T}$ when $|\mathcal{T}| \neq 0$ and $\mathcal{T}^\star = \{0\}$, otherwise. Consider an adversary $\mathcal{A}$ whose goal for a given $\alpha \in \{0,1\}$ is to distinguish between the functions $(\mathsf{TEM}_K, \alpha\mathsf{TEM}_K^{-1})$ with $K \xleftarrow{\$} \mathcal{K}$ and a randomly sampled permutation family $(\pi_{\mathcal{T},\mathcal{L}}, \alpha\pi_{\mathcal{T},\mathcal{L}}^{-1})$ by making oracle queries to them where

$$(\pi_{\mathcal{T},\mathcal{L}}, \pi_{\mathcal{T},\mathcal{L}}^{-1}) = \cup_{i \in \mathcal{L}} \{\{(\pi_{T,i}, \pi_{T,i}^{-1})\}_{T \in \mathcal{T}^\star} \subseteq^{\$} \mathcal{F}_i\}$$

i.e. for each distinct tweak $T \in \mathcal{T}^\star$ or message length $i \in \mathcal{L}$, the corresponding permutation $\pi_{T,i}$ is uniformly sampled at random from some set $\mathcal{F}_i \subseteq \mathrm{Perm}(i)$ with $\pi_{T,i}^{-1}$ defining the inverse map of $\pi_{T,i}$. Here the oracle $\alpha\pi_{T,i}^{-1}(\cdot,\cdot)$ returns the output of $\pi_{T,i}^{-1}$ when $\alpha = 1$ and the empty string, otherwise. The advantage of $\mathcal{A}$ in breaking the type-security of Π is defined as follows:

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathsf{type}}(\mathcal{A}) = \big| &\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{TEM}_K(\cdot,\cdot), \alpha\mathsf{TEM}_K^{-1}(\cdot,\cdot)} \Rightarrow 1] \\ &- \Pr[\cup_{i \in \mathcal{L}} \{(\pi_{\mathcal{T},i}, \pi_{\mathcal{T},i}^{-1}) \subseteq^{\$} \mathcal{F}_i\} : \mathcal{A}^{\pi_{\mathcal{T},\mathcal{L}}(\cdot,\cdot), \alpha\pi_{\mathcal{T},\mathcal{L}}^{-1}(\cdot,\cdot)} \Rightarrow 1]\big|,\end{aligned}$$

where for any T, M and C, $\pi_{\mathcal{T},\mathcal{L}}(T, M) = \pi_{T,|M|}(M)$ and $\pi_{\mathcal{T},\mathcal{L}}^{-1}(T, C) = \pi_{T,|C|}^{-1}(C)$.

Now that the generic definition of type is defined, we recall all the existing TEM security notions of (variable-input-length) (strong) (tweakable) pseudorandom permutation ((vil-)(s)(t)prp) and (variable-input-length) (tweakable) pseudorandom self-inverse permutation ((vil-)(t)prs) as special cases of it in Table 3.

<table>
<tr><th>type</th><th>prp</th><th>sprp</th><th>tprp</th><th>stprp</th><th>vil-prp</th><th>vil-sprp</th><th>vil-tprp</th><th>vil-stprp</th><th>prs</th><th>tprs</th><th>vil-prs</th><th>vil-tprs</th></tr>
<tr><td>α</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td colspan="4">0</td></tr>
<tr><td>$|\mathcal{T}|$</td><td colspan="2">≤ 1</td><td colspan="2">≥ 2</td><td colspan="2">≤ 1</td><td colspan="2">≥ 2</td><td>≤ 1</td><td>≥ 2</td><td>≤ 1</td><td>≥ 2</td></tr>
<tr><td>$|\mathcal{L}|$</td><td colspan="4">1</td><td colspan="4">≥ 2</td><td colspan="2">1</td><td colspan="2">≥ 2</td></tr>
<tr><td>$\mathcal{F}_i$</td><td colspan="8">$\mathrm{Perm}(i)$</td><td colspan="4">$\mathrm{SIPerm}(i)$</td></tr>
</table>

Table 3: Comparison of various TEM or wide block enciphering security notions. Here α is an indicator function that when 1 allows the inverse oracle access to the adversary and restricts otherwise. $|\mathcal{T}| \leq 1$ means that the targeted TEM is not tweakable or the tweak is fixed for the adversary. $|\mathcal{L}| = 1$ means that the targeted TEM is a fixed-input-length function and variable-input-length, otherwise.

4 Revisiting XCBv2 and HCI

XCB (short for eXtended Code Book) is a popular TEM proposed by McGrew and Fluhrer in 2004 [27]. XCB came originnally with no formal security proof.

In 2007 [28], the authors revised XCB (XCBv1 to XCBv2) and provided a proof for the updated version. However, in 2013, Chakraborty et al. [10] examined both versions and revealed that the security claims for XCBv2 were flawed, particularly for messages not aligning with the block cipher's block length. They demonstrated this through a distinguishing attack. In the same paper, the authors gave a new proof for XCBv1 and XCBv2 (but restricted to messages that align with block cipher's block length a.k.a. full block messages). XCBv2, which is also referred to as XCB-AES, is part of the IEEE 1619.2 (2010 and 2021) standard for encryption of sector-oriented storage media.

Here we show that even for full block messages, XCBv2 is not sprp secure which implies $\mathsf{vil\text{-}stprp}$ insecurity. We demonstrate a partial plaintext recovery attack with only two queries – one forward and one inverse call to XCBv2. We now recall the definition of XCBv2 [28]. We note that XCBv2 is already insecure when the underlying messages are not full-blocks [10]. The same attack argument applies when the tweak is not a full-block. Therefore, w.l.o.g. we consider only full-block messages and tweaks for the description of XCBv2.

Further, we note that the hash function as defined in [28] takes (excluding the key) two arguments - the tweak and the input that are individually padded and then concatenated into one string for polynomial hashing. However, since we consider only full-block tweaks, we can avoid tweak padding and thus can use a hash that takes only one argument - tweak concatenated with the input (see Def. 3 for such one argument hash).

4.1 XCBv2 Mode with Full-block Messages and Tweaks

We refer to XCBv2 [28] with full-block messages and tweaks as XCBv2fb (see [10]). XCBv2fb consists of two algorithms: encipher and decipher. The encipher algorithm takes a k-bit key K, a tweak $T \in \{0,1\}^{(t-1)n}$ and a message $M \in \{0,1\}^{mn}$ with $m, t \geq 1$, $M = M_L \| M_R$ and $|M_R| = n$. It then uses K to derive an n-bit key K_h and three k-bit keys K_e, K_d and K_c as

$$K_h = E_K(\mathsf{bin}_n(0)), K_e = \mathsf{msb}_k(E_K(\mathsf{bin}_n(1)) \| E_K(\mathsf{bin}_n(2))),$$
$$K_d = \mathsf{msb}_k(E_K(\mathsf{bin}_n(3)) \| E_K(\mathsf{bin}_n(4))), K_c = \mathsf{msb}_k(E_K(\mathsf{bin}_n(5)) \| E_K(\mathsf{bin}_n(6))) .$$

It uses a keyed hash function $H_{K_h} : \{0,1\}^{(m+t)n} \to \{0,1\}^n$, a counter mode (IV is the first n-bit input) $\mathsf{CTR}_{K_c} : \{0,1\}^n \times \{0,1\}^{(m-1)n} \to \{0,1\}^{(m-1)n}$ and two keyed block ciphers $E_{K_e} : \{0,1\}^n \to \{0,1\}^n$ and $E_{K_d} : \{0,1\}^n \to \{0,1\}^n$ (inverse calls denoted as $E^{-1}_{K_e}$ and $E^{-1}_{K_d}$) to compute:

$$C_L = \mathsf{CTR}_{K_c}(E_{K_e}(M_R) \oplus H_{K_h}(0^n \| T \| M_L \| 0^n), M_L)$$
$$C_R = E^{-1}_{K_d}(E_{K_e}(M_R) \oplus H_{K_h}(0^n \| T \| M_L \| 0^n) \oplus H_{K_h}(T \| 0^n \| C_L \| \mathsf{len}_{m,t}))$$

and returns C as $C_L \| C_R$. Here $\mathsf{len}_{m,t} = \mathsf{bin}_{n/2}(|T \| 0^n|) \| \mathsf{bin}_{n/2}(|C_L|)$ and the counter mode is defined using the block cipher E as

$$\begin{aligned}\mathsf{CTR}_{K_c}(IV, P) :=& (E_{K_c}(IV) \| E_{K_c}(IV \oplus \mathsf{bin}_n(1)) \| \cdots \\ & \cdots \| E_{K_c}(IV \oplus \mathsf{bin}_n(\lceil |P|/n \rceil - 1))) \oplus_{|P|} P .\end{aligned}$$

The XCBv2fb decipher algorithm swaps the block cipher keys and hash inputs' final blocks. For the same keys and a ciphertext C, it is defined as:

$$M_L = \mathsf{CTR}_{K_c}(E_{K_d}(C_R) \oplus H_{K_h}(T \| 0^n \| C_L \| \mathsf{len}_{m,t}), C_L)$$
$$M_R = E^{-1}_{K_e}(E_{K_d}(C_R) \oplus H_{K_h}(T \| 0^n \| C_L \| \mathsf{len}_{m,t}) \oplus H_{K_h}(0^n \| T \| M_L \| 0^n))$$

and returns the plaintext $M_L \| M_R$. For ease of understanding, a pictorial diagram for the enciphering of XCBv2fb is also provided in Appendix A.

Observations. For all valid IV and P it holds that $\mathsf{CTR}_{K_c}(IV, P) = \mathsf{CTR}_{K_c}(IV, 0^{|P|}) \oplus P$. Thus, from Fig. 2 the following equations can be extracted:

$$Z = X_L \oplus X_R = H_{K_h}(0^n \| T \| M_L \| 0^n) \oplus E_{K_e}(M_R)\,, \tag{1}$$
$$Z = Y_L \oplus Y_R = H_{K_h}(T \| 0^n \| C_L \| \mathsf{len}_{m,t}) \oplus E_{K_d}(C_R)\,, \tag{2}$$
$$M_L \oplus C_L = \mathsf{CTR}_{K_c}(Z, 0^{|M_L|})\,. \tag{3}$$

H_{K_h} in XCBv2fb is instantiated with the polynomial hash $\mathsf{H2}$ (Def. 3) as an XOR-universal hash function. We note that $\mathsf{H2}$ is a linear function and has the following relation for any value of A, B and D when $K_1 = K_2$:

$$\begin{aligned}\mathsf{H2}_{K_1}(A \oplus D) \oplus \mathsf{H2}_{K_2}(B) &= (\mathsf{H2}_{K_1}(A) \oplus \mathsf{H2}_{K_1}(D)) \oplus \mathsf{H2}_{K_2}(B)\\ &= \mathsf{H2}_{K_1}(A) \oplus (\mathsf{H2}_{K_2}(B) \oplus (\mathsf{H2}_{K_1}(D))\\ &= \mathsf{H2}_{K_1}(A) \oplus (\mathsf{H2}_{K_2}(B) \oplus (\mathsf{H2}_{K_2}(D))\\ &= \mathsf{H2}_{K_1}(A) \oplus \mathsf{H2}_{K_2}(B \oplus D)\,. \end{aligned} \tag{4}$$

This relation shows that *the sum of two linear XOR-universal hashes is not a universal hash when the keys to both of the hashes are same.* In AE modes like GCM [26] and GCM-SIV [18], the same hash key is used over multiple hash calls, however, they mask the output of each hash with a fresh random value and thus restrict the appearance of the relation or the exposure of the hash outputs.

XCBv2fb also tries to do the same by post-processing the hash outputs with a block cipher and then releasing it as the output. But in XCBv2fb a sum of hashes is already performed before this post-processing (see Eqn. 2), our simple two query attack shows that the above relation can *still* be verified and used to partially recover the plaintext for any given ciphertext.

Since for our attack, the key of XCBv2fb, length of the messages (mn bits for some positive integers m and n with $m \geq 2$) and the tweaks are kept identical over the queries, we abuse the notation to denote the $E_{K_e}(\cdot)$, $E_{K_d}(\cdot)$, $H_{K_h}(\cdot)$ and $\mathsf{CTR}_{K_c}(\cdot, 0^{mn-n})$ functions by $\mathsf{e}(\cdot), \mathsf{d}(\cdot)$, $\mathsf{h}(\cdot)$ and $\mathsf{c}(\cdot)$, respectively.

4.2 Plaintext Recovery Chosen Ciphertext Attack on XCBv2fb

Our XCBv2fb attack uses one enciphering and one deciphering call. Let $\mathcal{A}$ be an adversary against XCBv2fb who aims to partially recover the plaintext for a given ciphertext $C \in \{0,1\}^{mn}$ and tweak $T \in \{0,1\}^{(t-1)n}$.

Shared Difference Attack. $\mathcal{A}$ splits C as $C_L \| C_R$ where $|C_R| = n$, chooses a non-zero constant $S \in \{0,1\}^{mn-n}$, and makes two queries to XCBv2fb:

1. Query $(C_L \oplus S)\|C_R$ with tweak T to decipher and receive $M'_L\|M'_R$.
2. Query $(M'_L \oplus S)\|M'_R$ with tweak T to encipher and receive $C'_L\|C'_R$.

$\mathcal{A}$ returns the partially recovered left $(mn-n)$ bits of the target plaintext M as $M_L = C_L \oplus M'_L \oplus C'_L \oplus S$.

