IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 May 2024, accepted 7 June 2024, date of publication 14 June 2024, date of current version 21 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3414963

APPLIED RESEARCH

Context-Committing Authenticated Encryptions Using Tweakable Stream Cipher

DONGHOON CHANG[1,2,3] **AND MUNAWAR HASAN**[1,2]
[1]Indraprastha Institute of Information Technology Delhi, New Delhi 110020, India
[2]National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
[3]Strativia, USA

Corresponding author: Munawar Hasan (munawarh@iiitd.ac.in)

ABSTRACT Committing security of authenticated encryption schemes is an emerging area and an active field of research and is highly motivated by real-world scenarios. CMT-4 security of authenticated encryption scheme is a security notion, where an adversary must create two distinct tuples, each containing a key, a nonce, an associated data and a message for the encryption sub-routine of the authenticated encryption scheme, such that outputs produced by the encryption sub-routine for the two tuples are the same. In this paper, we analyze CMT-4 security of four tweakable wide block cipher schemes HBSH, HCTR2, double-decker and docked-double-decker under encode-then-encipher paradigm by prepending zeros, and present CMT-4 attacks with $O(1)$ time complexity for all the four schemes. We introduce the notion of tweakable stream cipher (tS in short) with the property of partial collision resistance, and use it to create four new tweakable wide block cipher schemes: HBtSH, HtS, tS-double-decker and tS-docked-double-decker. These four proposed schemes can be used to create a CMT-4 secure authenticated encryption scheme with the property of partial collision under encode-then-encipher paradigm. Further, we provide security proof with partial collision resistance for the four proposed schemes against a CMT-4 adversary.

INDEX TERMS Authenticated encryption, context commitment, double-decker, docked-double-decker, disk encryption, HBSH, HCTR2, tweakable stream cipher.

I. INTRODUCTION

A tweakable wide block cipher scheme [1] takes three inputs: a key, a tweak and a plaintext, and produces a ciphertext as the output, where size of the ciphertext is equal to size of the plaintext. The scheme is particularly useful in applications like disk encryption. The disk is partitioned into sectors, typically of size 512 bytes; in some newer hard disks and solid state devices using advance format, sector size of 4096 bytes are used. Each of these sectors have a permanent index that identifies them uniquely. Encrypting a disk requires encryption of each of these sectors. To enhance user experience and performance, preference might be given to an algorithm that considers encryption of a sector mutually exclusive with the encryption of other sectors of the disk. Further, when content of a sector changes, then the sector is encrypted again. HBSH [2], a tweakable wide block cipher scheme, emerged as one of the major work in this area. Adiantum (HBSH specification) is available in Android 9.0 and higher for the encryption [3]. HCTR2 [4] is another tweakable wide block cipher scheme, that is available in Android 14.0 and higher for file encryption applications [5]. Double-decker and docked-double-decker [6] are two deck function (doubly-extendable cryptographic keyed function) based tweakable wide block cipher schemes targeting file encryption applications. In [7], the authors present two instances of docked-double-decker based on AES [8], and hence, it can make use of existing cryptographic hardware accelerators. Another target area of the tweakable wide block cipher scheme is the design space of IoT devices, such devices are computationally menial when compared to desktop or smartphone grade chips, and often lack cryptographic hardware instructions. Schemes like HBSH, double-decker and docked-double-decker are useful in such scenarios.

Encode-then-encipher [9] (EtE in short) paradigm can be used to construct an authenticated encryption scheme from a tweakable wide block cipher scheme by inserting zeros at a specified position. An authenticated encryption

The associate editor coordinating the review of this manuscript and approving it for publication was Mahdi Zareei.

scheme provides both privacy (or confidentiality) and authenticity (or integrity) of data simultaneously. This dual functionality prevents unauthorized access to the encrypted data while also detecting any attempts at tampering or modification of the data. Due to this streamline and efficient process, authenticated encryption schemes are widely used for numerous security critical applications like including secure communication protocols, data storage, file encryption, blockchain and cryptocurrencies, cloud services etc. We target the notion of committing security of authenticated encryption scheme, which is inspired by real-world attacks and is an active area of research. Further, NIST proposal on the requirements for an accordion mode mentions key commitment and context commitment as the desirable properties [10]. AEZ [11], a CAESAR [12] submission, is an authenticated encryption scheme that uses zero appending technique together with nonce, associated data and length of the append (stretch) as the tweak, resulting into a ciphertext that is longer by the length of the append. During the third NIST workshop on block cipher modes of operation [13], key commitment security with time complexity $O(1)$ was presented on AEZ [14], and an overview of CMT-4 attack was also presented on HBSH and HCTR2 under encode-then-encipher paradigm. Facebook's message franking technique in their end-to-end encrypted messenger for generating cryptographically verifiable report for shared images and video, based on AES-GCM [15], was shown to be vulnerable to key commitment attack, due to non-key committing authenticated encryption scheme [16], [17]. Facebook uses a hash of the AES-GCM ciphertext, along with a randomly generated value, as an identifier for the attachment. The attack in [16], finds two different keys and a ciphertext efficiently for a message, such that one of the key decrypts the ciphertext to an abusive attachment while the other key successfully decrypts the same ciphertext, but to another harmless attachment. The adversary, then sends two messages with different keys but the same attachment ciphertext, making Facebook's deduplicate algorithm to report only non-abusive attachment. This vulnerability was patched by making deduplicate algorithm more vigilant. In [18],[1] the authors showed that a key commitment lacking AES-GCM based ciphertext can be decrypted into two plaintexts of different file formats such as PDF, Windows executable, DICOM etc.

In this paper, we explore the idea of creating an authenticated encryption scheme using HBSH, HCTR2, double-decker and docked-double-decker under encode-then-encipher paradigm. For message encoding, we use zero prepending technique, and then for enciphering, we use the four tweakable wide block cipher schemes. Under this EtE setup, we evaluate CMT-4 security of each of these authenticated encryption schemes, and provide detailed CMT-4 attack with time complexity $O(1)$ for all the four schemes. We introduce the notion of tweakable stream cipher (or tS in short) and use tS to create four new tweakable wide block cipher schemes that are CMT-4 secure .i.e., provide partial collision resistance under encode-then-encipher paradigm.

[1]https://github.com/kste/keycommitment

A. MOTIVATION

The idea of context commitment security starts with the sub-routine $\mathcal{E}$ committing to a key K. A CMT-1 adversary generates two distinct tuples (K_1, N_1, A_1, M_1) and (K_2, N_2, A_2, M_2) .i.e., $\tau = \{(K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2)\}$, such that $\mathcal{E}(K_1, N_1, A_1, M_1) = \mathcal{E}(K_2, N_2, A_2, M_2)$ but $K_1 \neq K_2$ .i.e., that two keys K_1 and K_2 are distinct. Since, $\mathcal{E}$ is committing to only one of the input parameters .i.e., the key, hence we use the notation as CMT-1. In a more generic sense, we can use the notation CMT-ℓ, where ℓ denotes the number of inputs of $\mathcal{E}$, to show the commitment of the encryption sub-routine on the number of inputs it takes respectively. In case of CMT-4, the task of adversary is to generate $\tau = \{(K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2)\}$, such that $\mathcal{E}(K_1, N_1, A_1, M_1) = \mathcal{E}(K_2, N_2, A_2, M_2)$ but $(K_1, N_1, A_1, M_1) \neq (K_2, N_2, A_2, M_2)$. Intuitively, a CMT-4 secure $\mathcal{E}$ requires commitment to all of its inputs. Further, we can also see that CMT-4 $\implies$ CMT-1 [19]. CMT-4 security is a strong goal, since commitment is required for all the four inputs of the encryption sub-routine.

HBSH [2], HCTR2 [4] and the two deck based schemes .i.e., double-decker and docked-double-decker [6] target applications like file encryption. Further, HBSH, double-decker and docked-double-decker target the area of low powered devices where cryptographic hardware instructions are absent. Using EtE paradigm, one can convert these tweakable wide block cipher scheme to an authenticated encryption scheme. The authenticated encryption schemes so created must provide security assurance under various notions like committing security. An overview of CMT-4 attack was presented [14] for HBSH and HCTR2 under encode-then-encipher paradigm. We take inspiration from this overview of CMT-4 attack and present detailed CMT-4 analysis with time complexity $O(1)$, for all the four tweakable wide block cipher schemes under EtE paradigm. We further, try to investigate the design of the four tweakable wide block cipher schemes, and introduce the notion of tweakable stream cipher (tS). We use tweakable stream cipher to create four new constructions of tweakable wide block cipher schemes that are CMT-4 secure, when used for creating authenticated encryption schemes by prepending zeros.

B. CONTRIBUTION

- We analyze CMT-4 security of four tweakable wide block cipher schemes under the encode-then-encipher (EtE) paradigm by prepending zeros. An overview of CMT-4 attack of EtE-HBSH and EtE-HCTR2 was presented at the third NIST workshop on the block cipher modes of operation [13]. We present the algorithm for CMT-4 attack on EtE-HBSH and EtE-HCTR2 with time complexity $O(1)$. Further, we present two new

results of CMT-4 attack under EtE paradigm of the two deck function based schemes .i.e., EtE-double-decker and EtE-docked-double-decker, with time complexity $O(1)$.
- We introduce the notion of a tweakable stream cipher (tS) with the property of partial collision resistance. Tweakable stream cipher takes three inputs: a key, a nonce and a tweak and produces a key stream as the output. The key stream can be used for message encryption (or decryption in case of ciphertext). Further, we demonstrate the procedure to create a tweakable stream cipher using eXtendable-Output Function (XOF).
- We present four new tweakable wide block cipher schemes based on a tweakable stream cipher, namely HBtSH, HtS, tS-double-decker and tS-docked-double-decker. These proposed schemes provide CMT-4 secure constructions of authenticated encryption schemes under encode-then-encipher paradigm. We provide CMT-4 security proof to prove our claim.