Attack Correctness. For the target ciphertext $C = C_L\|C_R$ and its corresponding secret plaintext $M = M_L\|M_R$, the relation from Eqn. 2 and 3 holds:

$$M_L \oplus C_L = \mathsf{c}(\mathsf{h}(T\|0^n\|C_L\|\mathsf{len}_{m,t}) \oplus \mathsf{d}(C_R))\,. \tag{5}$$

Similarly, using Eqn. 1 and 3 over the input and output of the encipher query made by $\mathcal{A}$, we get the following relation,

$$(M'_L \oplus S) \oplus C'_L = \mathsf{c}(\mathsf{h}(0^n\|T\|(M'_L \oplus S)\|0^n) \oplus \mathsf{e}(M'_R))\,, \tag{6}$$

Then, from the input and output of deciphering and Eqn. 1 and 2, we get

$$\mathsf{h}(T\|0^n\|(C_L \oplus S)\|\mathsf{len}_{m,t}) \oplus \mathsf{d}(C_R) = \mathsf{h}(0^n\|T\|M'_L\|0^n) \oplus \mathsf{e}(M'_R)\,.$$

Recall that h in XCBv2fb is instantiated by $\mathsf{H2}$ and thus, we can add $\mathsf{h}(0^{tn}\|S\|0^n)$ on both side of the equation and use the relation of Exp. 4 to get:

$$\mathsf{h}(T\|0^n\|C_L\|\mathsf{len}_{m,t}) \oplus \mathsf{d}(C_R) = \mathsf{h}(0^n\|T\|(M'_L \oplus S)\|0^n) \oplus \mathsf{e}(M'_R)\,. \tag{7}$$

Combining, Eqn. 5, 6 and 7, gives us $M_L \oplus C_L = M'_L \oplus S \oplus C'_L$ and hence allows $\mathcal{A}$ to successfully recover M_L (i.e., all but last n bits of M) within two oracle queries and with probability 1.

4.3 Flaw in the Existing Analyses

Both existing XCBv2fb analyses use H as a linear XOR-universal hash function (Theorem 1 in [28] and Lemma 1 in [10]), which as our attack shows, is not sufficient for the claimed sprp security. The main flaw lies in the implicit assumption that the XOR of H on two different inputs $X_1 \neq X_2$ is also a universal hash and thus can be used to generate an unpredictable IV for the CTR mode. More specifically, let $\mathsf{XH}_{K,K}(X_1, X_2) = H_K(X_1) \oplus H_K(X_2)$ with H as an ϵ-XOR-universal then for the CTR IV generating part, it is assumed that the probability of finding a collision in XH outputs for any two distinct inputs but same key K (sampled freshly and uniformly at random) is reasonably small. This is not true as for all Δ we have $\mathsf{XH}_{K,K}(X_1, X_2) = \mathsf{XH}_{K,K}(X_1 \oplus \Delta, X_2 \oplus \Delta)$ with probability 1.

Other Applications of Shared Difference Attack. Our attack is also valid for the XCB-style, variable-input-length tweakable pseudorandom involution ($\mathsf{vil\text{-}tprs}$) HCI [30] TEM and the $\mathsf{vil\text{-}stprp}$ secure variant of it MXCB [30], and breaks their $\mathsf{vil\text{-}tprs}$ and $\mathsf{vil\text{-}stprp}$ security claims, respectively.

HCI differs from XCBv2 in: 1) it uses same K_b for both block cipher calls; 2) its left-right partitioning of the inputs is switched i.e., for any $M = M_L\|M_R$, now $|M_L| = n$, and 3) it has the same padding for both hash inputs, i.e., both hash functions take inputs $\mathsf{pad}_n(T)\|\mathsf{pad}_n(X)$ where T is the tweak and X is the right partition of plaintext or ciphertext depending on the hash call. For a full HCI description, we refer to [30]. While our attack applies directly, no easy fix seems to allow HCI to regain its $\mathsf{vil\text{-}tprs}$ security without losing its involution (self-inverse) property. Keeping the involution property requires the use of same hash and block cipher keys, which means the XCBv2fb fix cannot be applied.

MXCB [30] is an XCB TEM variant that generically constructs a $\mathsf{vil\text{-}stprp}$ mode from a $\mathsf{vil\text{-}tprs}$ building block (see [30]). MXCB is defined as HCI but with masked first (n-bit) output block using an XOR with an extra key. Its security relies on the $\mathsf{vil\text{-}tprs}$ HCI proof. Our attack invalidates the existing lower bound on the MXCB $\mathsf{vil\text{-}stprp}$ security. However, we believe that a dedicated proof similar to XCBv2 [10] can still be valid for a modified version where not only the first but also the last output n-bit block is masked with the same key. Such masking prohibits the adversary to see the actual left or right partitions of underlying HCI's outputs (i.e., C_L or C_R) and thus avoids the attack.

Inapplicability to HCTR-based TEMs. Our attack does not apply to HCTR-style TEMs such as FAST [9] and HCTR2 [12] due to the middle block cipher that additionally encrypts/decrypts one of the hash outputs before XORing it with the other hash output, and hence restricts the verification of Eqn. 4.

Inapplicability to XCB* [34]. Our attack does not apply to XCB* due to its PIV [32]-style input processing that makes the block cipher outputs dependent of the XOR-universal hash function outputs and hence restricts a relation like Eqn. 4 to appear.

5 GEM: A Generic Enciphering Mode

In this section, we propose a new generic TEM design GEM that improves upon XCB*. GEM is based on a variable-input-length PRF vilF, a variable-output length PRF volF and an n-to-n-bit function E that can be either a block cipher or a PRF. The GEM structure and support of primitives can be seen as a generalization of the Cipher-Hash-CTR-Hash-Cipher approach that is used in XCB-style TEMs. GEM relaxes the "Cipher" call requirement from $2n$-bit sprp primitive to n-bit PRP or PRF (in a two-round Feistel). The generic nature of GEM allows for different optimizations under suitable components and achieves all desirable TEM properties (see Table 1, 4 and Section 6). Additionally, the design modularity helps in achieving provable security results with component-wise separated bad-case analyses that are simpler and thus less error-prone.

We now define some relevant conventions. Let λ be the target $\mathsf{vil\text{-}stprp}$ security size (in bits). Then, for GEM, we need that the output size of the vilF and both input and output sizes of the volF are equal to 2λ bits each (to avoid trivial birthday attacks). This means that the minimum size of message input to GEM modes is 4λ bits. For simplicity, we fix λ to n (the block size of the underlying E) and hence set a target of n-bit security for the GEM mode.

GEM Design. GEM is a tweakable enciphering mode that takes in a tweak $T \in \{0,1\}^*$, a key $K = K_1 \| K_2 \| K_3 \in \{0,1\}^*$ with $|K_1| = 2k_{\mathsf{bc}}$, $|K_2| = k_{\mathsf{vil}}$ and $|K_3| = k_{\mathsf{vol}}$ for some non-negative integers $k_{\mathsf{bc}}, k_{\mathsf{vil}}$ and k_{vol}, and a message $M \in \{0,1\}^*$ with $M = M_L \| M_R$ where $|M_R| \geq |M_L| = 2n$. It then uses the variable-input-length PRF $\mathsf{vilF} : \{0,1\}^{k_{\mathsf{vil}}} \times \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^{2n}$, the variable-output-length PRF $\mathsf{volF} : \{0,1\}^{k_{\mathsf{vol}}} \times \{0,1\}^{2n} \rightarrow \{0,1\}^*$, and the function $\mathsf{E} : \{0,1\}^{k_{\mathsf{bc}}} \times \{0,1\}^n \rightarrow \{0,1\}^n$ as shown in Figure 1 and returns a ciphertext C as $C_L \| C_R$ with $|C_R| \geq |C_L| = 2n$ and $|C| = |M|$.

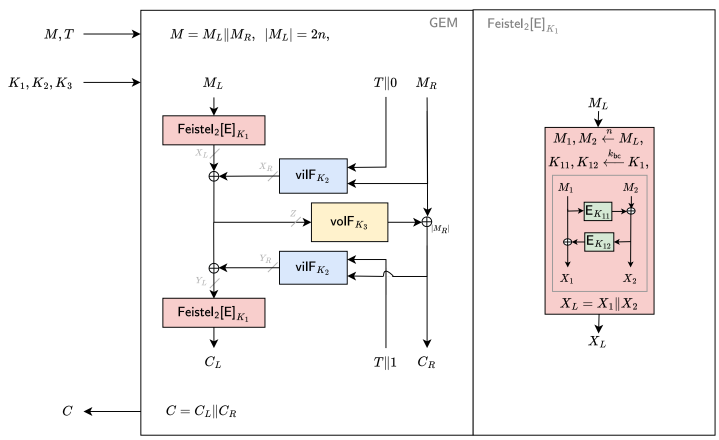

Fig. 1: Generic Enciphering Mode (GEM).

The volF in GEM can return arbitrary large outputs, however, for the XOR followed by the volF call, we need only $|M_R|$ many bits. Hence, the output of volF is truncated to the first $|M_R|$ bits and then XORed with M_R (denoted by $\oplus_{|M_R|}$ in Figure 1). The deciphering of GEM is defined in the same way as its enciphering but with $M, T\|0$ and $\mathsf{Feistel}_2$ swapped with $C, T\|1$ and $\mathsf{Feistel}_2^{-1}$, respectively in Figure 1. For simplicity, we keep the key inputs of $\mathsf{vilF}, \mathsf{volF}$ and E implicit and denote them with $\mathsf{vilF}_A, \mathsf{volF}_B$ and E_D for their keys A, B and D, respectively. Further, we keep the $\mathsf{vilF}, \mathsf{volF}$ and E components of GEM as parameters to its function and denote an instantiation of it by GEM[$\mathsf{vilF}, \mathsf{volF}, \mathsf{E}$].

5.1 Security of GEM

Theorem 1 states the $\mathsf{vil\text{-}stprp}$ security of GEM[$\mathsf{vilF}, \mathsf{volF}, \mathsf{E}$] and its proof is deferred to Section 7.1.

Theorem 1. *Let* GEM[$\mathsf{vilF}, \mathsf{volF}, \mathsf{E}$] *be the TEM as defined above then for any adversary $\mathcal{A}$ who makes at most q_e enciphering and q_d deciphering queries to* GEM *with input lengths $\geq 4n$ bits and $q = q_e + q_d \leq 2^{n-2}$, we have*

$$\mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[\mathsf{vilF},\mathsf{volF},\mathsf{E}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{vilF}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{volF}}(\mathcal{C}) + 2\,\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\mathcal{D}) + \frac{11q^2}{2^{2n}},$$

$$\mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[\mathsf{vilF},\mathsf{volF},\mathsf{E}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{vilF}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{volF}}(\mathcal{C}) + 2\,\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{E}}(\mathcal{D}) + \frac{4q^2}{2^{2n}}$$

for some adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ making at most $2q$ many vilF*, $2q$ many* E *and q many* volF *queries (under fixed but secret, random and independent keys), respectively, and running in time given by the running time of $\mathcal{A}$ plus $\gamma_0 \cdot q$ for some "small" constant γ_0.*

5.2 Single Solution for Both vilF and volF

We propose an efficient way to instantiate both vilF and volF using an XOR-universal hash function and a weakly secure PRF. To define those, we fix $k_{\mathsf{vil}} = k_{\mathsf{vol}} = k_{\mathsf{h}} + k_{\mathsf{wk}}$ for some non-negative integers k_{h} and k_{wk}. Let for all positive integers ℓ, $H : \{0,1\}^{k_{\mathsf{h}}} \times \{0,1\}^{2\ell n} \rightarrow \{0,1\}^{2n}$ be an ϵ_ℓ-XOR-universal hash function with block aligned messages and $\mathsf{wPRF} : \{0,1\}^{k_{\mathsf{wk}}} \times \{0,1\}^{2n} \rightarrow \{0,1\}^*$ be a wk-prf-secure PRF. For GEM, we now define

$$\mathsf{vilF}_{K_2}(A,B) = \mathsf{wPRF}_{K_{22}}(H_{K_{21}}(\mathsf{pad}_{2n}(A)\|\mathsf{pad}_{2n}(B))),$$
$$\mathsf{volF}_{K_3}(C) = \mathsf{wPRF}_{K_{32}}(H_{K_{31}}(C))$$

for any $A, B \in \{0,1\}^*$ with $|A| \geq 1$ and $|B| \geq 2n$, $C \in \{0,1\}^{2n}$, $K_{21}, K_{31} \in \{0,1\}^{k_{\mathsf{h}}}$, and $K_{22}, K_{32} \in \{0,1\}^{k_{\mathsf{wk}}}$ with $K_2 = K_{21}\|K_{22}$ and $K_3 = K_{31}\|K_{32}$.

Observation. We highlight the following detail that in GEM, all vilF inputs satisfy $|A| \geq 1$ and $|B| \geq 2n$ (in GEM, A represents $T\|0$s or $T\|1$s whereas B represents M_Rs or C_Rs), therefore, we have $|\mathsf{pad}_{2n}(A)\|\mathsf{pad}_{2n}(B)| \geq 8n$. On the other hand, all volF inputs have $|C| = 2n$ i.e., the input domains of vilF and volF are different. Even when $K_2 = K_3$ i.e., both vilF and volF are keyed with same hash and wPRF keys, their outputs are independent and distributed uniformly as long as there is no internal hash output collision.

Let $\mathcal{X}_{\mathsf{vil}}$ be the set of all binary strings of the form $\mathsf{pad}_{2n}(A)\|\mathsf{pad}_{2n}(B)$ for arbitrary strings A and B satisfying $|A| \geq 1$ and $|B| \geq 2n$. Let $\mathcal{X}_{\mathsf{vol}} = \{0,1\}^{2n}$ and $\mathcal{X}_H = \mathcal{X}_{\mathsf{vil}} \cup \mathcal{X}_{\mathsf{vol}}$. We call a set "*length-extension-free over $2n$-bit blocks*" when no two strings S_1 and S_2 in it can be written as $S_2 = 0^{2an}\|S_1$ for any positive integer a. We claim that $\mathcal{X}_H$ is length-extension-free over $2n$-bit blocks. We prove it under the following two exhaustive cases:

1) When $|S_1| > 2n$. Every string S_1 in $\mathcal{X}_H$ with length $> 2n$ bits has the form $\mathsf{pad}_{2n}(A)\|\mathsf{pad}_{2n}(B)$ for some strings A and B where the injective function pad_{2n} uniquely allows to separate $\mathsf{pad}_{2n}(A)$ from $\mathsf{pad}_{2n}(B)$. Also, the last $2n$-bit blocks of both $\mathsf{pad}_{2n}(A)$ and $\mathsf{pad}_{2n}(B)$ are always non-zero and denote the lengths of A and B, respectively. Hence, S_2 of the form $0^{2an}\|S_1$ with $a \geq 1$ (i.e., different length than S_1) cannot have the valid form $\mathsf{pad}_{2n}(A')\|\mathsf{pad}_{2n}(B')$ for some strings A' and B's and cannot be in $\mathcal{X}_H$.

2) When $|S_1| = 2n$. For every string S_1 in $\mathcal{X}_H$ with length $= 2n$ bits, we know that any $S_2 = 0^{2an} \| S_1$ with $a \geq 1$ will have size $> 2n$ and hence for any such S_2 to be in $\mathcal{X}_H$ we need it to have a form of $\mathsf{pad}_{2n}(A') \| \mathsf{pad}_{2n}(B')$ for some non-empty strings A' and B'. Since $|S_1| = 2n$, we will have $\mathsf{pad}_{2n}(A') = 0^{2bn}$ for some $0 < b < a$ and thus $|A'| = 0$ which contradicts with the non-empty requirement of A'. This implies no such S_2 can exist in $\mathcal{X}_H$.

We are now ready to define the hash function H and to compute its output collision probability in GEM (when used for both vilF and volF as defined above).

Let the total number of queries made to the underlying hash function H in GEM be q_H and let the lengths of these queries (in $2n$-bit blocks) be $\ell_1, \ell_2, \ldots, \ell_q$ where w.l.o.g., $\ell_i \geq \ell_j$ for all $j \leq i$ and $\sigma_H = \sum_{i=1}^{q_H} \ell_i$. Further, let H for inputs (K, X) with $|K| = k_{\mathsf{h}} = 2n$, $X \in \mathcal{X}_H$ and $X_1, X_2, \ldots, X_\ell \xleftarrow{2n} X$ be defined as the standard $\ell \cdot 2^{-2n}$-XOR-universal polynomial hash [33] as:

$$H(K, X) = K^\ell \cdot X_1 \oplus K^{\ell-1} \cdot X_2 \oplus \ldots \oplus K \cdot X_\ell \,. \tag{8}$$

Note that since $\mathcal{X}_H$ is length-extension-free, the probability of finding a collision in H's outputs is upper bounded by $\sum_{i=1}^{q_H-1} i(\ell_{i+1} \cdot 2^{-2n}) \leq q_H \sigma_H / 2^{2n}$. This is true because for the $(i+1)^{th}$ query there are a total of i many other queries, namely the 1^{st} to i^{th} queries (as $\ell_{i+1} \geq \ell_j$ for all $j \leq i+1$), that can have an output collision with it with probability $\leq \ell_{i+1}/2^{2n}$. More concretely, we have

Theorem 2. *Let* vilF *and* volF *be two keyed functions as defined above that has same keys ($K_2 = K_3$) and use the $\ell \cdot 2^{-2n}$-XOR-universal polynomial hash function H (as of Eqn 8) and the weak* PRF wPRF *then for any adversaries $\mathcal{B}$ and $\mathcal{C}$ making at most $2q$ queries to* vilF *and q queries to* volF*, respectively (i.e., a total of $q_H = 3q$ queries to H), such that the total number of $2n$-bit input blocks processed under H is σ_H, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{vilF}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{volF}}(\mathcal{C}) \leq 2\, \mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{wPRF}}(\mathcal{E}) + \frac{q_H \sigma_H}{2^{2n}}$$

for some adversary $\mathcal{E}$ making at most q_H many wPRF *queries (under fixed but secret and uniform random key) and running in time given by the running time of $\mathcal{A}$ plus $\gamma_1 \cdot q_H$ for some "small" constant γ_1.*

The straightforward proof of Theorem 2 is derived by the observation above. We denote GEM[vilF, volF, E] with vilF and volF defined as above using a weak PRF wPRF and the concrete polynomial hash H of Eqn. 8 by GEM′[wPRF, E].

6 Crafted GEMs: KohiNoor and DaryaiNoor

We propose two instantiations KohiNoor and DaryaiNoor for GEM′. Both use AES as the function E. Their differences lie in the choice of wPRF. KohiNoor uses the key stream generator of GCTR$_s$-3 [2] (denoted by TCTR$_s$-3) which is based on a tweakable expanding primitive such as ButterKnife [3] and enables better parallelization and improved performance for long messages. DaryaiNoor uses the sum of CTR mode SoCTR variable-output-length PRF which is based on an n-bit PRP and provides support for AES cipher applications. Both GCTR$_s$-3 and SoCTR are state-of-the-art solutions for efficient key-stream generation.

6.1 TCTR$_s$-3

GCTR [2] is an encryption mode that can be seen as a generalization of the well-deployed CTR mode of encryption in terms of underlying primitive type and output sizes. Formally, GCTR takes in a k-bit key K, an $(n+t)$-bit IV (originally defined as a pair of n-bit nonce N and t-bit random value R) and a message M, and returns a ciphertext C with $|C| = |M|$ for some positive integers k, n and t. GCTR$_s$-3 is one of the most secure variants of GCTR that internally makes a sequence of calls to its underlying expanding primitive (here an expanding PRF $\mathsf{F}_s : \{0,1\}^k \times \{0,1\}^{2n} \rightarrow \{0,1\}^{sn}$ and hence $n = t$) with $N \| (R \oplus \mathsf{bin}_n(r)) = IV \oplus (0^n \| \mathsf{bin}_n(r))$ as input while incrementing the counter r over primitive calls to generate a key-stream V until $|V| \geq |M|$. The ciphertext is defined as $C = M \oplus_{|M|} V$.

GCTR$_s$-3 with messages of the form $M = 0^v$ can be seen as a key-stream generator that generates v bits of output for the given input IV. We denote this generator by TCTR_s-3. For more details on GCTR$_s$-3, we refer the reader to [2].

We highlight that the original security proof of GCTR$_s$-3, as provided in [2] makes strong assumption on the IV input to be of the form $N \| R$ where N is a unique nonce and R is a uniform random value. In GEM$'$, however, this strong assumption cannot be made as the IVs are generated using an XOR-universal hash function and are not guaranteed to be unique or uniform random. This means the existing analysis of GCTR$_s$-3 becomes inapplicable here and a new security proof (and bound) is required for TCTR_s-3.

Security of TCTR_s-3. TCTR_s-3 that takes $K \in \{0,1\}^k, IV_i \in \{0,1\}^{2n}$ and 0^{l_i} for some integer $l_i > 0$ as input for its i^{th} query simply evaluates a sequence of $\ell'_i = \lceil l_i/(sn) \rceil$ many F_s calls with inputs $IV_i \oplus (0^n \| \mathsf{bin}_n(j))$ where j is a counter that runs from 0 to $\ell'_i - 1$.

Let the total number of queries to TCTR_s-3 (as wPRF in GEM$'$) be upper bounded by q_F with $\sigma_F = \sum_{i=1}^{q_F} \ell'_i$. Let Q be the set of all queried σ_F many underlying F_s inputs here defined as $\cup_{i=1}^{q_F} \{IV_i \oplus (0^n \| \mathsf{bin}_n(j))\}_{j=0}^{\ell'_i - 1}$ and let IV_is here be the outputs of the XOR-universal hash function H (as defined in Eqn. 8) on some distinct input values X_is of length $2\ell_i n$s, respectively with $\sigma'_H = \sum_{i=1}^{q_F} \ell_i$. Notice that when $s \geq 2$, we have $\min\{\ell_i, \ell'_i\} = 1$ for all is which holds due to the input and output domain separation between vilF and volF in GEM$'$ and we have $\sigma'_H + \sigma_F \leq \sigma_H$ (where σ_H is defined in Theorem 2). We call TCTR_s-3 queries that satisfy this relation "*GEM$'$-compatible*". We now state the security of TCTR_s-3 (as wPRF in GEM$'$) in Theorem 3 and defer its proof to Appendix B.1.

Theorem 3. *Let* TCTR_s-3$[\mathsf{F}_s]$ *be the keyed function as defined above with* $s \geq 2$ *then for any adversary* $\mathcal{E}$ *making at most* q_F *GEM$'$-compatible queries* *to* TCTR_s-3 *such that the total number of* sn*-bit and* $2n$*-bit input blocks processed under* F_s *and* H *are* σ_F *and* σ'_H*, respectively, with* $\sigma'_H + \sigma_F \leq \sigma_H$ *then we have*

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[\mathsf{F}_s]}(\mathcal{E}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}_s}(\mathcal{F}) + \frac{\sigma_H^2}{2^{2n}}$$

for some adversary $\mathcal{F}$ *making at most* σ_F *many* F_s *queries (under fixed but secret and uniform random key) and running in time given by the running time of* $\mathcal{E}$ *plus* $\gamma_2 \cdot \sigma_F$ *for some "small" constant* γ_2*.*

We instantiate F_s with ButterKnife [3]. $\mathsf{ButterKnife}_s$ (with an expansion parameter s) is a keyed expanding PRF that maps a string from $\{0,1\}^{2n}$ to a string in $\{0,1\}^{sn}$ for $n = 128$ and $1 \leq s \leq 8$. In other words, it operates on 256-bit inputs and generates up to 1024-bit outputs. ButterKnife is based on AES and leverages the Deoxys-BC design described in [24]. One of the advantages of ButterKnife is its compatibility with processors that support AES native instructions (NI). Skye [7] key derivation function (KDF) for Signal-like applications is a recent examples of ButterKnife's use which benefits from its high parallelizability and AES-based structure on platforms with hardware acceleration.

KohiNoor is a GEM′ where wPRF is instantiated with TCTR_8-3[$\mathsf{ButterKnife}_8$] and achieves full security i.e., $\approx n = 128$ bits (Theorem 1, 2 and 3).

6.2 SoCTR

Let us recall from Section 4.1 that for a given k-bit key K, an n-bit IV and a plaintext M with length m, CTR mode is defined as $\mathsf{CTR}_K(IV, M) := (E_K(IV) \| E_K(IV \oplus \mathsf{bin}_n(1)) \| \cdots \| E_K(IV \oplus \mathsf{bin}_n(\lceil m/n \rceil - 1))) \oplus_m M$ where E is an n-bit block cipher.

Similar to GCTR$_s$-3, CTR mode with messages of the form $M = 0^m$ can be seen as a key-stream generator that generates m bits of output for the given input IV. We denote this generator by CTRg. In other words, for the same inputs, $\mathsf{CTRg}_K(IV, 0^m) := \mathsf{CTR}_K(IV, 0^m)$.

We define SoCTR (short for sum of CTRg) as a domain extended key-stream generator that takes two k-bit keys K_1 and K_2, a $2n$-bit IV $IV_1 \| IV_2$ with $|IV_1| = |IV_2| = n$ and a zero message 0^m and generates an m-bit output as

$$\mathrm{SoCTR}_{K_1,K_2}(IV_1 \| IV_2, 0^m) := \mathsf{CTR}_{K_1}(IV_1, 0^m) \oplus \mathsf{CTR}_{K_2}(IV_2, 0^m)\,.$$

Security of SoCTR. SoCTR that takes $K_1 \in \{0,1\}^k, K_2 \in \{0,1\}^k, IV_i \in \{0,1\}^{2n}$ and 0^{l_i} for some integer $l_i > 0$ as input for its i^{th} query evaluates a sequence of $\ell'_i = \lceil l_i/n \rceil$ many $E_{K_1}(IV_{i1} \oplus \mathsf{bin}_n(j)) \oplus E_{K_2}(IV_{i2} \oplus \mathsf{bin}_n(j))$ calls where j is a counter that runs from 0 to $\ell'_i - 1$ and IV_{i1} and IV_{i2} are two n-bit strings such that $IV_{i1} \| IV_{i2} = IV_i$.

Let the total number of queries to SoCTR (as wPRF in GEM′) be upper bounded by q_E with $\sigma_E = \sum_{i=1}^{q_E} \ell'_i$. Let Q be the set of all queried σ_E many underlying E input pairs $(IV_{i1} \oplus \mathsf{bin}_n(j), IV_{i2} \oplus \mathsf{bin}_n(j))$ and let IV_is here be the outputs of the XOR-universal hash function H (as defined in Eqn. 8) on some distinct input values X_is of length $2\ell_i n$s, respectively with $\sigma'_H = \sum_{i=1}^{q_E} \ell_i$. One can notice that we have $1 \leq \min\{\ell_i, \ell'_i\} \leq 2$ for all is which holds due to the input and output domain separation between vilF and volF in GEM′ and we have $\sigma'_H + \sigma_E \leq \sigma_H$ (where σ_H is defined in Theorem 2). We call SoCTR queries that satisfy this relation "*GEM′-compatible*". We now state the security of SoCTR (as wPRF in GEM′) in Theorem 4 and defer its proof to Appendix B.2.

Theorem 4. *Let* SoCTR$[E]$ *be the keyed function as defined above then for any adversary $\mathcal{E}$ making at most q_E GEM′-compatible queries to* SoCTR *such that the total number of n-bit and $2n$-bit input blocks processed under E_{K_1} (or E_{K_2}) and H are σ_E and σ'_H, respectively, with $\sigma'_H + \sigma_F \leq \sigma_H$ then for $\sigma_E \leq 2^{n-4}$ and $n \geq 4$, we have*

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[E]}(\mathcal{E}) \leq 2\,\mathbf{Adv}^{\mathsf{prp}}_{E}(\mathcal{F}) + \frac{5 \cdot (\sigma_H + 2q_E)^2}{2^{2n}} + \left(\frac{\sigma_E}{2^n}\right)^{1.5}$$

for some adversary $\mathcal{F}$ making at most σ_E many E queries (under fixed but secret and uniform random key) and running in time given by the running time of $\mathcal{E}$ plus $\gamma_3 \cdot \sigma_E$ for some "small" constant γ_3.

$\mathsf{DaryaiNoor}$ is defined as GEM′ with wPRF instantiated with SoCTR[AES-128] and as shown by Theorem 1, 2 and 4, it achieves full $n \approx 128$ bits of security.

Key Size and Derivation. $\mathsf{KohiNoor}$ and $\mathsf{DaryaiNoor}$ require each a total of four keys - a $2n$-bit hashing key for H, two k-bit Feistel keys (used for AES) and a key for wPRF which is k-bit for $\mathsf{ButterKnife}$ in $\mathsf{KohiNoor}$ and $2k$-bit for the two AES instances in $\mathsf{DaryaiNoor}$. This sums to $2n + 3k$ and $2n + 4k$ bits of key material for $\mathsf{KohiNoor}$ and $\mathsf{DaryaiNoor}$, respectively. Note that all of these keys can also be easily generated from a single k-bit key K using the underlying wPRF primitive of GEM′ with input fixed to a constant such as 0^{2n}.

7 Security Analysis

In this section, we provide the deferred proof of Theorem 1.