C. ORGANIZATION OF THIS PAPER

The rest of the paper is organized as follows. In section II, we present the related work in the area of encode-then-encipher paradigm and the research around context commitment security. Section III, is divided into two parts. In section III-A, we present the notations used in this paper. In section III-B, we present definitions that we use in the paper. Section IV presents the four tweakable wide block cipher schemes and their corresponding EtE versions. In section V, we present CMT-4 attack on the four tweakable wide block cipher schemes under EtE paradigm. In section VI, we present the four new tweakable wide block cipher schemes based on a tweakable stream cipher. We present design rationale of the four new proposed schemes in section VII, followed by the security proof of the proposed schemes in section VIII. We conclude our paper in section IX. In appendix A, we present collision attack on Farfalle [20].

II. RELATED WORK

Bellare and Rogaway presented a method for combining confidentiality and authenticity, called encode-then-encipher (EtE) [9]. The authors introduced various encoding schemes for the message, and analyzed the enciphering of the encoded message. The EtE paradigm was initially overlooked, but with the advent of CAESAR competition [12], it found traction. A tweakable wide block cipher scheme provides only confidentiality and not authenticity. Using encode-then-encipher paradigm, a tweakable wide block cipher scheme can be converted into an authenticated encryption scheme by inserting 0^λ at a specified position, where $\lambda > 0$. The common methodology is to prepend or append 0^λ. A *robust* (key-committing) authenticated encryption scheme (RAE in short) [11], must provide confidentiality and authenticity for the chosen value of λ. AEZ [11] based on AES [8] round function is an encode-then-encipher construction. AEZ appends 0^λ to the message M and uses a tweak T comprising of a nonce N, associated data A and λ .i.e., $T = (N \parallel A \parallel \lambda)$. Authenticator [11] .i.e., 0^λ, is used to provide integrity under EtE paradigm. We focus on creating authenticated encryption scheme using a tweakable wide block cipher scheme [1]. HBSH [2], HCTR2 [4] and two deck function based schemes .i.e., double-decker and docked-double-decker [6] are popular tweakable wide block cipher schemes that target file encryption and related applications. HBSH, double-decker and docked-double-decker target devices that lack cryptographic hardware instructions like AES. CBC [21] and XTS-AES [22] became quite popular in disk encryption applications like BitLocker [23], yet they were not very popular in the design space of IoT devices, since AES may be slow when hardware support is absent. CBC encrypted binary files was shown to be vulnerable to malleability, leading to arbitrary code execution [24]. XTS is based on ECB mode and hence is also prone to malleability .i.e., changing a bit of ciphertext influences its corresponding plaintext block only. The design space of IoT devices is called lightweight cryptography [12], [25]. There is an associated trade-off between the security and the efficiency in the design space of lightweight cryptography. This field has gained a lot of interest in the resent years, leading to proposal of several authenticated encryption schemes in this category [26], [27], [28], [29] under various security notions. In this paper, we concentrate on creating a CMT-4 secure authenticated encryption scheme from a tweakable wide block cipher under encode-then-encipher paradigm.

Key commitment security analysis on AEGIS [30], one of the winners of CAESAR competition, was presented in [31] with $O(1)$ time complexity. In [32], the authors analyze the committing security of Ascon [26], winner of NIST lightweight competition. KIVR [33] is a new mode that transforms existing authenticated encryption schemes to have CMT-4 security without increasing the ciphertext size. In [34], the authors provide committing security analysis for all the NIST LWC competition finalists (except Grain-128AEAD). In [35], the author describes methodologies to add key commitment property to an authenticated encryption scheme.

III. PRELIMINARIES

We begin by presenting the list of abbreviations in table 1.

A. NOTATIONS

The notions are defined in table 2. Next we define a zero padding function: pad-$0_l(X) = X \parallel 0^v$, where $v \geq 0$ and $(|X| + v) \pmod{l} = 0$. Now, we define another padding function: for a given $X \in \{0,1\}^{<\eta}$:

$$\text{pad}_\eta(X) = \begin{cases} X \parallel 10^{\eta-|X|-1} & \text{if } |X| < \eta \\ \perp & \text{otherwise.} \end{cases} \quad (1)$$

TABLE 1. Abbreviations used in the paper.

Abbreviation	Definition
HBSH	Hash, Block cipher, Stream cipher, Hash.
HCTR2	Updated version of HCTR [36].
tS	Tweakable Stream cipher.
HBtSH	Hash, Block cipher, Tweakable Stream cipher, Hash.
HtS	Hash, Tweakable Stream cipher.
tS-double-decker	Tweakable Stream cipher double-decker.
tS-docked-double-decker	Tweakable Stream cipher docked-double-decker.
EtE	Encode-then-Encipher.
XOF	eXtendable-Output Function.
Deck	Doubly-extendable cryptographic keyed.
bhk	Blinded keyed hash.

TABLE 2. Notations used in the paper.

$\{0,1\}^*$	Set of all finite bit strings including the empty string ϵ.
$\{0,1\}^+$	Set of all finite bit strings excluding ϵ.
$\|X\|$	Bit length of X, where $X \in \{0,1\}^*$. $\|\epsilon\| = 0$.
$X\|\|Y$ (or XY)	Concatenation of X and Y in respective order.
$\text{int}(X)$	Denotes integer representation of X.
$\text{bin}_l(y) = X$	Denotes a unique l bit sequence such that $\text{int}(X) \equiv y \pmod{2^l}$.
0^i (or 1^i)	String of i zero (or one) bits, for example: $1 \parallel 0^i$ or 10^i refers to bit 1 followed by i zero bits.
$X[a;l]$	Subsequence of X of length l and starting at index a (one-based indexing), where $X \in \{0,1\}^*$.
$X \oplus Y$	Bitwse xor between X and Y where $\|X\| = \|Y\|$.
$\mathbb{Z}/2^{128}\mathbb{Z}$	Group generated by $\pmod{2^{128}}$.
$\boxplus, \boxminus$	Addition and subtraction inside group.
$\mathbb{N}$	The set of all natural numbers.

B. SECURITY NOTIONS AND DEFINITIONS

We describe several concepts and definitions in this section, that are used in this paper. Let $\mathcal{K}$ denote the key space, $\mathcal{N}$ denote the nonce space, the message space is denoted by $\mathcal{M}$, and the tweak space is denoted by $\mathcal{T}$. $K \xleftarrow{\$} \mathcal{K}$ denotes a randomly selected key K from the key space $\mathcal{K}$. We follow the security notions of [1]. Let $\mathcal{A}$ be a computationally bounded adversary. We denote t as the run time taken by $\mathcal{A}$, t includes two things; memory occupied by $\mathcal{A}$ and the time to answer all the queries that $\mathcal{A}$ makes to the oracle. We know that the security of a scheme under various notions is outlined by the advantage that an adversary has on that scheme. We denote advantage as 'max' function over all the adversaries that has some restricted access to the oracle. We first start with the definition of *ϵ-almost-Δ-universal* family of hash functions. Then we define the primitives used in this paper with the advantages the adversary has over these primitives.

Definition 1 (ϵ-Almost-Δ-Universal (ϵ-ΔU) Family of Hash Functions): Let H be a family of hash functions with domain D and range R. Further, let R be an Abelian group where '$-$' denotes the group subtraction operation. Let ϵ be a constant such that $\frac{1}{|R|} \leq \epsilon < 1$. Then, for two distinct inputs $x \in D$ and $y \in D$, and for any $b \in R$, H is called ϵ-almost-Δ-universal (ϵ-AΔU or in short ϵ-ΔU) family of hash functions [37], if we have:

$$Pr[h(x) - h(y) = b : h \xleftarrow{\$} H] \leq \epsilon. \tag{2}$$

Definition 2 (Blinded Keyed Hash (bhk) [6]): Let H be a keyed hash, such that $H = \{H_K : \{0,1\}^* \to \{0,1\}^n | K \xleftarrow{\$} \mathcal{K}\}$. For two inputs (X, Δ), let $\mathcal{R} : \mathcal{O}_1(H_K(X) \oplus \Delta)$ be the real world oracle, and $\mathcal{I} : \mathcal{O}_1(X, \Delta)$ be the ideal world oracle, where $\mathcal{O}_1$ and $\mathcal{O}_2$ are two secret random oracles, then the advantage for an adversary A for the blinded keyed hash or bhk is given by:

$$\begin{aligned} \text{Adv}_H^{\text{bhk}}(A) &= |Pr[A^{\mathcal{I}} \mapsto 1] - Pr[A^{\mathcal{R}} \mapsto 1]| \\ \text{Adv}_H^{\text{bhk}}(q,\sigma) &= \max_{A \in \mathcal{A}(q,\sigma)} \text{Adv}_H^{\text{bhk}}(A), \end{aligned} \tag{3}$$

where $\mathcal{A}(q,\sigma)$ is an adversary that makes at most q with data complexity $\sigma = \sum_i^q |X_i| + |\Delta_i|$.

Definition 3 (Tweakable Stream Cipher): We introduce the notion of tweakable stream cipher (tS in short). A Tweakable Stream Cipher is defined as $S : \mathcal{K} \times \mathcal{N} \times \mathcal{T} \to \{0,1\}^{l_S}$, where $\mathcal{K}$ is the key space, $\mathcal{N}$ is the nonce space, $\mathcal{T}$ is the tweak space and l_S denotes the maximum output length. S generates a key stream for encryption of the message (or decryption of the respective ciphertext). Let q_i be the i^{th} query that consists of a tuple $(N_i, T_i, l_i) \in (\mathcal{N} \times \mathcal{T} \times \mathbb{N})$, where $0 < l_i \leq l_S$, and the adversary A receives $S_K(N_i, T_i)[1; l_i]$ as the response of the i^{th} query, for a randomly chosen key K. Then we define:

$$\begin{aligned} \text{Adv}_S^{\text{tS}}(A) = Pr[A^{S_K(\cdot,\cdot)[1;\cdot]} \mapsto 1 : K \xleftarrow{\$} \mathcal{K}] \\ - Pr[A^{\mathcal{F}(\cdot,\cdot)[1;\cdot]} \mapsto 1 : \\ \mathcal{F} \xleftarrow{\$} ((\mathcal{N} \times \mathcal{T}) \to \{0,1\}^{l_{\text{tS}}})] \\ \text{Adv}_S^{\text{tS}}(q, l, l', t) = \max_{A \in \mathcal{A}(q,l,l',t)} \text{Adv}_S^{\text{tS}}(A), \end{aligned} \tag{4}$$

where $(\mathcal{N} \times \mathcal{T}) \to \{0,1\}^{l_S}$ denotes the set of all functions from $(\mathcal{N} \times \mathcal{T})$ to $\{0,1\}^{l_S}$. $\mathcal{A}(q, l, l', t)$ is the set of all adversaries that makes at most q queries and takes at most t time to execute, such that, l is the upper bound on total input length .i.e., $\sum_i^q (|N_i| + |T_i|)$ and l' is the upper bound on total output length .i.e., $\sum_i^q l_i$.