7.1 Proof of Theorem 1

Proof (Theorem 1). We first replace the keyed functions vilF and volF with two random functions f_{vil} and f_{vol}, respectively where $f_{\mathsf{vil}} \xleftarrow{\$} \mathrm{Func}(*, 2n)$ and $f_{\mathsf{vol}}(\cdot) = f'_{\mathsf{vol}}(\cdot)[0 \ldots l-1]$ with $f'_{\mathsf{vol}} \xleftarrow{\$} \mathrm{Func}(2n, l_{\mathsf{max}})$. Here l represents the length of the corresponding volF output and $l_{\mathsf{max}} = \max_i\{l_i\}$ i.e., maximum of all possible queried output lengths. Note that a keyed vilF is a double argument function that takes two strings as input, thus, to be compatible with the input type, we concatenate both the inputs before evaluating them under f_{vil}. We then also replace $\mathsf{E}_{K_{11}}$ and $\mathsf{E}_{K_{12}}$ with two independent permutations $\pi_1 \leftarrow^{\$} \mathrm{Perm}(n)$ and $\pi_2 \leftarrow^{\$} \mathrm{Perm}(n)$ and denote the resulting mode as $\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$ which gives us

$$\mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[\mathsf{vilF},\mathsf{volF},\mathsf{E}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{vilF}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{volF}}(\mathcal{C}) + 2\,\mathbf{Adv}^{\mathsf{prp}}_{\mathsf{E}}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi]}(\mathcal{A})\,. \tag{9}$$

With slight abuse of notation, let us denote the underlying enciphering and deciphering functions of $\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$ by $\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$ and $\mathrm{GEM}^{-1}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$, respectively. With the above inequality, $\mathcal{A}$ is now left with the goal of distinguishing between the oracles ($\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$, $\mathrm{GEM}^{-1}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$) and a randomly sampled permutation family $(\pi_{\mathcal{T},\mathcal{L}}, \pi^{-1}_{\mathcal{T},\mathcal{L}})$ by making oracle queries to them where

$$(\pi_{\mathcal{T},\mathcal{L}}, \pi^{-1}_{\mathcal{T},\mathcal{L}}) = \cup_{i\in\mathcal{L}}\{\{(\pi_{T,i}, \pi^{-1}_{T,i})\}_{T\in\mathcal{T}} \subseteq^{\$} \mathrm{Perm}(i)\}$$

i.e. for each distinct tweak $T \in \mathcal{T}$ or message length $i \in \mathcal{L}$, the corresponding permutation $\pi_{T,i}$ is uniformly sampled at random from the set $\mathrm{Perm}(i)$ with $\pi^{-1}_{T,i}$ defining the inverse map of $\pi_{T,i}$. Here $\mathcal{T}$ and $\mathcal{L}$ are the tweak space and the message length space of GEM.

$\mathsf{PRP\text{-}PRF}$ **Switching.** We denote duplicate queries and cross-oracle known output queries (i.e., querying the deciphering oracle with an output of enciphering under the same tweak and message length or vice versa) as *trivial* queries and the rest as *non-trivial* queries. We note that trivial queries cannot help $\mathcal{A}$ in increasing its advantage here as the outputs for them are already known and thus are independent of the queried oracle. Hence, we can assume w.l.o.g., that $\mathcal{A}$ only makes non-trivial queries.

We now recall that as per the standard $\mathsf{PRP\text{-}PRF}$ switching lemma [5], a randomly sampled $(\pi_{\mathcal{T},\mathcal{L}}, \pi^{-1}_{\mathcal{T},\mathcal{L}})$ is indistinguishable up to the birthday bound (in the input size) from a randomly sampled function family $(f_{1,\mathcal{T},\mathcal{L}}, f_{2,\mathcal{T},\mathcal{L}})$ for non-trivial oracle queries where

$$(f_{1,\mathcal{T},\mathcal{L}}, f_{2,\mathcal{T},\mathcal{L}}) = \cup_{i\in\mathcal{L}}\{\{(f_{1,T,i}, f_{2,T,i})\}_{T\in\mathcal{T}} \subseteq^{\$} \mathrm{Func}(i,i) \times \mathrm{Func}(i,i)\}$$

i.e., for each distinct tweak $T \in \mathcal{T}$ or message length $i \in \mathcal{L}$, the corresponding functions $f_{1,T,i}$ and $f_{2,T,i}$ are independently and uniformly sampled at random from the set $\mathrm{Func}(i,i)$. More formally, for any adversary $\mathcal{A}$ that makes at most $q_{T,i}$ many non-trivial queries with tweak T and input length i (to both given oracles in total) such that $q = \sum_{(T,i)\in\mathcal{T}\times\mathcal{L}} q_{T,i}$, we have that

$$\left|\Pr[\mathcal{A}^{\pi_{\mathcal{T},\mathcal{L}}(\cdot,\cdot),\pi^{-1}_{\mathcal{T},\mathcal{L}}(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{f_{1,\mathcal{T},\mathcal{L}}(\cdot,\cdot),f_{2,\mathcal{T},\mathcal{L}}(\cdot,\cdot)} \Rightarrow 1]\right| \leq \sum_{(T,i)\in\mathcal{T}\times\mathcal{L}} \frac{q^2_{T,i}}{2^{i+1}} \leq \frac{q^2}{2^{4n+1}}$$

where for any T, M and C, $\pi_{\mathcal{T},\mathcal{L}}(T,M) = \pi_{T,|M|}(M)$, $\pi^{-1}_{\mathcal{T},\mathcal{L}}(T,C) = \pi^{-1}_{T,|C|}(C)$, $f_{1,\mathcal{T},\mathcal{L}}(T,M) = f_{1,T,|M|}(M)$ and $f_{2,\mathcal{T},\mathcal{L}}(T,C) = f_{2,T,|C|}(C)$. Here the last inequality holds due to the assumption that $i \geq 4n$ (see Theorem 1 statement). This implies

$$\begin{aligned}
\mathbf{Adv}^{\textsf{vil-stprp}}_{\mathrm{GEM}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi]}(\mathcal{A}) &= \big| \Pr[\mathcal{A}^{\mathrm{GEM}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi](\cdot,\cdot),\mathrm{GEM}^{-1}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi](\cdot,\cdot)} \Rightarrow 1] \\
&\qquad - \Pr[\mathcal{A}^{\pi_{\mathcal{T},\mathcal{L}}(\cdot,\cdot),\pi^{-1}_{\mathcal{T},\mathcal{L}}(\cdot,\cdot)} \Rightarrow 1]\big| \\
&\le \big| \Pr[\mathcal{A}^{\mathrm{GEM}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi](\cdot,\cdot),\mathrm{GEM}^{-1}[f_{\mathsf{vil}},f_{\mathsf{vol}},\pi](\cdot,\cdot)} \Rightarrow 1] \\
&\qquad - \Pr[\mathcal{A}^{f_{1,\mathcal{T},\mathcal{L}}(\cdot,\cdot),f_{2,\mathcal{T},\mathcal{L}}(\cdot,\cdot)} \Rightarrow 1]\big| + \frac{q^2}{2^{4n+1}}\,. \qquad (10)
\end{aligned}$$

For simplicity, we denote the world with oracles $(\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi](\cdot,\cdot), \mathrm{GEM}^{-1}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi](\cdot,\cdot))$ as the "real world" and with oracles $(f_{1,\mathcal{T},\mathcal{L}}(\cdot,\cdot), f_{2,\mathcal{T},\mathcal{L}}(\cdot,\cdot))$ as the "ideal world". Let the set of $\mathcal{A}$'s queries and responses to the enciphering/forward oracle be $Q_e = \{(M^e_i, T^e_i, C^e_i)^{q_e}_{i=1}\}$ where (M^e_i, T^e_i) represents the query and C^e_i its response. Similarly, let the set of $\mathcal{A}$'s queries and responses to the deciphering/inverse oracle be $Q_d = \{(M^d_i, T^d_i, C^d_i)^{q_d}_{i=1}\}$ where (C^d_i, T^d_i) represents the query and M^d_i its response. We can now define the set of all queries as $Q = Q_e \cup Q_d$.

For simplicity of the analysis, we re-index the elements in Q and redefine it as $Q = \{(M_i, T_i, C_i)^{q_e+q_d}_{i=1}\}$ where $(M_i, T_i, C_i) := (M^e_i, T^e_i, C^e_i)$ for $i = 1$ to q_e and $(M_i, T_i, C_i) := (M^d_{i-q_e}, T^d_{i-q_e}, C^d_{i-q_e})$ for $i = q_e + 1$ to $q_e + q_d$. We emphasize that the re-indexing here is done only to make the description of the upcoming bad events simple and easy to follow and it has no effect on bad event probabilities as they can be seen independent of the query order.

For any message M_i in Q, we denote the leftmost $2n$-bit block of it by $M_{i,L}$ (with the first n-bit of it as $M_{i,L1}$ and the last n-bit as $M_{i,L2}$) and the rest of it by $M_{i,R}$. Similarly, for any ciphertext C_i in Q, the leftmost $2n$-bit block is denoted by $C_{i,L}$ (with the first n-bit of it as $C_{i,L1}$ and the last n-bit as $C_{i,L2}$) and the rest by $C_{i,R}$. We now define three internal variables for $\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$ (as depicted in Figure 1 for $\mathrm{GEM}[\mathsf{vilF}, \mathsf{volF}, \mathsf{E}]$) as

$$\begin{aligned}
X_{i,L} &= (M_{i,L1} \oplus \pi_2(M_{i,L2} \oplus \pi_1(M_{i,L1}))) \| (M_{i,L2} \oplus \pi_1(M_{i,L1}))\,, \\
X_{i,R} &= f_{\mathsf{vil}}(T_i \| 0 \| M_{i,R})\,, \\
Y_{i,L} &= (C_{i,L1} \oplus \pi_2(C_{i,L2})) \| (C_{i,L2} \oplus \pi_1(C_{i,L1} \oplus \pi_2(C_{i,L2})))\,, \\
Y_{i,R} &= f_{\mathsf{vil}}(T_i \| 1 \| C_{i,R})\,, \\
Z_i &= X_{i,L} \oplus X_{i,R} = Y_{i,L} \oplus Y_{i,R}\,. \qquad (11)
\end{aligned}$$

Bad Cases and Analysis. With the notation defined, we now perform an exhaustive case analysis for internal state collisions in the *real* world by defining the following bad events:

Bad1 (f_{vol} Input Collision). For some pair of indices (i, j) with $1 \le i < j \le q_e + q_d$, the two query-response tuples (M_i, T_i, C_i) and (M_j, T_j, C_j) in Q satisfy $Z_i = Z_j$.

Bad2 (Second f_{vil} Input Collision). For some pair of indices (i, j) with $1 \le i < j \le q_e + q_d$, the two query-response tuples (M_i, T_i, C_i) and (M_j, T_j, C_j) in Q satisfy one of the followings:

1. $1 \leq i < j \leq q_e$ and $(C_{i,R}, T_i) = (C_{j,R}, T_j)$.
2. $q_e + 1 \leq i < j \leq q_e + q_d$ and $(M_{i,R}, T_i) = (M_{j,R}, T_j)$.

Note that when none of these cases occur over Q in the real world, we know the followings:

– All Z_i variables for $i = 1$ to $q_e + q_d$ are unique. This means that the right part of every response output which is defined for $i = 1$ to q_e as $C_{i,R} = f_{\mathsf{vol}}(Z_i) \oplus M_{i,R}$ and for $i = q_e + 1$ to $q_e + q_d$ as $M_{i,R} = f_{\mathsf{vol}}(Z_i) \oplus C_{i,R}$ is sampled uniformly at random and thus indistinguishable from uniform random strings.
– All $(C_{i,R}, T_i)$ pairs are unique for $i = 1$ to q_e and all $(M_{i,R}, T_i)$ pairs are unique for $i = q_e + 1$ to $q_e + q_d$. Now, since the left part of every response output can be seen for $1 \leq i \leq q_e$ as

$$C_{i,L} = P_2(P_1(M_{i,L}) \oplus f_{\mathsf{vil}}(T_i \| 1 \| C_{i,R}) \oplus f_{\mathsf{vil}}(T_i \| 0 \| M_{i,R}))$$

and for $q_e + 1 \leq i \leq q_e + q_d$ as

$$M_{i,L} = P_1(P_2(C_{i,L}) \oplus f_{\mathsf{vil}}(T_i \| 1 \| C_{i,R}) \oplus f_{\mathsf{vil}}(T_i \| 0 \| M_{i,R}))$$

for some $2n$-bit permutations P_1 and P_2 and since $(T_i \| 1 \| C_{i,R}) \neq (T_i \| 0 \| M_{i,R})$ for every i due to the domain separation bit, we have that at least one of the two f_{vil} inputs here are unique for every i. This means that the inputs of P_2 for $1 \leq i \leq q_e$ and the inputs of P_1 for $q_e + 1 \leq i \leq q_e + q_d$ are sampled uniformly at random. Now, since P_2 and P_1 are permutations i.e., bijective functions, we have that $C_{i,L}$ for $1 \leq i \leq q_e$ and $M_{i,L}$ for $q_e + 1 \leq i \leq q_e + q_d$ are also uniformly distributed and thus indistinguishable from uniform random strings.

This means when none of these bad events occurs, the responses in the real world are sampled uniformly at random and thus are indistinguishable from uniform random strings. In other words, from Exp. 10 we have that

$$\mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]}(\mathcal{A}) - \frac{q^2}{2^{4n+1}} \leq \Pr(\mathsf{Bad1}) + \Pr(\mathsf{Bad2} \mid \neg\mathsf{Bad1}) \tag{12}$$

where $q = q_e + q_d$. We now provide Lemma 1 that bounds these probability terms individually and defer its proof to Section 7.2.

Lemma 1. *Let* $\mathsf{Bad1}$ *and* $\mathsf{Bad2}$ *are bad cases for* $\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]$ *as defined above then for* $q \leq 2^{n-2}$,

$$\Pr(\mathsf{Bad1}) + \Pr(\mathsf{Bad2} | \neg\mathsf{Bad1}) \leq \frac{q(11q - 1)}{2^{2n}}.$$

Combining Exp. 12 and Lemma 1 together with assuming $q \leq 2^{n-2}$, we get

$$\mathbf{Adv}^{\mathsf{vil\text{-}stprp}}_{\mathrm{GEM}[f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi]}(\mathcal{A}) \leq \frac{11q^2}{2^{2n}}$$

and together with Exp. 9 achieve the first result of Theorem 1. We omit the proof for the second result of Theorem 1 as it can be seen identical as this one but with E modeled as a random function. Hence a difference will appear only in probability bounding of $\mathsf{Bad1.3.1}$ where the sampling probability will now be replaced from being $\leq 1/(2^n - 2q)$ to $1/2^n$. □

7.2 Proof of Lemma 1

Proof (Lemma 1). **Bounding** $\mathsf{Bad1}$**.** Recalling the definition of Z, we know that for any pair of indices (i,j) with $1 \leq i < j \leq q_e + q_d$, $Z_i = Z_j$ can only occur when one of the followings is true:

1. $\mathsf{Bad1.1}$: $1 \leq i < j \leq q_e$ and $X_{i,L} \oplus X_{i,R} = X_{j,L} \oplus X_{j,R}$.
2. $\mathsf{Bad1.2}$: $q_e + 1 \leq i < j \leq q_e + q_d$ and $Y_{i,L} \oplus Y_{i,R} = Y_{j,L} \oplus Y_{j,R}$.
3. $\mathsf{Bad1.3}$: $1 \leq i \leq q_e < j \leq q_e + q_d$ and $X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}$.

In simpler words, it means that $Z_i = Z_j$ can occur among two enciphering queries, two deciphering queries or between one enciphering and one deciphering query.

$\mathsf{Bad1.1}$: We recall that f_{vol} is a uniform random function and since there are no trivial queries in Q, we have that for $1 \leq i < j \leq q_e, (M_i, T_i, C_i) \neq (M_j, T_j, C_j)$ and thus $X_{i,L} \oplus X_{i,R} = X_{j,L} \oplus X_{j,R}$ which is same as $P_1(M_{i,L}) \oplus f_{\mathsf{vil}}(T_i \| 0 \| M_{i,R}) = P_1(M_{j,L}) \oplus f_{\mathsf{vil}}(T_j \| 0 \| M_{j,R})$ (for some permutation P_1) can occur with probability at most $1/2^{2n}$. Since there can be at most $\binom{q_e}{2}$ many such (i,j)s, we get $\Pr(\mathsf{Bad1.1}) \leq q_e(q_e - 1)/2^{2n+1}$.

$\mathsf{Bad1.2}$: Similarly, for $q_e + 1 \leq i < j \leq q_e + q_d, (M_i, T_i, C_i) \neq (M_j, T_j, C_j)$ and thus $Y_{i,L} \oplus Y_{i,R} = Y_{j,L} \oplus Y_{j,R}$ which is same as $P_2(C_{i,L}) \oplus f_{\mathsf{vil}}(T_i \| 1 \| C_{i,R}) = P_2(C_{j,L}) \oplus f_{\mathsf{vil}}(T_j \| 1 \| C_{j,R})$ (for some permutation P_2) can occur with probability at most $1/2^{2n}$. Since there can be at most $\binom{q_d}{2}$ many such (i,j)s, we get $\Pr(\mathsf{Bad1.2}) \leq q_d(q_d - 1)/2^{2n+1}$.

$\mathsf{Bad1.3}$: For $1 \leq i \leq q_e < j \leq q_e + q_d$, we still have $(M_i, T_i, C_i) \neq (M_j, T_j, C_j)$. Let us now consider the following two sub-cases: 1) $\mathsf{Bad1.3.1}$: when $T_i = T_j$ and there exists another query in Q of the form (M_p, T_p, C_p) with $1 \leq p \leq q_e + q_d$ and $p \notin \{i,j\}$ such that $T_p = T_i$ and $(M_{i,R}, C_{j,R}) = (M_{p,R}, C_{p,R})$ and 2) $\mathsf{Bad1.3.2}$: otherwise.

In simple words, these cases capture the scenario when the targeted pair of queries (i,j) have a relation which is known to $\mathcal{A}$ due to some other queries and can be used to achieve $Z_i = Z_j$.

$\mathsf{Bad1.3.1}$: For $1 \leq i \leq q_e < j \leq q_e + q_d$ and $1 \leq p \leq q_e + q_d$ with $p \notin \{i,j\}$, we have that $(M_i, C_i), (M_j, C_j)$ and (M_p, C_p) are distinct, $T_i = T_j = T_p$ and $(M_{i,R}, C_{j,R}) = (M_{p,R}, C_{p,R})$. Additionally, from the definition of Z_p (see Eqn. 11 or Figure 1) we have that $X_{p,L} \oplus Y_{p,L} = X_{p,R} \oplus Y_{p,R}$ which from this known relation is now equal to $X_{i,R} \oplus Y_{j,R}$. The probability for the targeted collision can now be computed as

$$
\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr(X_{i,L} \oplus Y_{j,L} = X_{i,R} \oplus Y_{j,R}) = \Pr(X_{i,L} \oplus Y_{j,L} = X_{p,L} \oplus Y_{p,L}) \\
&= \Pr(P_1(M_{i,L}) \oplus P_2(C_{j,L}) = P_1(M_{p,L}) \oplus P_2(C_{p,L}))
\end{aligned}
$$

where P_1 and P_2 are two permutations. Only at this point, we need to define these permutations to upper bound the probability of the expression above.

In GEM[$f_{\mathsf{vil}}, f_{\mathsf{vol}}, \pi$], we recall that these permutations are defined as shown in Exp. 11 i.e., for every i,

$$P_1(M_{i,L}) := X_{i,L} = (M_{i,L1} \oplus \pi_2(\underbrace{M_{i,L2} \oplus \pi_1(M_{i,L1})}_{\Delta_{M,i}}))\|(M_{i,L2} \oplus \pi_1(M_{i,L1}))\,,$$

$$P_2(C_{i,L}) := Y_{i,L} = (C_{i,L1} \oplus \pi_2(C_{i,L2}))\|(C_{i,L2} \oplus \pi_1(\underbrace{C_{i,L1} \oplus \pi_2(C_{i,L2})}_{\Delta_{C,i}}))\,.$$

Hence, we get

$$\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr(P_1(M_{i,L}) \oplus P_2(C_{j,L}) = P_1(M_{p,L}) \oplus P_2(C_{p,L})) \\
&= \Pr((M_{i,L1} \oplus C_{j,L1} \oplus \pi_2(C_{j,L2}) \oplus \pi_2(\Delta_{M,i})) \\
&\qquad\qquad \|(M_{i,L2} \oplus C_{j,L2} \oplus \pi_1(M_{i,L1}) \oplus \pi_1(\Delta_{C,j})) = \\
&\qquad (M_{p,L1} \oplus C_{p,L1} \oplus \pi_2(C_{p,L2}) \oplus \pi_2(\Delta_{M,p})) \\
&\qquad\qquad \|(M_{p,L2} \oplus C_{p,L2} \oplus \pi_1(M_{p,L1}) \oplus \pi_1(\Delta_{C,p}))\,.
\end{aligned} \tag{13}$$

We also highlight an important observation that (M_i, C_i), (M_j, C_j) and (M_p, C_p) are different queries yet $(M_{i,R}, C_{j,R}) = (M_{p,R}, C_{p,R})$. This means $M_{i,L} \neq M_{p,L}$ and $C_{j,L} \neq C_{p,L}$. We are now set to bound the probability of Exp. 13. Let us consider the following four cases:

$\mathsf{Bad1.3.1.1}$: When $(C_{j,L2}, M_{i,L1}) = (C_{p,L2}, M_{p,L1})$. Clearly, under this case, we have that $C_{j,L1} \neq C_{p,L1}$ and $M_{i,L2} \neq M_{p,L2}$ which means $\Delta_{M,i} \neq \Delta_{M,p}$ and $\Delta_{C,j} \neq \Delta_{C,p}$ and hence from Exp. 13, we have that

$$\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr((C_{j,L1} \oplus \pi_2(\Delta_{M,i}))\|(M_{i,L2} \oplus \pi_1(\Delta_{C,j})) = \\
&\qquad (C_{p,L1} \oplus \pi_2(\Delta_{M,p}))\|(M_{p,L2} \oplus \pi_1(\Delta_{C,p})) \leq \frac{1}{(2^n - 2q)^2}\,.
\end{aligned}$$

Here the last inequality holds as there are at most $2q$ many π_1 and π_2 calls that can be induced under q many GEM calls and therefore the probability of predicting their output is upper bounded by $1/(2^n - 2q)$ each.

$\mathsf{Bad1.3.1.2}$: When $C_{j,L2} \neq C_{p,L2}$ but $M_{i,L1} = M_{p,L1}$. This means $M_{i,L2} \neq M_{p,L2}$ and $\Delta_{M,i} \neq \Delta_{M,p}$ and thus from Exp. 13, we have

$$\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr((C_{j,L1} \oplus \pi_2(\Delta_{M,i}) \oplus \pi_2(C_{j,L2}))\|(M_{i,L2} \oplus \pi_1(\Delta_{C,j}) \oplus C_{j,L2}) = \\
&\quad (C_{p,L1} \oplus \pi_2(\Delta_{M,p}) \oplus \pi_2(C_{p,L2}))\|(M_{p,L2} \oplus \pi_1(\Delta_{C,p}) \oplus C_{p,L2}) \leq \frac{6}{(2^n - 2q)^2}\,.
\end{aligned}$$

Here the last inequality holds as there are at most $2q$ many π_1 and π_2 calls that can be induced under q many GEM calls and therefore the probability of predicting their outputs, when the inputs are unique i.e., when $\{\Delta_{M,i}, \Delta_{M,p}\} \neq$

$\{C_{j,L2}, C_{p,L2}\}$ for π_2 and when $\Delta_{C,j} \neq \Delta_{C,p}$ for π_1 is upper bounded by $1/(2^n - 2q)$ each.

Note that by definition, each Δ here can be seen as an expression containing one random permutation output, which means we get $\Pr(\{\Delta_{M,i}, \Delta_{M,p}\} = \{C_{j,L2}, C_{p,L2}\}) \leq 2/(2^n - 2q)$ and $\Pr(\Delta_{C,j} = \Delta_{C,p}) \leq 1/(2^n - 2q)$ and hence by the total probability theorem, we get $\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \leq (1/(2^n - 2q) + 2/(2^n - 2q))(1/(2^n - 2q) + 1/(2^n - 2q)) = 6/(2^n - 2q)^2$.

Bad1.3.1.3 : When $M_{i,L1} \neq M_{p,L1}$ but $C_{j,L2} = C_{p,L2}$. This means $C_{j,L1} \neq C_{p,L1}$ and $\Delta_{C,j} \neq \Delta_{C,p}$ and thus from Exp. 13, we have

$$\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr((C_{j,L1} \oplus \pi_2(\Delta_{M,i}) \oplus M_{i,L1}))\|(M_{i,L2} \oplus \pi_1(\Delta_{C,j}) \oplus \pi_1(M_{i,L1})) = \\
&\quad (C_{p,L1} \oplus \pi_2(\Delta_{M,p}) \oplus M_{p,L1}))\|(M_{p,L2} \oplus \pi_1(\Delta_{C,p}) \oplus \pi_1(M_{p,L1})) \leq \frac{6}{(2^n - 2q)^2}.
\end{aligned}$$

Here the last inequality holds as there are at most $2q$ many π_1 and π_2 calls that can be induced under q many GEM calls and therefore the probability of predicting their outputs, when the inputs are unique i.e., when $\{\Delta_{C,j}, \Delta_{C,p}\} \neq \{M_{i,L1}, M_{p,L1}\}$ for π_1 and when $\Delta_{M,i} \neq \Delta_{M,p}$ for π_2 is upper bounded by $1/(2^n - 2q)$ each.

Again, by definition, each Δ here can be seen as an expression containing one random permutation output, which means we get $\Pr(\{\Delta_{C,j}, \Delta_{C,p}\} = \{M_{i,L1}, M_{p,L1}\}) \leq 2/(2^n - 2q)$ and $\Pr(\Delta_{M,i} = \Delta_{M,p}) \leq 1/(2^n - 2q)$ and hence by the total probability theorem, we get $\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \leq (1/(2^n - 2q) + 1/(2^n - 2q))(1/(2^n - 2q) + 2/(2^n - 2q)) = 6/(2^n - 2q)^2$.

Bad1.3.1.4 : When $M_{i,L1} \neq M_{p,L1}$ and $C_{j,L2} \neq C_{p,L2}$. We have

$$\begin{aligned}
&\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \\
&= \Pr((M_{i,L1} \oplus C_{j,L1} \oplus \pi_2(C_{j,L2}) \oplus \pi_2(\Delta_{M,i})) \\
&\qquad\qquad \|(M_{i,L2} \oplus C_{j,L2} \oplus \pi_1(M_{i,L1}) \oplus \pi_1(\Delta_{C,j})) = \\
&\quad (M_{p,L1} \oplus C_{p,L1} \oplus \pi_2(C_{p,L2}) \oplus \pi_2(\Delta_{M,p})) \\
&\qquad\qquad \|(M_{p,L2} \oplus C_{p,L2} \oplus \pi_1(M_{p,L1}) \oplus \pi_1(\Delta_{C,p})) \leq \frac{9}{(2^n - 2q)^2}.
\end{aligned}$$

Here the last inequality holds as there are at most $2q$ many π_1 and π_2 calls that can be induced under q many GEM calls and therefore the probability of predicting their outputs, when the inputs are unique i.e., when $\{\Delta_{M,i}, \Delta_{M,p}\} \neq \{C_{j,L2}, C_{p,L2}\}$ for π_2 and $\{\Delta_{C,j}, \Delta_{C,p}\} \neq \{M_{i,L1}, M_{p,L1}\}$ for π_1 is upper bounded by $1/(2^n - 2q)$ each.

Since each Δ can be written as an expression containing one random permutation output, we get $\Pr(\{\Delta_{M,i}, \Delta_{M,p}\} = \{C_{j,L2}, C_{p,L2}\}) \leq 2/(2^n - 2q)$ and $\Pr(\{\Delta_{C,j}, \Delta_{C,p}\} = \{M_{i,L1}, M_{p,L1}\}) \leq 2/(2^n - 2q)$ and hence by the total probability theorem, we get $\Pr(X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}) \leq (1/(2^n - 2q) + 2/(2^n - 2q))(1/(2^n - 2q) + 2/(2^n - 2q)) = 9/(2^n - 2q)^2$.