Definition 4 (eXtendable-Output Function (XOF)): An eXtendable-Output function takes input a message M from the message space $\mathcal{M}$ and an output length, hence XOF : $\mathcal{M} \times \mathbb{N} \to \{0,1\}^{\leq l_S}$., where l_S is the maximum output length.

In this paper, we propose XOF-based tweakable stream cipher. For a given key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N} \wedge |N| < \eta$, a tweak $T \in \mathcal{T}$, a maximum output length of tweakable stream cipher l_{tS} and a reversible encoding denoted by ENC, a tweakable stream cipher can be constructed using XOF as follows: $S(K, N, T) = \text{XOF}(K \parallel \text{ENC}(N, T), l_S)$. ENC can be written as $(\text{pad}_\eta(N) \parallel T)$ for a nonce N with $|N| < \eta$, a tweak T and the padding function defined in equation (1).

Definition 5 (Block Cipher): Let E be an n bit block cipher that takes a key $K \xleftarrow{\$} \mathcal{K}$ and a message $M \in \mathcal{M}$ as input and

produces a ciphertext $C \in \mathcal{C}$ as the output .i.e., $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and $E^{-1} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$, where $\mathcal{C}$ is the ciphertext space, $|K| = k$, $|M| = n$ and $|C| = n$. Let Perm(n) denote the set of all permutations on $\{0,1\}^n$. Then for a randomly chosen key K, we define the $\pm$PRP advantage that an adversary A has over the block cipher E in the following way:

$$\begin{aligned}\mathrm{Adv}_{E}^{\pm\mathrm{PRP}}(A) &= Pr[A^{E_K, E_K^{-1}} \mapsto 1 : K \xleftarrow{\$} \mathcal{K}] \\ &\quad - Pr[A^{\pi,\pi^{-1}} \mapsto 1 : \pi \xleftarrow{\$} \mathrm{Perm}(n)] \\ \mathrm{Adv}_{E}^{\pm\mathrm{PRP}}(q,t) &= \max_{A \in \mathcal{A}(q,t)} \mathrm{Adv}_{E}^{\pm\mathrm{PRP}}(A), \end{aligned} \tag{5}$$

where $\mathcal{A}(q,t)$ is an adversary that makes at most q queries and takes at most t time to execute.

Definition 6 (Tweakable Enciphering Scheme): Let $\boldsymbol{E}$ be a tweakable enciphering scheme defined as $\boldsymbol{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ for a key space $\mathcal{K}$, a tweak space $\mathcal{T}$ and a message space $\mathcal{M}$, such that, for every key $K \in \mathcal{K}$ and tweak $T \in \mathcal{T}$, $\boldsymbol{E}(K,T,\cdot)$ is a length preserving permutation .i.e., $|\boldsymbol{E}(K,T,M)| = |M|$. Further, $\boldsymbol{E}^{-1}(K,T,\boldsymbol{E}(K,T,M)) = M$. Let $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ denote the set of all functions $\boldsymbol{\pi} : \mathcal{T} \times \mathcal{M} \to \mathcal{M}$, such that for $T \in \mathcal{T}$, $M \in \mathcal{M}$, $|\boldsymbol{\pi}(T,M)| = |M|$, and $\boldsymbol{\pi}(T,\cdot)$ is bijective. Then the $\pm\widetilde{PRP}$ advantage that an adversary A has is given by the following expression:

$$\begin{aligned}\mathrm{Adv}_{\boldsymbol{E}}^{\pm\widetilde{\mathrm{PRP}}}(A) &= Pr[A^{\boldsymbol{E}_K, \boldsymbol{E}_K^{-1}} \mapsto 1 : K \xleftarrow{\$} \mathcal{K}] \\ &\quad - Pr[A^{\boldsymbol{\pi},\boldsymbol{\pi}^{-1}} \mapsto 1 : \\ &\quad \boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})] \\ \mathrm{Adv}_{\boldsymbol{E}}^{\pm\widetilde{\mathrm{PRP}}}(q, l_T, l_M, t) &= \max_{A \in \mathcal{A}(q,l_T,l_M,t)} \mathrm{Adv}_{\boldsymbol{E}}^{\pm\widetilde{\mathrm{PRP}}}(A), \end{aligned} \tag{6}$$

where $\mathcal{A}(q, l_T, l_M, t)$ is the set of all adversaries that make at most q queries, total tweak length is at most l_T, and total message length is at most l_M, and take at most t time.

Algorithm 1 Game Partial Collision Resistance (PCR): $\mathbf{G}_H^{\mu\text{-PartialColl}}(\mathcal{A})$

1: $\{X_1, X_2\} \xleftarrow{\$} \mathcal{A}$ ▷ $\{X_1, X_2\} \in \mathcal{X}$
2: $Y_1 \leftarrow H(X_1)$ ▷ $Y_1 \in \mathcal{Y}$
3: $Y_2 \leftarrow H(X_2)$ ▷ $Y_2 \in \mathcal{Y}$
4: **if** $Y_1[1;\mu] = Y_2[1;\mu]$ **then** ▷ Condition for Partial Collision
5: **return** *True*
6: **else**
7: **return** *False*
8: **end if**

Definition 7 (Authenticated Encryption (AEAD)): An authenticated encryption with associated data (in short AEAD) is a family of algorithms denoted by $AEAD = (\mathcal{E}, \mathcal{D})$ that consists of an encryption sub-routine $\mathcal{E}$ and a decryption sub-routine $\mathcal{D}$. The encryption sub-routine takes input as a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, an associated data $A \in \mathcal{M}$, a message $M \in \mathcal{M}$ and produces output as a ciphertext $C \in \mathcal{C}$ such that:

Algorithm 2 Game CMT-4: $\mathbf{G}_{\mathrm{AEAD}}^{\mathrm{CMT\text{-}4}}(\mathcal{A})$, Where AEAD=$(\mathcal{E}, \mathcal{D})$

1: $\{(K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2)\} \xleftarrow{\$} \mathcal{A}$
2: $C_1 \leftarrow \mathcal{E}(K_1, N_1, A_1, M_1)$
3: $C_2 \leftarrow \mathcal{E}(K_2, N_2, A_2, M_2)$
4: **if** $(K_1, N_1, A_1, M_1) = (K_2, N_2, A_2, M_2)$ or $C_1 \neq C_2$ **then**
5: **return** *False*
6: **else**
7: **return** *True*
8: **end if**

$$\begin{aligned}\mathcal{E} &: (K, N, A, M) \to C \\ \mathcal{D} &: (K, N, A, C) \to M / \perp \end{aligned} \tag{7}$$

where:

$$\mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M)) \to M.$$

Definition 8 (Partial Collision Resistance): Let $H : \mathcal{X} \to \mathcal{Y}$ be a function with $\mathcal{X}$ as domain space and $\mathcal{Y}$ as range space, where $\mathcal{X}$ is the set of tuples of inputs and $\mathcal{Y}$ is the set of outputs. For an adversary $\mathcal{A}$, we define the advantage of breaking the μ bit partial collision resistance of the function H using Algorithm 1, in the following way:

$$\mathrm{Adv}_H^{\mu\text{-PartialColl}}(\mathcal{A}) = Pr[\mathbf{G}_H^{\mu\text{-PartialColl}}(\mathcal{A}) \mapsto 1]. \tag{8}$$

Definition 9 (CMT-4 Security of AEAD): Let AEAD be an authenticated encryption scheme with associated data defined as $AEAD = (\mathcal{E}, \mathcal{D})$. Then CMT-4 security of AEAD is given by the game in algorithm 2. For an adversary $\mathcal{A}$, the CMT-4 advantage on AEAD is given by the following equation:

$$\mathrm{Adv}_{\mathrm{AEAD}}^{\mathrm{CMT\text{-}4}}(\mathcal{A}) = Pr[\mathbf{G}_{\mathrm{AEAD}}^{\mathrm{CMT\text{-}4}}(\mathcal{A}) \mapsto 1]. \tag{9}$$

Definition 10 (Encode-then-Encipher (EtE) [9]): An AEAD takes four inputs .i.e., a key K, a nonce N, an associated data A and a message M. AEAD can be constructed under EtE paradigm using a two step approach. The first step is to use message encoding, followed by message enciphering. We use a tweakable enciphering scheme (definition 6) for message enciphering. Let μ be the number of zero bits that are used for the prepending technique, then we define message encoding and message enciphering as follows:

1) MESSAGE ENCODING

We borrow the notation of encoding the messages from [9]. Let $\mathcal{M}$ be the message space and let $\mathcal{M}^*$ be the encoding space.

- We present the Encode sub-routine in equation (10) below:

$$M' = \mathrm{Encode}(M, \mu) = \begin{cases} (0^\mu \parallel M) & \text{if } \mu > 0 \\ \perp & \text{otherwise.} \end{cases} \tag{10}$$

- The Decode sub-routine in equation (11) below:

$$\text{Decode}(M', \mu) = \begin{cases} M & \text{if } \mu > 0 \wedge (M' = (0^{\mu} \parallel M)) \\ \perp & \text{otherwise.} \end{cases} \tag{11}$$

2) MESSAGE ENCIPHERING

After the message is encoded using the Encode sub-routine of equation (10), a tweakable enciphering scheme $\boldsymbol{E}$ (definition 6) is invoked to encrypt M' and generate a ciphertext C, .i.e,

$$\boldsymbol{E}(K, T, M') \mapsto C,$$

where K is the key of the AEAD and nonce and associated of AEAD is used to create the tweak .i.e., $T \leftarrow (N \parallel A)$.

In case of decryption, $\boldsymbol{E}^{-1}$ is invoked to recover back M' from C .i.e.,

$$\boldsymbol{E}^{-1}(K, T, C) \mapsto M',$$

where K is the key of the AEAD and the tweak is given by: $T \leftarrow (N \parallel A)$. Decode sub-routine of equation (11) is then used to get back the original message M. If the first μ bits of M' are non-zero, then we reject the message M and output $\perp$ (error).

IV. TWEAKABLE WIDE BLOCK CIPHER SCHEMES AND EtE PARADIGM

In this section, we discuss four tweakable wide block cipher schemes and present their respective EtE versions. We describe HBSH in section IV-A and present EtE-HBSH in section IV-A1. HCTR2 is describe in section IV-B while EtE-HCTR2 is presented in section IV-B1. The two deck function based schemes .i.e., double-decker and docked-double-decker are described section IV-C, EtE-double-decker and and EtE-docked-double-decker are presented in section IV-C3.

A. HBSH

Hash Block cipher Stream cipher Hash (or HBSH) [2] is shown in figure 1. HBSH is a tweakable wide cipher scheme [1] and resembles an unbalanced Feistel structure [38]. It takes in three inputs: a key K from the key space $\mathcal{K} \in \{0,1\}^k$, a tweak T from the tweak space $\mathcal{T} \in \bigcup_{i=0}^{l_S}\{0,1\}^i$ and a message (or plaintext) P from the message space $\mathcal{M} \in \bigcup_{i=n}^{i=l_S}\{0,1\}^i$. During the invocation of encryption sub-routine, the plaintext P is first split into two parts, P_L and P_R respectively, where $(P_L \parallel P_R) \leftarrow P$ and $|P_R| = n$. Hence, $|P| \geq n$. The ciphertext C, is also generated by the encryption sub-routine in two parts (refer figure 1), .i.e., $(C_L \parallel C_R) \leftarrow C$. HBSH derives three keys K_H, K_E and K_S from K; K_H is used for the hash function H, K_E is used for the block cipher and K_S is used for the stream cipher. Each of these three cryptographic primitives that HBSH uses to generate a ciphertext C from the given plaintext P (or recover P from C) is explained below:

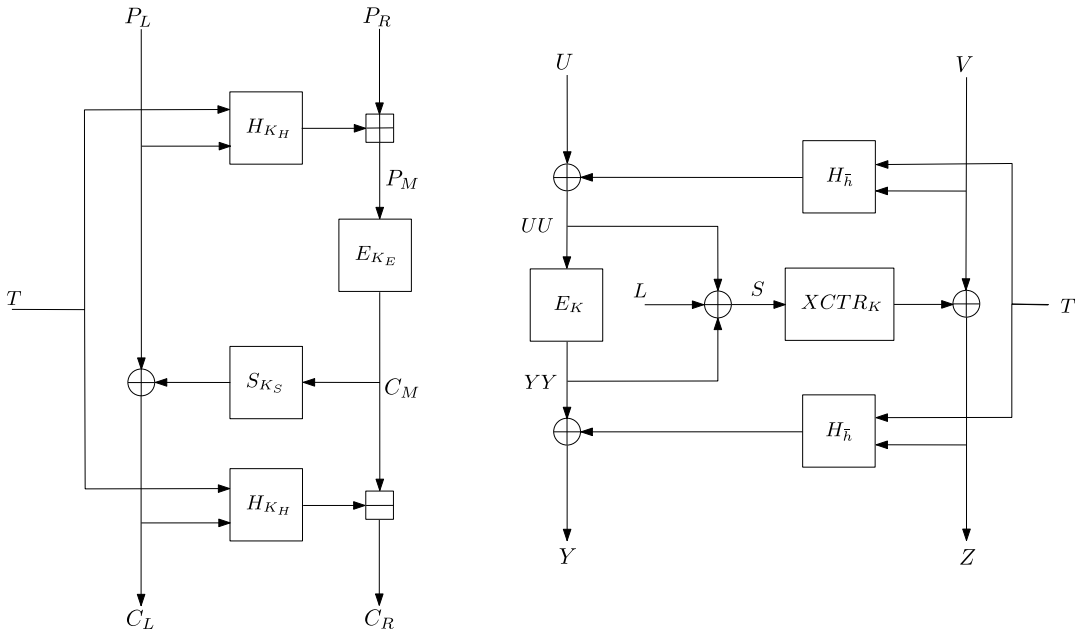
- **Hash:** H is an ϵ-almost-Δ-universal (ϵ-ΔU) function (definition 1) that represents a group element in an n bit string. The group operations $\boxplus$ and $\boxminus$ is computed using $\mathbb{Z}/2^n\mathbb{Z}$. The hash function is invoked two times.
  - For the first invocation, H takes in a tweak T and the left part of the plaintext .i.e., P_L, and uses key K_H to produce a fixed size output. This output is added with P_R .i.e., the right part of the plaintext P, to obtain P_M. Mathematically, $P_M \leftarrow H_{K_H}(T, P_L) \boxplus P_R$ (refer figure 1).
  - During the second invocation, H produces the right part of the ciphertext C (C_R) .i.e., $C_R \leftarrow H_{K_H}(T, C_L) \boxminus C_M$ (refer figure 1), where C_M is the output of the block cipher E and C_L is the left part of the ciphertext.
- **Block cipher:** A single invocation of block cipher (denoted by E) is used with block size n and key K_E (refer figure 1). E produces C_M .i.e., $C_M \leftarrow E_{K_E}(P_M)$. C_M is used as a nonce for the stream cipher.
- **Stream cipher:** In figure 1, S denotes a stream cipher that takes in a key K_S and a nonce C_M and produces a long random stream. This long random stream is *xored* with the left part of the plaintext P_L, producing the left part of the ciphertext C_L respectively.

As mentioned earlier, HBSH derives three keys K_H, K_E and K_S from K using a key derivation function. The stream cipher S is used as the key derivation function by instantiating it with a zero-length nonce .i.e., $K_E \parallel K_H = S_{K_S}(\epsilon)$, where $K_S = K$ and $|\epsilon| = 0$. We now describe Adiantum, a specification of HBSH.

Adiantum: We obtain Adiantum from HBSH when we fix $n = 128$ and $l_S = 2^{73}$. Hence, for the three inputs: a key K, a tweak T and a message P: $n = 128$ and $l_S = 2^{73}$. For the three cryptographic primitives, Adiantum uses XChaCha12 [39] as the stream cipher with key $K_S(= K)$ and nonce C_M from the nonce space $\mathcal{N} \in \bigcup_{i=0}^{191}$; such that, $S_{K_S} = \text{XChaCha12}_{K_S}(\text{pad}_{192}(C_M \parallel 1))$. For the block cipher, Adiantum uses AES256 [8], [40] with key K_E. C_M is generated from the block cipher E_{K_E}, and is used as the nonce after padding it to 192 bits. For the hashing, Adiantum uses combination of NH [41] and Poly1305 [42] with key K_H. Poly1305 is invoked two times, once with NH and once for hashing message length and tweak. Hence, we need a total of three keys for hashing .i.e., $K_T \leftarrow K_H[0:128]$ for Poly1305 which hashes message length and tweak, $K_N \leftarrow K_H[256:8576]$ for NH and $K_L \leftarrow K_H[128:256]$ for Poly1305, there by making a total of $128 + 8576 + 128 = 8832$ bit hash key or K_H. Let $H_T \leftarrow \text{Poly1305}_{K_T}(\text{bin}_{128}(|L|) \parallel T)$ and $H_L \leftarrow \text{Poly1305}_{K_L}(\text{NH}_{K_N}(\text{pad-0}_{128}(L)))$, then the final hash is given by $H_T \boxplus H_L$. Now, using the definition 10, we define EtE-HBSH.

1) ETE-HBSH

In this section, we present an authenticated encryption scheme called EtE-HBSH, based on the tweakable block



(a) HBSH (b) HCTR2

FIGURE 1. HBSH (Hash Block cipher Stream cipher Hash) and HCTR2 (extension of HCTR).

cipher scheme HBSH (refer section IV-A), using encode-then-encrypt paradigm (or EtE in short) [9]. As described in definition 10, constructing such an authenticating encryption scheme is a two step approach, message encoding followed by enciphering. We prepend μ zero bits to the message for message encoding. For enciphering, we use HSBH. We now describe the EtE-HBSH. Let $\mathcal{E}$ be the encryption sub-routine and $\mathcal{D}$ be the decryption sub-routine of EtE-HBSH. $\mathcal{E}$ takes in three inputs, a key K, a tweak T and a message P, where $T \leftarrow (N \parallel A)$ for a fixed size nonce N and associated data A. The message encoding is done using equation (12), where the plaintext P is first split into two parts P_L and P_R; then we create P'_L from P_L by prepending μ zero bits to P_L .i.e., $P'_L \leftarrow (0^\mu \parallel P_L)$, where $\mu > 0$. The tweakable block cipher scheme, HBSH, generates the ciphertext using key, tweak and the encoded message.

$$\text{Encode}(P, \mu) = \begin{cases} P' & \text{if } \mu > 0 \\ & \quad \text{where } P' \leftarrow (P'_L \parallel P_R) \\ & \quad\quad \wedge\, P'_L \leftarrow (0^\mu \parallel P_L) \\ & \quad\quad \wedge\, (P_L \parallel P_R) \leftarrow P \\ \perp & \text{otherwise.} \end{cases}$$

$$\text{Decode}(P', \mu) = \begin{cases} P & \text{if } \mu > 0 \wedge (P'_L = (0^\mu \parallel P_L)) \\ & \quad \text{where } (P'_L \parallel P_R) \leftarrow P' \\ & \quad\quad \wedge\, P \leftarrow (P_L \parallel P_R) \\ \perp & \text{otherwise.} \end{cases} \tag{12}$$

In equation (12), P' is the input of HBSH and P is the input of AEAD .i.e, EtE-HBSH. The decryption sub-routine $\mathcal{D}$, takes in key K, tweak T and the ciphertext C. After the ciphertext is decrypted using HBSH (refer section IV-A), then equation (12) is used to retrieve the message or plaintext. The decryption sub-routine $\mathcal{D}$ of EtE-HBSH returns message if the first μ bits of P'_L are zero, else it returns $\perp$ (error).