Now, for a given (i, j) there can be more than one values of p, let say, $1 \leq p_1 < p_2 \leq q_e + q_d$ only if $f_{\mathsf{vol}}(Z_{p_1}) = f_{\mathsf{vol}}(Z_{p_2})$ due to the relation $M_{p_1,R} \oplus C_{p_1,R} = M_{p_2,R} \oplus C_{p_2,R}$. Let us consider that event $\mathsf{Bad1.1}$ and $\mathsf{Bad1.2}$ does not occur then we know that $Z_{p_1} \neq Z_{p_2}$ and hence $\Pr(f_{\mathsf{vol}}(Z_{p_1}) = f_{\mathsf{vol}}(Z_{p_2})) \leq 1/2^{2n}$. Since there can be at most $q = (q_e + q_d)$ many choices for p_1 and p_2, we get that for a given (i, j) there can be more than one values of p with at most probability of $q^2/2^{2n+1}$.

Since there can be at most q_e many choices for i and q_d many choices for j, we get $\Pr(\mathsf{Bad1.3.1}|\neg\mathsf{Bad1.1} \wedge \neg\mathsf{Bad1.2}) \leq q_e \cdot q_d \cdot \max\{1, 6, 6, 9\}/(2^n - 2q)^2 + q^2/2^{2n+1} = 9q_e q_d/(2^n - 2q)^2 + q^2/2^{2n+1}$.

Remark. We highlight that the bad event $\mathsf{Bad1.3.1}$ captures the possibility of the shared difference attack as shown in Section 4.2 and 4.3.

$\mathsf{Bad1.3.2}$: For $1 \leq i \leq q_e < j \leq q_e + q_d$, we have $(M_i, T_i, C_i) \neq (M_j, T_j, C_j)$. Additionally, we have that either $T_i \neq T_j$ or for all queries in Q of the form (M_p, T_p, C_p) with $1 \leq p \leq q_e + q_d$ and $p \notin \{i, j\}$ we have $T_p \neq T_i$ or $(M_{i,R}, C_{j,R}) \neq (M_{p,R}, C_{p,R})$. In other words, there is no known relation between i^{th} and j^{th} query. Hence, we have that $X_{i,L} \oplus X_{i,R} = Y_{j,L} \oplus Y_{j,R}$ which is same as $P_1(M_{i,L}) \oplus f_{\mathsf{vil}}(T_i \| 0 \| M_{i,R}) = P_2(C_{j,L}) \oplus f_{\mathsf{vil}}(T_j \| 1 \| C_{j,R})$ (for some permutations P_1 and P_2) can occur with probability at most $1/2^{2n}$ (due to the domain separated and thus independent f_{vil} calls). Since there can be at most $q_e \cdot q_d$ many such (i, j)s, we get $\Pr(\mathsf{Bad1.3.2}) \leq q_e \cdot q_d/2^{2n}$.

$\mathsf{Bad2}$: We first assume that the bad event $\mathsf{Bad1}$ does not occur which implies that for every query-response tuple (M, T, C) in Q, the right part of the response is uniformly distributed. More concretely, all $C_{i,R}$s with $1 \leq i \leq q_e$ and all $M_{i,R}$s with $q_e + 1 \leq i \leq q_e + q_d$ are uniformly distributed. Now, since for all $C_{i,R}$s and $M_{i,R}$s, $|C_{i,R}| \geq 2n$ and $|M_{i,R}| \geq 2n$ and there are total $\binom{q_e}{2} + \binom{q_d}{2}$ ways to choose (i, j), we get

$$\Pr(\mathsf{Bad2}|\neg\mathsf{Bad1}) \leq \frac{(q_e(q_e - 1) + q_d(q_d - 1))}{2^{2n+1}}.$$

In total, we get from the union bound that $\Pr(\mathsf{Bad1}) + \Pr(\mathsf{Bad2}|\neg\mathsf{Bad1})$

$$\begin{aligned}
&\leq \Pr(\mathsf{Bad1.1}) + \Pr(\mathsf{Bad1.2}) + \Pr(\mathsf{Bad1.3.1}|\neg\mathsf{Bad1.1} \wedge \neg\mathsf{Bad1.2}) + \Pr(\mathsf{Bad1.3.2}) \\
&\qquad + \Pr(\mathsf{Bad2}|\neg\mathsf{Bad1}) \\
&\leq 2\Big(\frac{q_e^2 - q_e}{2^{2n+1}} + \frac{q_d^2 - q_d}{2^{2n+1}}\Big) + \Big(\frac{9q_e q_d}{(2^n - 2q)^2} + \frac{q^2}{2^{2n+1}}\Big) + \frac{q_e q_d}{2^{2n}} \\
&\leq \frac{q^2 - q}{2^{2n}} + \Big(\frac{9q^2}{4(2^n - 2q)^2} + \frac{q^2}{2^{2n+1}}\Big).
\end{aligned}$$

Now, assuming $q \leq 2^{n-2}$, we get

$$\Pr(\mathsf{Bad1}) + \Pr(\mathsf{Bad2}|\neg\mathsf{Bad1}) \leq \frac{q(11q - 1)}{2^{2n}}$$

and therefore the result of Lemma 1. $\square$

Acknowledgements

This work was supported by CyberSecurity Research Flanders with reference number VR20192203. This work was supported in part by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058 and by the Flemish Government through FWO Project G.0835.16 A security Architecture for IoT. Elena Andreeva was supported by the Austrian Science Fund (FWF) SpyCoDe grant with number 10.55776/F8507-N.

References

1. Abel, A., Reineke, J.: uops.info: Characterizing latency, throughput, and port usage of instructions on intel microarchitectures. In: ASPLOS. pp. 673–686. ASPLOS '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3297858.3304062, http://doi.acm.org/10.1145/3297858.3304062
2. Andreeva, E., Bhati, A.S., Preneel, B., Vizár, D.: 1, 2, 3, Fork: Counter Mode Variants based on a Generalized Forkcipher. IACR Trans. Symmetric Cryptol. **2021**(3), 1–35 (2021)
3. Andreeva, E., Cogliati, B., Lallemand, V., Minier, M., Purnal, A., Roy, A.: Masked Iterate-Fork-Iterate: A new Design Paradigm for Tweakable Expanding Pseudorandom Function. In: 22nd International Conference on Applied Cryptography and Network Security (2024)
4. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer (2000)
5. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. EUROCRYPT 2006 (2006)
6. Bernstein, D.J.: Cryptographic competitions: CAESAR. http://competitions.cr.yp.to
7. Bhati, A.S., Dufka, A., Andreeva, E., Roy, A., Preneel, B.: Skye: A Fast KDF based on Expanding PRF and its Application to Signal. In: 19th ACM ASIA Conference on Computer and Communications Security. ACM (2024)
8. Bhaumik, R., List, E., Nandi, M.: ZCZ–Achieving n-bit SPRP Security with a Minimal Number of Tweakable-block-cipher Calls. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 336–366. Springer (2018)
9. Chakraborty, D., Ghosh, S., López, C.M., Sarkar, P.: FAST: disk encryption and beyond. Cryptology ePrint Archive (2017)
10. Chakraborty, D., Hernandez-Jimenez, V., Sarkar, P.: Another look at XCB. Cryptography and Communications **7**, 439–468 (2015)
11. Cogliati, B., Ethan, J., Lallemand, V., Lee, B., Lee, J., Minier, M.: CTET+: A beyond-birthday-bound secure tweakable enciphering scheme using a single pseudorandom permutation. IACR Transactions on Symmetric Cryptology **2021**(4), 1–35 (Dec 2021). https://doi.org/10.46586/tosc.v2021.i4.1-35, https://tosc.iacr.org/index.php/ToSC/article/view/9327
12. Crowley, P., Huckleberry, N., Biggers, E.: Length-preserving encryption with HCTR2. Cryptology ePrint Archive (2021)

13. Dai, W., Hoang, V.T., Tessaro, S.: Information-theoretic indistinguishability via the chi-squared method. In: Annual International Cryptology Conference. pp. 497–523. Springer (2017)
14. Dobraunig, C., Mennink, B., Neves, S.: EliMAC: Speeding Up LightMAC by around 20%. IACR Transactions on Symmetric Cryptology **2023**(2), 69–93 (2023)
15. Dworkin, M.J.: Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices (2010)
16. Gueron, S.: A New Interpretation for the GHASH Authenticator of AES-GCM. In: Dolev, S., Gudes, E., Paillier, P. (eds.) Cyber Security, Cryptology, and Machine Learning. pp. 424–438. Springer Nature Switzerland, Cham (2023)
17. Gueron, S., Kounavis, M.E.: Intel® carry-less multiplication instruction and its usage for computing the GCM mode (2014)
18. Gueron, S., Lindell, Y.: GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 109–119 (2015)
19. Halevi, S.: EME*: Extending EME to handle arbitrary-length messages with associated data. In: International Conference on Cryptology in India. pp. 315–327. Springer (2004)
20. Halevi, S.: Invertible universal hashing and the TET encryption mode. In: Advances in Cryptology-CRYPTO 2007. pp. 412–429. Springer (2007)
21. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Cryptographers' Track at the RSA Conference. pp. 292–304. Springer (2004)
22. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust Authenticated-Encryption AEZ and the Problem That It Solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer (2015)
23. Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v5: Authenticated Encryption by Enciphering (2017), CAESAR 3rd Round, `https://competitions.cr.yp.to/round3/aezv5.pdf`
24. Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: Deoxys v1.41 (2016), `https://competitions.cr.yp.to/round3/deoxysv141.pdf`
25. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Doklady **7**, 595 (12 1962)
26. McGrew, D., Viega, J.: The Galois/counter mode of operation (GCM). submission to NIST Modes of Operation Process (2004), `https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf`
27. McGrew, D.A., Fluhrer, S.R.: The extended codebook (XCB) mode of operation. Cryptology ePrint Archive (2004)
28. McGrew, D.A., Fluhrer, S.R.: The security of the extended codebook (XCB) mode of operation. In: Selected Areas in Cryptography. pp. 311–327. Springer (2007)
29. Mouha, N., Dworkin, M.: Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series. Tech. rep., National Institute of Standards and Technology (2023)
30. Nandi, M.: An Efficient SPRP-secure Construction based on Pseudo Random Involution. Cryptology ePrint Archive (2008)
31. Sarkar, P.: Improving upon the TET mode of operation. In: Information Security and Cryptology-ICISC 2007: 10th International Conference, Seoul, Korea, November 29-30, 2007. Proceedings 10. pp. 180–192. Springer (2007)
32. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Advances in Cryptology-ASIACRYPT 2013: 19th

International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I 19. pp. 405–423. Springer (2013)
33. Wang, P., Li, Y., Zhang, L., Zheng, K.: Related-key almost universal hash functions: definitions, constructions and applications. In: International Conference on Fast Software Encryption. pp. 514–532. Springer (2016)
34. Wang, P., Mao, S., Xu, R., Jing, J., Wang, Y.: How to Recover the Full Plaintext of XCB. Cryptology ePrint Archive, Paper 2024/1527 (2024), `https://eprint.iacr.org/2024/1527`

A XCBv2fb: Block Diagram

We provide a block diagram of XCBv2fb in Figure 2.

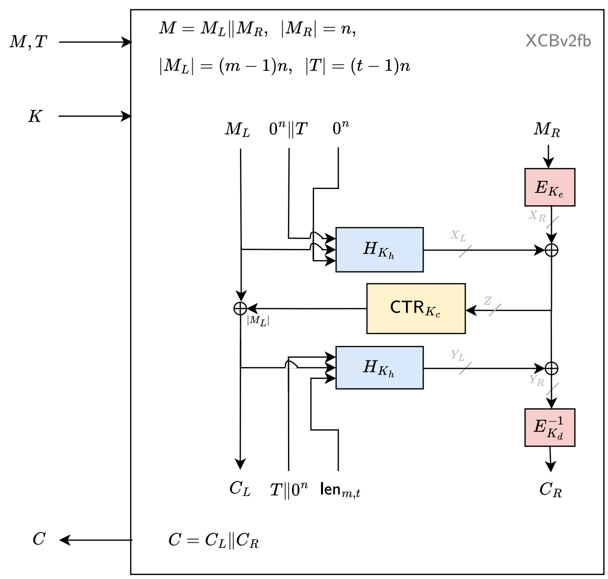

Fig. 2: XCBv2fb mode (a.k.a. full-block XCBv2). Here K_h, K_e, K_d and K_c are derived from K as shown in Sec. 4.1 and $\mathsf{len}_{m,t}$ represents $\mathsf{bin}_{n/2}(|T|) \| \mathsf{bin}_{n/2}(|C_L|)$.