B. HCTR2

HCTR2 [4] is an extension of HCTR [36]. From the likes of the Adiantum, HCTR2's primary focus is also on low powered devices and disk encryption scenarios. The encryption sub-routine of HRCR2 takes three inputs: a key K, a tweak T and a message or plaintext P. The key K is used for AES encryption, where $K \in \{\{0,1\}^{128}, \{0,1\}^{192}, \{0,1\}^{256}\}$. $\bar{h} \leftarrow E_K(\text{bin}(0))$, where $\bar{h}$ denotes the key used by the hash function and $|\bar{h}| = 128$. HCTR2 uses polyval for hashing. For a hash key $\bar{h} \in \{0,1\}^n$, a tweak T and a message M, the hash function is defined as follows:

$$H_{\bar{h}}(T, M) = \begin{cases} \text{POLYVAL}\big(\bar{h}, \text{bin}(2|T|+2) \parallel \text{pad}(T) \\ \quad \parallel M\big) & \text{if } n \text{ divides } |M| \\ \text{POLYVAL}\big(\bar{h}, \text{bin}(2|T|+3) \parallel \text{pad}(T) \\ \quad \parallel \text{pad}(M \parallel 1)\big) & \text{otherwise.} \end{cases}$$

Polyval is calculated in the folowing way: $\text{POLYVAL}(\bar{h}, \lambda) = 0^n$ and $\text{POLYVAL}(\bar{h}, A \parallel B) = (\text{POLYVAL}(\bar{h}, A) \oplus B) \otimes \bar{h} \otimes x^{-n}$, where $|\bar{h}| = |B| = n = 128$ and $\otimes$ is multiplication over finite field. The polynomial used for reduction is $x^{128} + x^{127} + x^{126} + x^{121} + 1$ and $x^{-n} = x^{127} + x^{124} + x^{121} + x^{114} + 1$. Let $(U \parallel V) \leftarrow P$ (refer figure 1), where $|U| = n = 128$. Let $L \leftarrow E_k(\text{bin}(1))$, $UU \leftarrow U \oplus H_{\bar{h}}(T, N)$, $YY \leftarrow E_K(UU)$, and $S \leftarrow (UU \oplus YY \oplus L)$. HCTR2 uses *XCTR* as the stream encryption: $\text{XCTR}_K(S) = E_K(S \oplus \text{bin}(1)) \parallel E_K(S \oplus \text{bin}(2)) \parallel E_K(S \oplus \text{bin}(3)) \parallel \ldots$ and so on. The ciphertext C can be

generated as follows: $Z \leftarrow (V \oplus \text{XCTR}_K(S)[1;|V|])$, $Y \leftarrow (YY \oplus H_{\tilde{h}}(T, Z))$, and $C \leftarrow (Y \parallel Z)$.

1) ETE-HCTR2

In case of HCTR2, we know that the plaintext P is divided into two parts .i.e., $(U \parallel V) \leftarrow P$ (refer figure 1), where $|M| = n$. Using definition 10, we now present the EtE-HCTR2. Let $\mathcal{E}$ and $\mathcal{D}$ be the encryption and decryption sub-routine of EtE-HCTR2 respectively, under the encode-then-encipher paradigm. We use Encode sub-routine (refer equation 13) for V during the invocation of $\mathcal{E}$, and Decode sub-routine (refer equation 13) during the invocation of $\mathcal{D}$.

$$\text{Encode}(P,\mu) = \begin{cases} P' & \text{if } \mu > 0 \\ & \quad \text{where } P' \leftarrow (U \parallel V') \\ & \quad\quad \wedge\, V' \leftarrow (0^{\mu} \parallel V) \\ & \quad\quad \wedge\, (U \parallel V) \leftarrow P \\ \perp & \text{otherwise.} \end{cases}$$

$$\text{Decode}(P',\mu) = \begin{cases} P & \text{if } \mu > 0 \wedge (V' = (0^{\mu} \parallel V)) \\ & \quad \text{where } (U \parallel V') \leftarrow P' \\ & \quad\quad \wedge\, P \leftarrow (U \parallel V) \\ \perp & \text{otherwise.} \end{cases} \tag{13}$$

In equation (13), P' is the input of HCTR2 and P is the input of AEAD .i.e, EtE-HCTR2. Similar to EtE-HBSH, the plaintext from the decryption sub-routine is considered valid if and only if, the first μ bits of V' are zero.

C. DECK-BASED WIDE BLOCK CIPHER SCHEMES

In this section we first describe two deck (doubly-extendable cryptographic keyed) function based tweakable wide block cipher schemes [6], namely, double-decker (section IV-C1) and docked-double-decker (section IV-C2). Double-decker (figure 2) is based on Farfalle [20], while docked-double-decker (figure 2) is a slight modification of the double-decker scheme. In section IV-C3, we present the EtE based authenticated encryption schemes of double-decker and docked-double-decker.

1) DOUBLE-DECKER

The construction of double-decker (figure 2) can be seen as a generalized four-round Feistel structure, with F_{K_1} and F_{K_2} as the two deck functions and two invocations of H_K. Double-decker is defined over a key space $\mathcal{K}$, a tweak space $\mathcal{W}$, a message space $\mathcal{M}$. The encryption sub-routine $\mathcal{E}$, takes in three keys $(K, K_1, K_2) \in \mathcal{K}$, a tweak $W \in \mathcal{W}$ and a message $P \in \mathcal{M}$, and produces a ciphertext $C \in \mathcal{C}$, where $|P| = |C|$ and $\mathcal{C}$ is the ciphertext space. The decryption sub-routine $\mathcal{D}$, takes input $(K, K_1, K_2) \in \mathcal{K}$, tweak $W \in \mathcal{W}$ and the ciphertext $C \in \mathcal{C}$ and returns back the output P as the plaintext, if C was generated using (K, K_1, K_2), W and P. From the construction of double-decker in figure 2, we know that the plaintext is divided into four separate parts .i.e., $(U_L \parallel U_R \parallel V_L \parallel V_R) \leftarrow P$, where $|U_L| = |V_R| = n$ and U_R and V_L can be of arbitrary length, while the ciphertext is the combination of four separate parts .i.e. $C \leftarrow (X_L \parallel X_R \parallel Y_L \parallel Y_R)$, where $|X_L| = |Y_R| = n$ and $|X_R| = |U_R|$, and $|Y_L| = |V_L|$. We define EtE-double-decker in section IV-C3.

2) DOCKED-DOUBLE-DECKER

Similar to the construction double-decker (section IV-C1), docked-double-decker (figure 2) can also be seen as a generalized four-round Feistel structure. In this case, the plaintext is split into three separate parts .i.e., $(T \parallel U \parallel V) \leftarrow P$, with $|T| = |V| = n$ and U can be of arbitrary length. The ciphertext is the combination of three separate parts .i.e. $C \leftarrow (X \parallel Y \parallel Z)$, where $|X| = |Z| = n$ and $|Y| = |U|$. From figure 2, we can see that the input length to the two deck functions F_{K_1} and F_{K_2} are fixed, hence, for a fixed tweak length one can conceptualize docked-double-decker as a stream cipher [6]. We now define EtE-double-decker and EtE-docked-double-decker.

3) ETE-DOUBLE-DECKER AND ETE-DOCKED-DOUBLE-DECKER

Using definition 10, we present the encode-then-encipher version of double-decker and docked-double-decker.

- **EtE-Double-Decker:** Let $\mathcal{E}$ and $\mathcal{D}$ be the encryption and decryption sub-routines of EtE-double-decker respectively. From section IV-C1, we know that the plaintext is split into four separate parts .i.e., $(U_L \parallel U_R \parallel V_L \parallel V_R) \leftarrow P$ respectively (refer figure 2). To create encoding for EtE-double-decker from double-decker, we utilize the second split of the plaintext P .i.e., U_R. Invocation of $\mathcal{E}$, starts with the encoding of U_R, by using the Encode sub-routine of equation (14). In case of the decryption sub-routine $\mathcal{D}$, we have an addition step .i.e., Decode sub-routine of equation (14).

$$\text{Encode}(P,\mu) = \begin{cases} P' & \text{if } \mu > 0, \text{ where} \\ & \quad P' \leftarrow (U_L \parallel U'_R \parallel V_L \parallel V_R) \\ & \quad \wedge\, U'_R \leftarrow (0^{\mu} \parallel U_R) \\ & \quad \wedge\, (U_L \parallel U_R \parallel V_L \parallel V_R) \leftarrow P \\ \perp & \text{otherwise.} \end{cases}$$

$$\text{Decode}(P',\mu) = \begin{cases} P & \text{if } \mu > 0 \wedge (U'_R = (0^{\mu} \parallel U_R)) \\ & \quad \text{where,} \\ & \quad (U_L \parallel U'_R \parallel V_L \parallel V_R) \leftarrow P' \\ & \quad\quad \wedge \\ & \quad P \leftarrow (U_L \parallel U_R \parallel V_L \parallel V_R) \\ \perp & \text{otherwise.} \end{cases} \tag{14}$$

In equation (14), P' is the input of double-decker and P is the input of AEAD .i.e, EtE-double-decker.

- **EtE-Docked-Double-Decker:** In case of docked-double-decker, we know that the plaintext is split into three parts .i.e., $(T \parallel U \parallel V) \leftarrow P$. We use U,

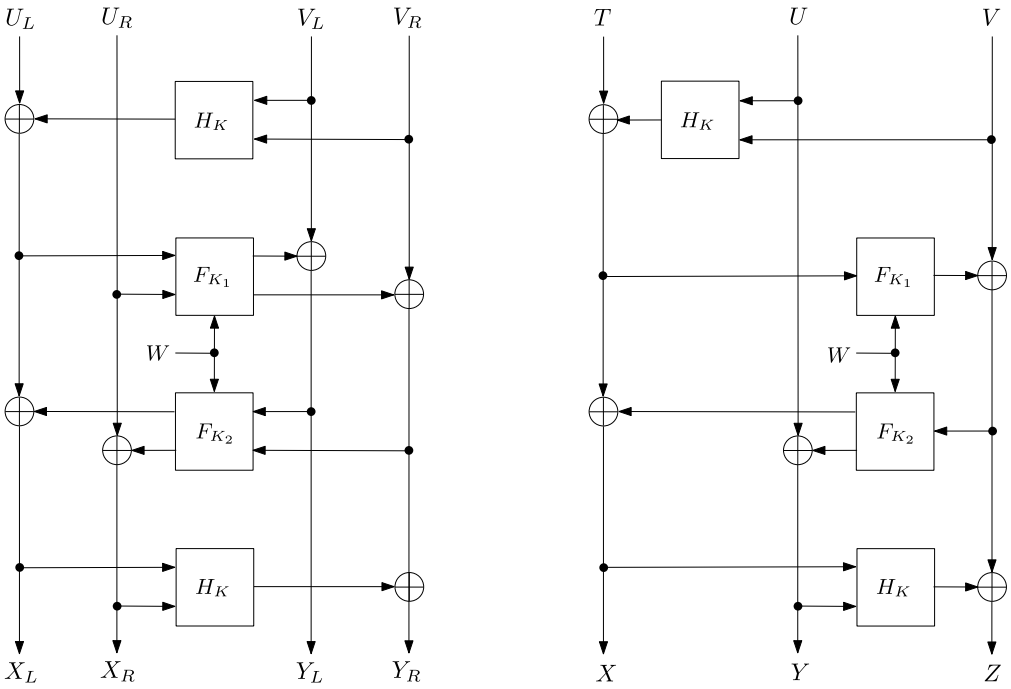


FIGURE 2. Double-decker and Docked-double-decker.

to create EtE-docked-double-decker. Let $\mathcal{E}$ and $\mathcal{D}$ be the encryption and decryption sub-routines of EtE-docked-double-decker, then we use equation (15), for creating encoding and decoding rules of $\mathcal{E}$ and $\mathcal{D}$ respectively.

$$\text{Encode}(P,\mu) = \begin{cases} P' & \text{if } \mu > 0 \text{ where,} \\ & P' \leftarrow (T \parallel U' \parallel V) \\ & \wedge\ U' \leftarrow (0^{\mu} \parallel U) \\ & \wedge\ P \leftarrow (T \parallel U \parallel V) \\ \perp & \text{otherwise.} \end{cases}$$

$$\text{Decode}(P',\mu) = \begin{cases} P & \text{if } \mu > 0 \wedge (U' = (0^{\mu} \parallel U)) \\ & \text{where,} \\ & (T \parallel U' \parallel V) \leftarrow P' \\ & \wedge\ P \leftarrow (T \parallel U \parallel V) \\ \perp & \text{otherwise.} \end{cases} \tag{15}$$

In equation (15), P' is the input of double-decker and P is the input of AEAD .i.e, EtE-double-decker.

In case of EtE-double-decker, the plaintext is accepted if and only if, the first μ bits of U'_R are zero, while for EtE-docked-double-decker, the plaintext is accepted if and only if, the first μ bits of U' are zero.

V. CMT-4 ATTACK ON ENCODE-THEN-ENCIPHER SCHEMES

In this section, we present CMT-4 attack under EtE paradigm for four tweakable wide block cipher schemes, namely EtE-HBSH (section IV-A1, algorithm 3), EtE-HCTR2 (section IV-B1, algorithm 4), EtE-double decker and EtE-docked-double-decker (section IV-C3).

A. CMT-4 ATTACK ON EtE-HBSH

Let $\mathcal{A}$ be a computationally bounded CMT-4 adversary that has access to EtE-HBSH construction, defined in section IV-A1. $\mathcal{A}$ interacts with the encryption sub-routine $\mathcal{E}$ of EtE-HBSH. $\mathcal{E}$ takes in four inputs: a key K, a nonce N, an associated data A and a message P and returns a ciphertext C. The task of $\mathcal{A}$ is to create two distinct tuples (K_1, N_1, A_1, P_1) and (K_2, N_2, A_2, P_2) .i.e., $\tau = \{(K_1, N_1, A_1, P_1), (K_2, N_2, A_2, P_2)\}$ such that $(K_1, N_1, A_1, P_1) \neq (K_2, N_2, A_2, P_2)$ but $\mathcal{E}(K_1, N_1, A_1, P_1) = \mathcal{E}(K_2, N_2, A_2, P_2)$ .i.e., the ciphertext C generated by the encryption sub-routine for the two distinct tuples are the same. For EtE-HBSH, $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$. CMT-4 attack on EtE-HBSH is presented in Algorithm 3. We explain the attack step by step as described in the algorithm. We start by deriving the two keys K_E and K_H from K_S. We then obtain P_L and P_R from P, then we create EtE-HBSH .i.e., we apply Encode sub-routine of equation (12) on P_L to obtain P'_L .i.e., $P'_L \leftarrow (0^{\mu} \parallel P_L)$, where $\mu > 0$ and $(P_L \parallel P_R) \leftarrow P$. Let $P_M \leftarrow 0^n$ .i.e., we fix P_M to a constant value. Note that, in case of Adiantum, $n = 128$. Next, we fix the key to a constant value .i.e., $K_1 = K_2 = K$. We know that $K_S = K$ and K_E and K_H is obtained from the stream cipher using key K_S and instantiation with ϵ. Let $C_M \leftarrow E_{K_E}(P_M)$. We now compute the left part of the ciphertext .i.e., $C_L \leftarrow P'_L \oplus S_{K_S}(C_M)[1; |P'_L|]$. Now, we choose and fix a value for T_1, and compute $H_{K_H}(T_1, C_L)$. Then, it is trivial to find T_2 using the linear equation: $H_{K_H}(T_1, C_L) = H_{K_H}(T_2, C_L)$,

such that $T_1 \neq T_2$. Hence, $C_R \leftarrow H_{K_H}(T_1, C_L) \boxminus C_M$. We can now, simply append C_L and C_R to get the ciphertext, $C \leftarrow (C_L \parallel C_R)$. We can derive P_{R_1} and P_{R_2} as follows, $P_{R_1} = H_{K_H}(T_1, P'_L) \boxminus P_M$ and $P_{R_2} = H_{K_H}(T_2, P'_L) \boxminus P_M$. Hence, $P_1 \leftarrow (P'_L \parallel P_{R_1})$, $P_2 \leftarrow (P'_L \parallel P_{R_2})$. The adversary $\mathcal{A}$ has successfully generated two tuples (K, N_1, A_1, P_1) and (K, N_2, A_2, P_2) such that $\forall i\ (N_i \parallel A_i) \leftarrow T_i$, enforcing generation of same ciphertext C from EtE-HBSH using these two distinct tuples. Hence, $\tau = \{(K, N_1, A_1, P_1), (K, N_2, A_2, P_2)\}$, where $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$.

B. CMT-4 ATTACK ON EtE-HCTR2

Let $\mathcal{A}$ be a computationally bounded CMT-4 adversary that has access to the EtE-HCTR2 algorithm as described in section IV-B1. $\mathcal{A}$ must create two distinct tuples (K_1, N_1, A_1, P_1) and (K_2, N_2, A_2, P_2) such that the ciphertext C produced by the encryption sub-routine $\mathcal{E}$ of EtE-HCTR2 is the same. CMT-4 attack on EtE-HCTR2 is presented as a pseudocode in Algorithm 4. We now describe each step of the attack. First, we start by fixing the key .i.e., $K_1 = K_2 = K$. Now, we proceed by calculating $\bar{h}$ and L as follows: $\bar{h} \leftarrow E_K(\text{bin}(0))$ and $L \leftarrow E_K(\text{bin}(1))$ respectively. We know that $(U \parallel V) \leftarrow P$ (refer figure 1). We construct V' from V using equation (13) .i.e., $V' \leftarrow (0^\mu \parallel V)$, $\mu > 0$. Let $UU \leftarrow 0^n$, then we have $YY \leftarrow E_K(UU)$ and $S \leftarrow (UU \oplus YY \oplus L)$. We next calculate Z, $Z \leftarrow V' \oplus XCTR_K(S)[1; |V'|]$. We now fix T_1 and calculate $H_{\bar{h}}(T_1, V)$. Then we can obtain T_2 using equation $H_{\bar{h}}(T_1, V) = H_{\bar{h}}(T_2, V)$, such that $T_1 \neq T_2$. We can now calculate Y .i.e., $Y \leftarrow H_{\bar{h}}(T_1, V)$. Further, we can now find U_1 and U_2 as follows: $U_1 = H_{\bar{h}}(T_1, V') \oplus UU$ and $U_2 = H_{\bar{h}}(T_1, V') \oplus UU$. The ciphertext can be calculated as $C \leftarrow (Y \parallel Z)$. Further, P_1 and P_2 can be calculated as follows: $P_1 \leftarrow (U_1 \parallel V')$, $P_2 \leftarrow (U_2 \parallel V')$. Hence, $\tau = \{(K, N_1, A_1, P_1), (K, N_2, A_2, P_2)\}$, where $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$.

C. CMT-4 ATTACK ON EtE-DOUBLE-DECKER AND EtE-DOCKED-DOUBLE-DECKER

In this section, we present two new results of CMT-4 attack. In section V-C1, we present CMT-4 attack on EtE-double-decker and in section V-C2, we present the CMT-4 attack on EtE-docked-double-decker. Note that, we assume the deck function is based on Farfalle [20], for both double-decker and docked-double-decker. We present an overview of Farfalle with CMT-4 attack in appendix A.