B Omitted Proofs

B.1 Proof of Theorem 3

Proof (Theorem 3). We first replace the underlying primitive F_s of $\mathsf{TCTR}_s\text{-}3[\mathsf{F}_s]$ with a uniform random function $f_s \xleftarrow{\$} \text{Func}(2n, sn)$ and denote the updated $\mathsf{TCTR}_s\text{-}3$ as $\mathsf{TCTR}_s\text{-}3[f_s]$. This implies

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[\mathsf{F}_s]}(\mathcal{E}) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}_s}(\mathcal{F}) + \mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[f_s]}(\mathcal{E})\,. \tag{14}$$

Now, $\mathcal{E}$ is left with the goal of distinguishing the outputs of $\mathsf{TCTR}_s\text{-}3[f_s]$ from uniform random strings of same size. We recall that $\mathsf{TCTR}_s\text{-}3[f_s]$ for i^{th} input $(IV_i, 0^{l_i})$ with $l_i > 0$, $\ell'_i = \lceil l_i/(sn) \rceil$ and $IV_i = H(X_i)$ for some unique X_i is defined as

$$f_s(H(X_i) \oplus (0^n \| \mathsf{bin}_n(0))) \| \cdots \| f_s(H(X_i) \oplus (0^n \| \mathsf{bin}_n(\ell'_i - 1)))[0 \ldots l_i - 1] .$$

This means as long as the inputs of all f_s queries are unique, the outputs of $\mathsf{TCTR}_s\text{-}3[f_s]$ are uniformly distributed and thus indistinguishable from uniform random strings. More formally, for $Q = \cup_{i=1}^{q_F} \{H(X_i) \oplus (0^n \| \mathsf{bin}_n(j))\}_{j=0}^{\ell'_i - 1}$, we have

$$\begin{aligned} \mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[f_s]}(\mathcal{E}) \leq \Pr(&\text{Given } Q, \exists\ 1 \leq i < i' \leq q_F \\ &\text{and } 0 \leq j \leq \max\{\ell'_i, \ell'_{i'}\} - 1 \text{ such that} \\ &H(X_i) \oplus H(X_{i'}) = 0^n \| \mathsf{bin}_n(j)) . \end{aligned}$$

Now, since H is an $\ell \cdot 2^{-2n}$-XOR-universal hash function with ℓ representing the maximum length (in $2n$-bit blocks) of the XORed hash inputs, we get that the probability of $H(X_i) \oplus H(X_{i'}) = 0^n \| \mathsf{bin}_n(j)$ for a given i, i', j, X_i and $X_{i'}$ is $\leq \max\{\ell_i, \ell_{i'}\}/2^{2n}$. We note that there are total $\sum_{i'=2}^{q_F} \sum_{i=1}^{i'-1} \max\{\ell'_i, \ell'_{i'}\}$ choices for picking (i, j, j') which gives us

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[f_s]}(\mathcal{E}) \leq \sum_{i'=2}^{q_F} \sum_{i=1}^{i'-1} \frac{\max\{\ell'_i, \ell'_{i'}\} \cdot \max\{\ell_i, \ell_{i'}\}}{2^{2n}} .$$

Let us now recall that since all these q_F queries are GEM′-compatible, we have $\min\{\ell_i, \ell'_i\} = 1$ and hence

$$\begin{aligned} \mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathsf{TCTR}_s\text{-}3[f_s]}(\mathcal{E}) \leq& \sum_{i'=2}^{q_F} \sum_{i=1}^{i'-1} \frac{\max\{\ell'_i \cdot \ell_{i'}, \ell'_{i'} \cdot \ell_i\}}{2^{2n}} \leq \sum_{i'=2}^{q_F} \sum_{i=1}^{i'-1} \frac{(\ell'_i \cdot \ell_{i'}) + (\ell'_{i'} \cdot \ell_i)}{2^{2n}} \\ =& \frac{1}{2^{2n}} \cdot \left(\sum_{i'=2}^{q_F} \ell_{i'} \sum_{i=1}^{i'-1} \ell'_i + \sum_{i'=2}^{q_F} \ell'_{i'} \sum_{i=1}^{i'-1} \ell_i \right) \leq \frac{2\sigma'_H \cdot \sigma_F}{2^{2n}} \\ \leq& \frac{(\sigma'_H + \sigma_F)^2}{2^{2n}} \leq \frac{\sigma_H^2}{2^{2n}} . \end{aligned} \tag{15}$$

Now, combining the Exp. 14 and 15 gives us the result of Theorem 3. □

B.2 Proof of Theorem 4

Proof (Theorem 4). We first replace the underlying primitives E_{K_1} and E_{K_2} of SoCTR$[E]$ with two uniform random n-bit permutations π_1 and π_2 and denote the updated mode as SoCTR$[\pi]$. This implies

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[E]}(\mathcal{E}) \leq 2\, \mathbf{Adv}^{\mathsf{prp}}_{E}(\mathcal{F}) + \mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E}) . \tag{16}$$

Now, $\mathcal{E}$ is left with the goal of distinguishing the outputs of SoCTR$[\pi]$ from uniform random strings of same size. We recall that SoCTR$[\pi]$ for i^{th} input $(IV_i, 0^{l_i})$ with $l_i > 0, IV_i \in \{0,1\}^{2n}, \ell'_i = \lceil l_i/n \rceil$ and $IV_i = H(X_i)$ for some unique X_i is defined as

$$\begin{aligned}&(\oplus_{a=1}^{2}\pi_a(H_a(X_i) \oplus \mathsf{bin}_n(0)))\| \cdots \|(\oplus_{a=1}^{2}\pi_a(H_a(X_i) \oplus \mathsf{bin}_n(\ell'_i - 1)))[0 \dots l_i - 1] \\ =&(\|_{j=0}^{\ell'_i-1}(\oplus_{a=1}^{2}\pi_a(H_a(X_i) \oplus \mathsf{bin}_n(j))))[0 \dots l_i - 1]\end{aligned}$$

where $H_1(X_i) = H(X_i)[0 \dots n-1]$ and $H_2(X_i) = H(X_i)[n \dots 2n-1]$. We can now perform an exhaustive case analysis to bound the probability of bias in SoCTR$[\pi]$ outputs when compared with uniform random outputs. Let us consider a pair of indices $(i, j) < (i', j')$ with $1 \le i \le i' \le q_E, 0 \le j \le \ell'_i - 1$ and $0 \le j' \le \ell'_{i'} - 1$ then

1. when π_1 and π_2 input collides: That is, when $H_1(X_i) \oplus \mathsf{bin}_n(j)) = H_1(X_{i'}) \oplus \mathsf{bin}_n(j')$ and $H_2(X_i) \oplus \mathsf{bin}_n(j)) = H_2(X_{i'}) \oplus \mathsf{bin}_n(j')$, the outputs of $(i', j')^{th}$ π_1 and π_2 calls have no entropy and hence their sum is easily distinguishable from uniform random strings. Since H is an $\ell \cdot 2^{-2n}$-XOR universal hash and there can be at most $\max\{\ell'_i, \ell'_{i'}\}$ many choices for $\mathsf{bin}_n(j)) \oplus \mathsf{bin}_n(j')$, we get $\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E})$ under this case bounded by $\sum_{i'=2}^{q_E}\sum_{i=1}^{i'-1}\max\{\ell'_i, \ell'_{i'}\} \cdot \max\{\ell_i, \ell_{i'}\}/2^{2n}$.
2. when only π_1 input collides: That is, when $H_1(X_i) \oplus \mathsf{bin}_n(j)) = H_1(X_{i'}) \oplus \mathsf{bin}_n(j')$ and $H_2(X_i) \oplus \mathsf{bin}_n(j)) \neq H_2(X_{i'}) \oplus \mathsf{bin}_n(j')$. This means the output of $(i', j')^{th}$ π_2 call is still randomly sampled with probability at most $1/(2^n - \sigma_E)$. Since H is an $\ell \cdot 2^{-2n}$-XOR universal hash, we have that H_1 is also an $\ell \cdot 2^{-n}$-XOR universal hash and since there can be at most $\max\{\ell'_i, \ell'_{i'}\}$ many choices for $\mathsf{bin}_n(j)) \oplus \mathsf{bin}_n(j')$, we get $\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E})$ under this case bounded by $\sum_{i'=2}^{q_E}\sum_{i=1}^{i'-1}\max\{\ell'_i, \ell'_{i'}\} \cdot \max\{\ell_i, \ell_{i'}\}/(2^n(2^n - \sigma_E))$.
3. when only π_2 input collides: That is, when $H_1(X_i) \oplus \mathsf{bin}_n(j)) \neq H_1(X_{i'}) \oplus \mathsf{bin}_n(j')$ and $H_2(X_i) \oplus \mathsf{bin}_n(j)) = H_2(X_{i'}) \oplus \mathsf{bin}_n(j')$. This means the output of $(i', j')^{th}$ π_1 call is still randomly sampled with probability at most $1/(2^n - \sigma_E)$. Since H is an $\ell \cdot 2^{-2n}$-XOR universal hash, we have that H_2 is also an $\ell \cdot 2^{-n}$-XOR universal hash and since there can be at most $\max\{\ell'_i, \ell'_{i'}\}$ many choices for $\mathsf{bin}_n(j)) \oplus \mathsf{bin}_n(j')$, we get $\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E})$ under this case bounded by $\sum_{i'=2}^{q_E}\sum_{i=1}^{i'-1}\max\{\ell'_i, \ell'_{i'}\} \cdot \max\{\ell_i, \ell_{i'}\}/(2^n(2^n - \sigma_E))$.
4. when all π_1 and π_2 inputs are unique: Under this case, we note that the SoCTR$[\pi]$ outputs are identically distributed to the well-studied PRF design SoP2 a.k.a. XOR2 (short for sum of two independent permutations with distinct inputs). Hence applying the state-of-the-art result of Theorem 4 from [13], we get for $\sigma_E \le 2^{n-4}$ and $n \ge 4$, $\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E})$ bounded by $(\sigma_E/2^n)^{1.5}$.

Now, combining all these bounds together, we get for $\sigma_E \le 2^{n-4}$ and $n \ge 4$,

$$\mathbf{Adv}^{\mathsf{wk\text{-}prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E}) \le 5 \cdot \sum_{i'=2}^{q_E}\sum_{i=1}^{i'} \frac{\max\{\ell'_i, \ell'_{i'}\} \cdot \max\{\ell_i, \ell_{i'}\}}{2^{2n}} + \left(\frac{\sigma_E}{2^n}\right)^{1.5}.$$

Let us now recall that since all these q_E queries are GEM′-compatible, we have $\min\{\ell_i, \ell'_i\} \leq 2$ and $\min\{\ell_i + 1, \ell'_i + 1\} \geq 2$, and hence

$$\begin{aligned}
\mathbf{Adv}^{\textsf{wk-prf}}_{\mathrm{SoCTR}[\pi]}(\mathcal{E}) - \left(\frac{\sigma_E}{2^n}\right)^{1.5} &\leq 5 \cdot \sum_{i'=2}^{q_E} \sum_{i=1}^{i'} \frac{\max\{(\ell'_i+1)\cdot(\ell_{i'}+1), (\ell'_{i'}+1)\cdot(\ell_i+1)\}}{2^{2n}} \\
&\leq 5 \cdot \sum_{i'=2}^{q_E} \sum_{i=1}^{i'} \frac{(\ell'_i+1)\cdot(\ell_{i'}+1) + (\ell'_{i'}+1)\cdot(\ell_i+1)}{2^{2n}} \\
&= \frac{5}{2^{2n}} \cdot \left(\sum_{i'=2}^{q_E} (\ell_{i'}+1) \sum_{i=1}^{i'} (\ell'_i+1) + \sum_{i'=2}^{q_E} (\ell'_{i'}+1) \sum_{i=1}^{i'} (\ell_i+1) \right) \\
&\leq \frac{5 \cdot 2 \cdot (\sigma'_H + q_E) \cdot (\sigma_E + q_E)}{2^{2n}} \\
&\leq \frac{5 \cdot (\sigma'_H + \sigma_E + 2q_E)^2}{2^{2n}} \leq \frac{5 \cdot (\sigma_H + 2q_E)^2}{2^{2n}} . \qquad (17)
\end{aligned}$$

Now, combining the Exp. 16 and 17 gives us the result of Theorem 4. □

C Implementation Aspects of KohiNoor and DaryaiNoor

The performance of KohiNoor and DaryaiNoor is determined by their three major components, $\mathsf{ButterKnife}_8$ in the GCTR-3 mode, SoCTR instantiated with AES and the 256-bit polynomial hash. In this section, we discuss the implementation of these three components.

$\mathsf{ButterKnife}_8$ and SoCTR can be efficiently evaluated with native instructions for the AES round-function that can be found in most modern processors. Similarly, polynomial hashes over finite fields of characteristic 2 can be efficiently evaluated with the carry-less multiplication (`CLMUL` on x86_64) or polynomial multiplication (`PMULL` on ARM) instruction. Both the AES and polynomial multiplication instruction have high throughput, but also high latency on modern processors. For example on Intel's raptor cove microarchitecture, the AES round-function instruction `AESenc` has a throughput of 1 cycle and a latency of 3 cycles. Additionally, two `AESenc` instructions can be performed in parallel. `CLMUL` has the same throughput and latency, but only a single `CLMUL` instruction can be performed at a time [1][6]. Since the `AESenc` or `CLMUL` instructions, are dominant in all three components, the optimizations in this section focus on amortizing the latency over as many instruction executions as possible.

Implementation of GCTR-3 with $\mathsf{ButterKnife}_8$. $\mathsf{ButterKnife}_8$ has 3 major components to be implemented: the tweakey-schedule, the first 7 rounds of encryption and the 8 8-round branches in feed-forward mode. The tweakey-schedule

[6] Raptor cove is the successor of golden cove. Since there are no changes to the instruction pipelines, we actually use the results for golden cove.

is implemented in a straightforward manner. The key-part of the schedule is precomputed before enciphering and therefore has no impact on the performance. The tweak-part of the schedule is computed during enciphering and adds a constant factor to the runtime of the scheme.

The major bottleneck is therefore efficiently evaluating the 71 `AESenc` calls required to compute $\mathsf{ButterKnife}_8$. To fill up the `AESenc` pipelines and thus encrypt at an amortized 2 rounds per cycle, multiple independent encryptions have to be performed at the same time. To guarantee independent instruction, the evaluation of the first 7 rounds of multiple $\mathsf{ButterKnife}_8$ are batched together. Since the branches are already independent, they are then evaluated in order. In an ideal implementation, where only the `AESenc` instructions impact the performance, GCTR-3 with $\mathsf{ButterKnife}_8$ has a performance impact of $(71/2 \text{ cycles})/(8 \cdot 16 \text{ bytes}) \approx 0.28$ cycles/byte on raptor cove.

Implementation of SoCTR. For optimal performance, SoCTR with AES has to be implemented such that multiple encryptions are performed in parallel. This is straightforward as CTR mode is already trivially parallelisable and SoCTR consists of two CTR modes evaluated in parallel. The amortized performance impact of SoCTR with AES is exactly twice that of the CTR mode. On raptor cove that would be 0.625 cycles/byte with AES-128 and 0.875 cycles/byte with AES-256.

Implementation of 256-bit polynomial hash. While a 256-bit polynomial hash has been proposed before in [3], we present a different approach that simplifies the implementation complexity. The polynomial hash is evaluated over the field $\mathbb{F}_{2^{256}} \cong \mathbb{F}_{2^{128}}[y]/(y^2+xy+1)$ where $\mathbb{F}_{2^{128}} \cong \mathbb{F}_2[x]/(x^{128}+x^7+x^2+x+1)$ is the field of GHASH. Defining $\mathbb{F}_{2^{256}}$ in this way allows us to easily apply the usual polynomial hash optimizations such as aggregated reduction [17] and Karatsuba multiplication for polynomials [25] at the level of $\mathbb{F}_{2^{256}}$ and $\mathbb{F}_{2^{128}}$. Additionally the fast modular reduction for GHASH-128 [16] can still be applied.

At the top level, all powers of the hash key $K = K_2y + K_1$ up to K^n are precomputed. This allows for the aggregation of n polynomial multiplications in $\mathbb{F}_{2^{128}}[y]$ and reduces their sum to $\mathbb{F}_{2^{256}}$ once every n multiplications instead of every multiplication. This aggregation step pairs well with Karatsuba multiplication. In this case, it computes the product of two degree one polynomials in $\mathbb{F}_{2^{128}}[y]$ with three multiplications and 4 additions, instead of the usual 4 multiplications and 1 addition, i.e. $(M_2y + M_1)(K_2y + K_1) = M_2K_2y^2 + ((M_1 + M_2)(K_2 + K_1) + M_2K_2 + M_1K_1)y + M_1K_1$. While, the extra additions would in general slow down the multiplication, they do not all have to be evaluated directly. The addition of the two key halves $K_2 + K_1$, can be precomputed. Additionally, the sum in the coefficient of y can be delayed until right before the polynomial reduction. At the level of multiplications in $\mathbb{F}_{2^{128}}$ the same optimizations can be performed, reducing every multiplication to 3 `CLMUL` instructions.

Since the reduction can be aggregated over a large number of multiplications, the number of `CLMUL` instructions per multiplication in the polynomial ring is the dominant factor in the performance of this 256-bit polynomial hash.

Therefore, in an ideal implementation, the performance impact of this hash is 9 cycles/32 bytes $\approx$ 0.28 cycles/byte on raptor cove. This is only 1.5 times more than GHASH-128's impact of 3 cycles/16 bytes $\approx$ 0.19 cycles/byte.

C.1 Benchmarks

The performance measurments of KohiNoor and DaryaiNoor as well as XCB-AES, HCTR2 and CTET+ are given in Table 4 in cycles-per-byte on Intel's raptor cove and gracemont microarchitectures. The benchmarks are of 512B, 4KiB and 64KiB fixed length implementations of the TEMs. The lengths of 512B and 4KiB reflect the disk encryption application, where sector size are either 512B or 4KiB. The 64KiB length represents the performance of the scheme without constant overhead. Table 4 also provides performance of an ideal implementation of each TEM on the raptor cove and gracemont microarchitectures. This ideal performance assumes that there are two parallel `AESenc` pipelines and a `CLMUL` pipeline, all with a throughput of 1 cycle, as is the case in both microarchitectures. Additionally, it assumes that all constant overhead is negligible.

For a fair comparison to CTET+, all implementations are block-aligned. The code is compiled with the Clang `C++` compiler version 17.0.6 at O3 optimization level and with arch and tune compile options set to native. The measurement results are the median over 100 runs of the cycles per bytes to encrypt 16 MiB with the specified TEM and length. The time is measured using the thread-specific cpu-time clock. The cycles per byte are computed by multiplying the measured time by the frequency of the core and dividing it by 2^{24} bytes. Additionally, the utility `taskset` is used to guarantee that the code is run on a core with the correct microarchitecture.

TEM		Raptor cove			Gracemont		
	Ideal	512B	4KiB	64KiB	512B	4KiB	64KiB
XCB-AES	0.69	0.76	0.70	0.70	1.02	0.95	1.06
HCTR2	0.69	0.81	0.74	0.74	1.07	1.00	1.10
AES_6-CTET+	0.75	0.91	0.90	0.95	1.07	1.04	1.20
AES-CTET+	1.00	1.19	1.16	1.23	1.49	1.53	1.61
KohiNoor	0.84	1.40	0.93	0.87	1.61	1.12	1.08
DaryaiNoor	1.19	1.62	1.29	1.25	1.99	1.68	1.71

Table 4: Performance (in cycles/byte) comparison of XCB-AES, HCTR2, CTET+, KohiNoor and DaryaiNoor for 512B, 4096B and 64KiB blocks on the raptor cove and gracemont microarchitectures as measured on an Intel i7-13700 processor. XCB-AES, HCTR2, AES-CTET+ and DaryaiNoor are instantiated with AES-128.

Table 4 shows that KohiNoor and DaryaiNoor are most performant at longer lengths, where the extra constant overhead of the Feistel rounds, the tweakey schedule and the large field reduction are amortized. At 4 KiB KohiNoor is on par with AES_6-CTET+, and it is 26-33% slower than XCB-AES and HCTR2 on raptor cove and 12-18% slower on gracemont. Even at 512B, KohiNoor is still on par with AES-CTET+, although the gap with XCB-AES widens to around 58% on gracemont and 84% on raptor cove. DaryaiNoor is not as performant as KohiNoor, but for larger block lengths it is still on par with CTET+ while providing higher security with the same components. At the 512B block length, DaryaiNoor is twice as slow as XCB-AES on both microarchitectures, however for larger block lengths its performance improves to being only 61% slower than XCB-AES. This shows that for long messages it is possible to provide full block-size security with same standard components without doubling the runtime of the mode.

Additional benchmarks with AES-256 are provided in Appendix D.

D Additional Benchmarks

We provide additional benchmarks of the relevant modes instantiated with AES-256 in Table 5.

TEM		Raptor cove			Gracemont		
	Ideal	512B	4KiB	64KiB	512B	4KiB	64KiB
XCB-AES	0.81	0.94	0.88	0.87	1.32	1.22	1.30
HCTR2	0.81	0.95	0.86	0.85	1.33	1.24	1.25
AES-CTET+	1.25	1.50	1.46	1.56	2.02	1.96	2.06
DaryaiNoor	1.44	2.12	1.67	1.62	2.56	2.15	2.12

Table 5: Performance (in cycles/byte) comparison of XCB-AES, HCTR2, CTET+ and DaryaiNoor instantiated with AES-256 for 512B, 4096B and 64KiB blocks on the raptor cove and gracemont microarchitectures as measured on an Intel i7-13700 processor.

E Revisiting XCBv3: Attacks by Wang et al. [34]

In an earlier version of this ePrint, alongside with GEM, we also proposed a simple fix for XCBv2 called XCBv3. XCBv3 is defined as XCBv2 but with two independent hash keys to avoid shared difference attack. We also provided a proof of security for XCBv3.

In a very recent work, Wang et al. [34] proposed new attacks against XCB variants including XCBv3. We confirm that XCBv3 is prone to these attacks which implies that independent hash keys are not sufficient to avoid the exploitation of separability of the used XOR-universal hash functions.

In this section, we revisit our security analysis of XCBv3 and point out the exact flaw in the proof. The flaw lies in the analysis of Bad1.3 which is the same bad case as we identified in Section 4.3 for XCBv2 and XCBv2fb. We quote the bad case and its analysis from previous ePrint version below.

"For any message M_i in Q, we denote the rightmost n-bit block of it by $M_{i,R}$ and the rest of it by $M_{i,L}$. Similarly, for any ciphertext C_i in Q, the rightmost n-bit block is denoted by $C_{i,R}$ and the rest by $C_{i,L}$. We now define three internal variables for XCBv3$'[H, \mathsf{CTR}, f]$ (similar to the ones depicted in Figure 2 for XCBv2fb but now with different hash keys) as

$$\begin{aligned} X_{i,R} &= f_{e1}(M_{i,R})\,, \\ X_{i,L} &= H_{K_{h_1}}(0^n \| T_i \| M_{i,L} \| 0^n)\,, \\ Y_{i,R} &= f_{d1}(C_{i,R})\,, \\ Y_{i,L} &= H_{K_{h_2}}(T_i \| 0^n \| C_{i,L} \| \mathsf{len}_{m_i,t_i})\,, \\ Z_i &= X_{i,L} \oplus X_{i,R} = Y_{i,L} \oplus Y_{i,R}\text{"} \end{aligned} \tag{18}$$

"$\mathsf{Bad1.3}:\ 1 \leq i \leq q_e < j \leq q_e + q_d$ and $X_{i,L} \oplus Y_{j,L} = X_{i,R} \oplus Y_{j,R} \oplus \mathsf{bin}_n(c_i) \oplus \mathsf{bin}_n(c_j)$."

Analysis: "Since there are no trivial queries in Q, we also have that for $1 \leq i \leq q_e < j \leq q_e + q_d, (M_i, T_i) \neq (M_j, T_j), (C_i, T_i) \neq (C_j, T_j)$ and thus $X_{i,L} \oplus Y_{j,L} = Y_{i,R} \oplus Y_{j,R} \oplus \mathsf{bin}_n(c_i) \oplus \mathsf{bin}_n(c_j)$ which is same as $H_{K_{h_1}}(0^n \| T_i \| M_{i,L} \| 0^n) \oplus H_{K_{h_2}}(T_j \| 0^n \| C_{j,L} \| \mathsf{len}_{m_j,t_j}) = f_{e1}(M_{i,R}) \oplus f_{d1}(C_{j,R}) \oplus \mathsf{bin}_n(c_i) \oplus \mathsf{bin}_n(c_j)$ can occur with probability at most $\max\{\ell_i, \ell_j\}/2^n$."

We note that with two independent hash keys the relation above (in blue) appears to be captured by the definition of a universal hash and thus the estimated probability upper bound of $\max\{\ell_i, \ell_j\}/2^n$. However, the attack 2 (and its extension; attack 3) from [34] shows that this bound is incorrect. In fact, this probability can be shown 1 when $i \geq 2, j \geq 2$.

More formally, a generalization of their attack 2 can be described as follows - Let us consider $c_i = c_j = 0$ and $|C_i| = |M_i|$ then the above relation can be rewritten as

$$H_{K_{h_1}}(0^n \| T_i \| M_{i,L} \| 0^n) \oplus H_{K_{h_2}}(T_j \| 0^n \| C_{j,L} \| \mathsf{len}_{m_j,t_j}) = f_{e1}(M_{i,R}) \oplus f_{d1}(C_{j,R})\,. \tag{19}$$

Now, considering the scenario where the adversary is adaptive and has already made three another queries using some arbitrary constants α and β with $|\alpha| + n = |C_j|$ and $|\beta| = |T_j|$ as

- Deciphering with input $(C_j \oplus \alpha \| 0^n, T_j \oplus \beta)$ to get the output $M_{j'}$,
- Enciphering with input $(M_{j'} \oplus \alpha \| 0^n, T_j)$ to get the output $C_{j'}$,
- Deciphering with input $(C_{j'} \oplus \alpha \| 0^n, T_j \oplus \beta)$ to get the output $M_{j''}$,

and has chosen the targeted i^{th} query as $(M_i, T_i) = (M_{j''} \oplus \alpha, T_j)$, then we have

$$
\begin{aligned}
& f_{e1}(M_{i,R}) \oplus f_{d1}(C_{j,R}) \\
& = f_{e1}(M_{i,R}) \oplus (f_{e1}(M_{j',R}) \oplus H_{K_{h_1}}(0^n \| (T_j \oplus \beta) \| M_{j',L} \| 0^n) \\
& \quad \oplus H_{K_{h_2}}((T_j \oplus \beta) \| 0^n \| (C_{j,L} \oplus \alpha) \| \mathsf{len}_{m_j,t_j})) \\
& = f_{e1}(M_{i,R}) \oplus (f_{e1}(M_{j',R}) \oplus H_{K_{h_1}}(0^n \| T_j \| (M_{j',L} \oplus \alpha) \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(T_j \| 0^n \| (C_{j',L} \| \mathsf{len}_{m_j,t_j})) \oplus H_{K_{h_1}}(0^n \| \beta \| \alpha \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(\beta \| 0^n \| (C_{j,L} \oplus C_{j',L} \oplus \alpha) \| 0^n)) \\
& = f_{e1}(M_{i,R}) \oplus (f_{d1}(C_{j',R}) \oplus H_{K_{h_1}}(0^n \| \beta \| \alpha \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(\beta \| 0^n \| (C_{j,L} \oplus C_{j',L} \oplus \alpha) \| 0^n)) \\
& = f_{e1}(M_{i,R}) \oplus H_{K_{h_1}}(0^n \| \beta \| \alpha \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(\beta \| 0^n \| (C_{j,L} \oplus C_{j',L} \oplus \alpha) \| 0^n) \oplus f_{d1}(C_{j',R}) \\
& = f_{e1}(M_{i,R}) \oplus H_{K_{h_1}}(0^n \| \beta \| \alpha \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(\beta \| 0^n \| (C_{j,L} \oplus C_{j',L} \oplus \alpha) \| 0^n) \oplus (f_{e1}(M_{j'',R}) \\
& \quad \oplus H_{K_{h_1}}(0^n \| (T_j \oplus \beta) \| M_{j'',L} \| 0^n) \\
& \quad \oplus H_{K_{h_2}}((T_j \oplus \beta) \| 0^n \| (C_{j',L} \oplus \alpha) \| \mathsf{len}_{m_j,t_j})) \\
& = f_{e1}(M_{i,R}) \oplus f_{e1}(M_{j'',R}) \oplus H_{K_{h_1}}(0^n \| T_j \| (M_{j'',L} \oplus \alpha) \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(T_j \| 0^n \| C_{j,L} \| \mathsf{len}_{m_j,t_j}) \\
& = H_{K_{h_1}}(0^n \| T_j \| M_{i,L} \| 0^n) \\
& \quad \oplus H_{K_{h_2}}(T_j \| 0^n \| C_{j,L} \| \mathsf{len}_{m_j,t_j}) \,.
\end{aligned}
$$

Here the first, third and fifth equations hold from the Exp. 18, the second and sixth equations hold from the separability of $H_{K_{h_1}}$ and $H_{K_{h_2}}$ and the last equation holds because $(M_i, T_i) = (M_{j''} \oplus \alpha, T_j)$. This means the probability that Exp. 19 holds is 1.

Remark 1 – Improving the Attack. We note that the attack 2 as described in [34] requires two different tweaks for the attack to work and hence breaks the STPRP security of targeted XCB variants. However, as captured by the simple generalization above their attack can also be applied with same tweak (by setting β zero) yet different inputs (i.e., α non-zero) and thus can also be used to even break the basic SPRP security of the targeted XCB variants.

Remark 2 – Inapplicability to GEM. We note that GEM mode is not vulnerable to this attack, as it replaces hash functions with stronger primitives—VIL-PRFs—which, by definition, are not separable.