1) CMT-4 ATTACK ON ETE-DOUBLE-DECKER

Let $\mathcal{A}$ be a computationally bounded CMT-4 adversary. The task of $\mathcal{A}$ is to create two distinct tuples (K_1, N_1, A_1, P_1) and (K_2, N_2, A_2, P_2), where $W_1 \leftarrow (N_1 \parallel A_1)$ and $W_2 \leftarrow (N_2 \parallel A_2)$, such that the output or the ciphertext C generated by the encryption sub-routine of EtE-double-decker for the two tuples are the same. Note that, in case of double-decker and docked-double-decker, each key K_1 and K_2 is actually a tuple of three keys .i.e., (K_1, K_{1_1}, K_{1_2}) and (K_2, K_{2_1}, K_{2_2}). We start

Algorithm 3 CMT-4 Attack on EtE-HBSH (EtE-Adiantum When $n = 128$)

$K_S \xleftarrow{\$} \mathcal{K}$
$K_E \parallel K_H = S_{K_S}(\epsilon)$ ▷ $|\epsilon| = 0$, *Derive* K_H *and* K_E *from* K_S
$P \xleftarrow{\$} \mathcal{M}$, where $|P| > n$
$P_L \parallel P_R \leftarrow P$, $|P_L| = n$
$P'_L \leftarrow (0^\mu \parallel P_L)$, $\mu > 0$ ▷ *Construct* P'_L *from* P_L
$P_M \leftarrow 0^n$
$C_M \leftarrow E_{K_E}(P_M)$
$C_L \leftarrow P'_L \oplus S_{K_S}(C_M)[1; |P'_L|]$
$\exists\ (T_1, T_2) \mid H_{K_H}(T_1, C_L) = H_{K_H}(T_2, C_L)$
$P_{R_1} = H_{K_H}(T_1, P'_L) \boxminus P_M$ and $P_{R_2} = H_{K_H}(T_2, P'_L) \boxminus P_M$ ▷ *Derive* P_{R_1} *and* P_{R_2}
$P_1 \leftarrow (P'_L \parallel P_{R_1})$, $P_2 \leftarrow (P'_L \parallel P_{R_2})$
$\tau = \{(K, N_1, A_1, P_1), (K, N_2, A_2, P_2)\}$, where $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$
Return τ

Algorithm 4 CMT-4 Attack on EtE-HCTR2

$K \xleftarrow{\$} \mathcal{K}$
$\bar{h} \leftarrow E_K(\text{bin}(0))$
$L \leftarrow E_K(\text{bin}(1))$
$P \xleftarrow{\$} \mathcal{M}$, where $|P| > n$
$(U \parallel V) \leftarrow P$, $|U| = n$
$V' \leftarrow 0^\mu \parallel V$, $\mu > 0$ ▷ *Construct* V' *from* V
$UU \leftarrow 0^n$
$YY \leftarrow E_K(UU)$
$S \leftarrow (UU \oplus YY \oplus L)$
$Z \leftarrow V' \oplus XCTR_K(S)[1; |V'|]$
$\exists\ (T_1, T_2) \mid H_{\bar{h}}(T_1, Z) = H_{\bar{h}}(T_2, Z)$
$M_1 = H_{\bar{h}}(T_1, V') \oplus UU$ and $M_2 = H_{\bar{h}}(T_1, V') \oplus UU$ ▷ *Derive* U_1 *and* U_2
$P_1 \leftarrow (U_1 \parallel V')$, $P_2 \leftarrow (U_2 \parallel V')$
$\tau = \{(K, N_1, A_1, P_1), (K, N_2, A_2, P_2)\}$, where $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$
Return τ

by fixing the keys of the two tuples, and hence, enumerating the tuple as $\bar{K}$ .i.e., $\bar{K} = (K_1, K_{1_1}, K_{1_2}) = (K_2, K_{2_1}, K_{2_2})$. Further, we know that, the plaintext is split into four parts .i.e., $(U_L \parallel U_R \parallel V_L \parallel V_R) \leftarrow P$, where $|U_L| = |V_R| = n$. W denotes the tweak and the ciphertext C is given by $C \leftarrow (X_L \parallel X_R \parallel Y_L \parallel Y_R)$ where $|X_L| = |Y_R| = n$. In our attack, we assume $U_{R_1} = U_{R_2} = 0^\mu$ where $\mu > 0$. Hence, we can construct U'_{R_1} and U'_{R_2} using equation (14), in the following way: $U'_{R_1} = (0^\mu \parallel U_{R_1})$ and $U'_{R_2} = (0^\mu \parallel U_{R_2})$, thereby making the size of of U'_{R_1} and U'_{R_2} as $2 \cdot \mu$ bits. This setup of U_{R_1} and U_{R_1} is one of the example, our CMT-4 attack will work on any arbitrary size. We advise the reader to augment the explanation of the attack with figure 3.

Step 1: We select F_{K_2} for starting the CMT-4 attack. F_{K_2} is defined over three input parameters and a key (refer

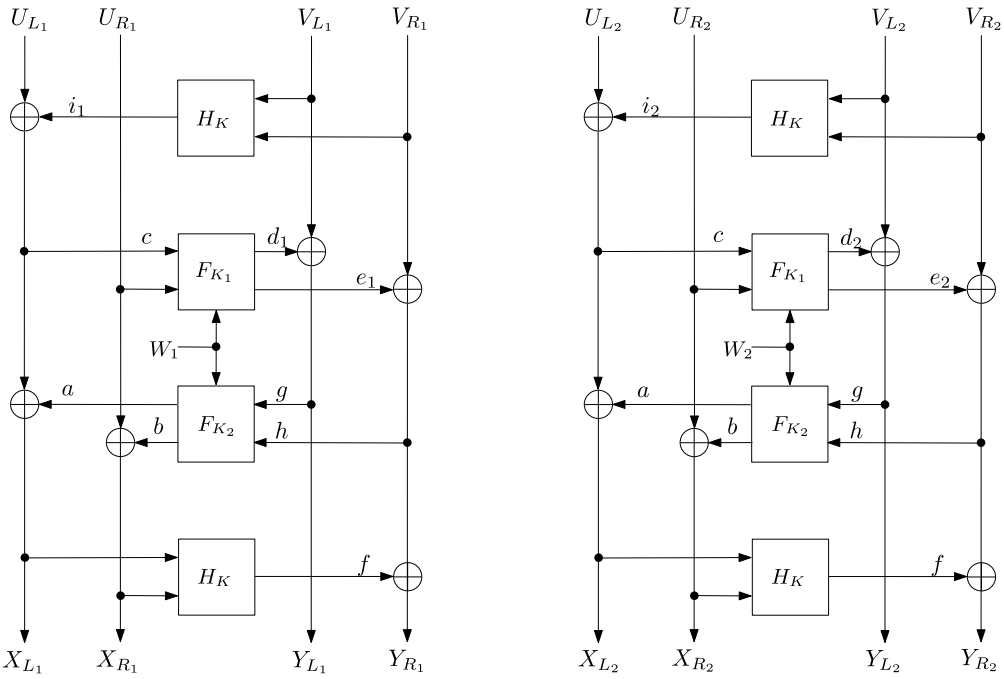


FIGURE 3. CMT-4 Attack on EtE-Double-Decker.

figure 3).We already fixed key .i.e.,

$$\bar{K} = (K_1, K_{1_1}, K_{1_2}) = (K_2, K_{2_1}, K_{2_2}).$$

Step 2: Fix two input parameters of F_{K_2} as g and h. Note that the third parameter is the tweak.
Now, we construct the inputs of Farfalle.[2] A padding function is required to ensure that inputs are multiple of $\bar{b}$ bits .i.e., $\text{pad10}^*(M) = M \parallel 10^{|M| \pmod{\bar{b}}-1}$ for $M \in \{0,1\}^*$. Let ENC be an encoding, then for two tweaks W_1 and W_2 we can construct input of Farfalle as follows:

$$\text{ENC}(W_1, g, h) = (\text{pad10}^*(W_1), \text{pad10}^*(g \parallel h)),$$
$$\text{ENC}(W_2, g, h) = (\text{pad10}^*(W_2), \text{pad10}^*(g \parallel h)).$$

Step 3: Using the collision attack on Farfalle (refer appendix A-C), we can say that, there exists two tweaks, such that output of F_{K_2} is deterministically produced .i.e., $\exists\ (W_1, W_2) \mid F_{K_2}(\text{ENC}(W_1, g, h)) = F_{K_2}(\text{ENC}(W_2, g, h)) = (a, b)$. This implies $X_{R_1} = X_{R_2} = b$, since $U_{R_1} = U_{R_2} = 0^\mu$ and $X_{R_1} = U_{R_1} \oplus b$ and $X_{R_2} = U_{R_2} \oplus b$ respectively.

Step 4: We now fix another value c, one of the parameters to the deck function F_{K_1} (please refer figure 3). Hence, $X_{L_1} = X_{L_2} = c \oplus a$. The input to H_K is deterministic .i.e., $c \oplus a$ (or X_L) and 0^μ (or X_R) are completely determined, there by fixing the output of H_K as f, we have $Y_{R_1} = Y_{R_2} = h \oplus f$.

Step 5: Since, g is fixed, hence $Y_{L_1} = Y_{L_2} = g$.

[2]Farfalle takes input as bits of string and a key, and processes the input in multiples of size $\bar{b}$-bits (refer appendix A-B).

Step 6: Since, $W_1 \neq W_2$, hence, $F_{K_1}(\text{ENC}(W_1, c, U_{R_1})) \neq F_{K_1}(\text{ENC}(W_2, c, U_{R_2}))$, or $F_{K_1}(\text{ENC}(W_1, c, U_{R_1})) = (d_1, e_1)$ and $F_{K_1}(\text{ENC}(W_2, c, U_{R_2})) = (d_2, e_2)$. Further, since, we already fixed g and h, hence $V_{R_1} = h \oplus e_1$ and $V_{R_2} = h \oplus e_2$. Also, $V_{L_1} = g \oplus d_1$ and $V_{L_2} = g \oplus d_2$.

Step 7: Since $V_{L_1} \neq V_{L_2}$, this makes the input to H_K as distinct .i.e., $H_K(g \oplus d_1, h \oplus e_1) = i_1$ and $H_K(g \oplus d_2, h \oplus e_2) = i_2$; making $U_{L_1} = i_1 \oplus c$ and $U_{L_2} = i_2 \oplus c$.

We can now construct the message and the ciphertext for the encryption sub-routine of EtE-double-decker:

$$\begin{aligned} P_1 &= (i_1 \oplus c) \parallel 0^\mu \parallel (g \oplus d_1) \parallel (h \oplus e_1), \quad \mu > 0 \\ C_1 &= (c \oplus a) \parallel b \parallel g \parallel (h \oplus f) \\ P_2 &= (i_2 \oplus c) \parallel 0^\mu \parallel (g \oplus d_2) \parallel (h \oplus e_2), \quad \mu > 0 \\ C_2 &= (c \oplus a) \parallel b \parallel g \parallel (h \oplus f). \end{aligned} \tag{16}$$

From equation (16), we see that $C_1 = C_2$, hence we can now construct two distinct tuples in the following way:

$$\begin{aligned} \tau = \{&(\bar{K}, N_1, A_1, P_1), (\bar{K}, N_2, A_2, P_2)\} \\ \text{where } &(N_1 \parallel A_1) \leftarrow W_1 \text{ and } (N_2 \parallel A_2) \leftarrow W_2, \\ &P_1 \leftarrow (i_1 \oplus c) \parallel 0^\mu \parallel (g \oplus d_1) \parallel (h \oplus e_1), \\ &P_2 \leftarrow (i_2 \oplus c) \parallel 0^\mu \parallel (g \oplus d_2) \parallel (h \oplus e_2). \end{aligned} \tag{17}$$

From equation (17), we have constructed two tuples, such that $P_1 \neq P_2$ and $W_1 \neq W_2$ but $C_1 = C_2$. This completes our CMT-4 attack on EtE-double-decker.

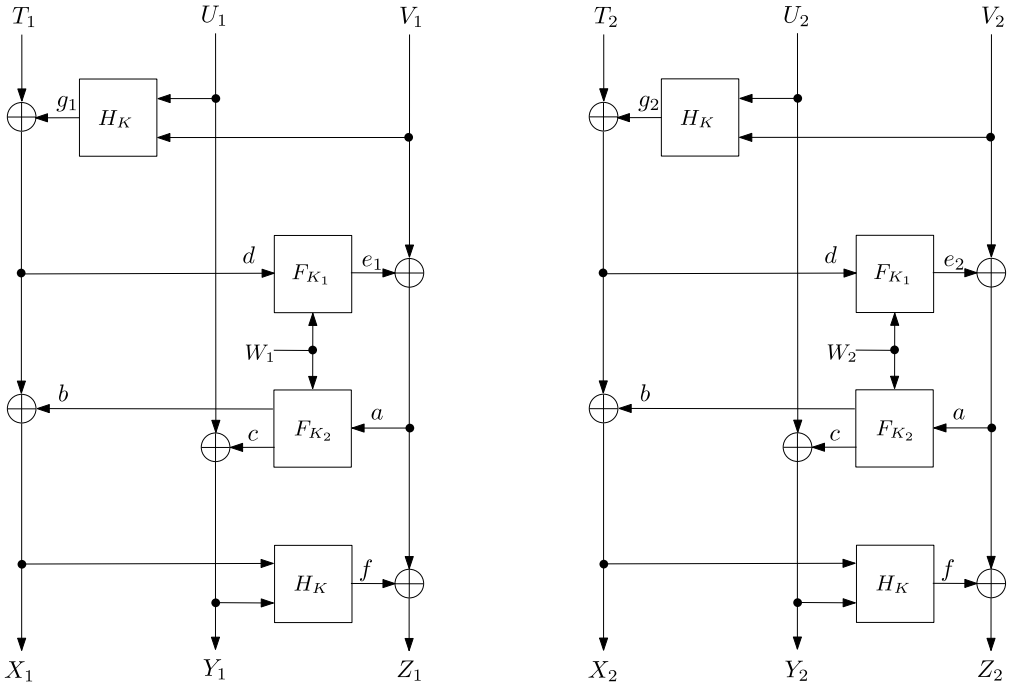


FIGURE 4. CMT-4 Attack on EtE-Docked-Double-Decker.

2) CMT-4 ATTACK ON ETE-DOCKED-DOUBLE-DECKER

The task of $\mathcal{A}$ is to create two distinct tuples (K_1, N_1, A_1, P_1) and (K_2, N_2, A_2, P_2), where $W_1 \leftarrow (N_1 \parallel A_1)$ and $W_2 \leftarrow (N_2 \parallel A_2)$, such that the output or the ciphertext C generated by the encryption sub-routine of EtE-docked-double-decker are the same. As mentioned earlier (section V-C1), each key K_1 and K_2 is actually a tuple of three keys .i.e., (K_1, K_{1_1}, K_{1_2}) and (K_2, K_{2_1}, K_{2_2}). We fix the keys by enumerating them as a tuple $\bar{K}$ .i.e., $\bar{K} = (K, K_1, K_2)$. The plaintext is divided into three parts .i.e., $(T \parallel U \parallel V) \leftarrow P$, where $|T| = |V| = n$. W denotes the tweak and the ciphertext C is given by $C \leftarrow (X \parallel Y \parallel Z)$ where $|X| = |Z| = n$. As in the case of EtE-double-decker (section V-C1), we proceed by making $U_1 = U_2 = 0^\mu$, where $\mu > 0$, and derieve U_1' and U_2' using equation (15) .i.e., $U_1' = (0^\mu \parallel U_1)$ and $U_2' = (0^\mu \parallel U_2)$, thereby $|U_1'| = |U_2'| = 2 \bullet \mu$. Please refer figure 4 for the explanation of the attack.

Step 1: We select F_{K_2} for the CMT-4 attack. The key is already fixed .i.e.,

$$\bar{K} = (K_1, K_{1_1}, K_{1_2}) = (K_2, K_{2_1}, K_{2_2}).$$

Step 2: We fix a .i.e., one of the parameters of F_{K_2}. Let ENC be an encoding, then for two tweaks W_1 and W_2 we can construct input of Farfalle as follows:

$$\text{ENC}(W_1, a) = (\text{pad10}^*(W_1), \text{pad10}^*(a)),$$
$$\text{ENC}(W_2, a) = (\text{pad10}^*(W_2), \text{pad10}^*(a)).$$

Step 3: Using the collision attack of Farfalle (refer appendix A-C), we can say that, there exists two tweaks, W_1 and W_2 such that the output from F_{K_2} is completely determined .i.e., $\exists\ (W_1, W_2) \mid F_{K_2}(\text{ENC}(W_1, a)) = F_{K_2}(\text{ENC}(W_2, a)) = (b, c)$. Hence, $Y = 0^\mu \oplus c = c$.

Step 4: We now fix d, then $X = b \oplus d$. Further, we have $H_K(b \oplus d, c) = f$, this implies the value of Z is also completely determined .i.e., $Z = f \oplus a$.

Step 5: Since, $W_1 \neq W_2$, hence, $F_{K_1}(\text{ENC}(W_1, d)) \neq F_{K_1}(\text{ENC}(W_2, d))$. Let $F_{K_1}(\text{ENC}(W_1, d)) = e_1$ and $F_{K_1}(\text{ENC}(W_2, d)) = e_2$. Proceeding this way, we have $V_1 = e_1 \oplus a$ and $V_2 = e_2 \oplus a$.

Step 6: Since, $g_1 = H_K(0^\mu, e_1 \oplus a)$ and $g_2 = H_K(0^\mu, e_2 \oplus a)$, hence $T_1 = d \oplus g_1$ and $T_2 = d \oplus g_2$.

We can now construct the message and the ciphertext for the encryption sub-routine of EtE-docked-double-decker:

$$\begin{aligned} P_1 &= (d \oplus g_1) \parallel 0^\mu \parallel (e_1 \oplus a), \quad \mu > 0 \\ C_1 &= (b \oplus d) \parallel c \parallel (f \oplus a). \\ P_2 &= (d \oplus g_2) \parallel 0^\mu \parallel (e_2 \oplus a), \quad \mu > 0 \\ C_2 &= (b \oplus d) \parallel c \parallel (f \oplus a). \end{aligned} \tag{18}$$

From equation (18), we see that $C_1 = C_2$, hence we can now construct two distinct tuples in the following way:

$$\begin{aligned} \tau &= \{(\bar{K}, N_1, A_1, P_1), (\bar{K}, N_2, A_2, P_2)\} \\ \text{where } (N_1 \parallel A_1) &\leftarrow W_1 \text{ and } (N_2 \parallel A_2) \leftarrow W_2, \\ P_1 &\leftarrow (d \oplus g_1) \parallel 0^\mu \parallel (e_1 \oplus a) \\ P_2 &\leftarrow (d \oplus g_2) \parallel 0^\mu \parallel (e_2 \oplus a). \end{aligned} \tag{19}$$

From equation (19), we have constructed two tuples such that $P_1 \neq P_2$ and $W_1 \neq W_2$ but $C_1 = C_2$. This completes our CMT-4 attack on EtE-docked-double-decker.

In table 3, we present the summary of CMT-4 attack on encode-then-encipher versions of HBSH, HCTR2,

TABLE 3. Summary of CMT-4 attack. EtE-HBSH, EtE-HCTR2, EtE-Double-Decker and EtE-Docked-Double-Decker. Two new results of CMT-4 attack are shown in red color. In case of EtE-HBSH and EtE-HCTR2, we present detailed analysis and algorithm of CMT-4 attack.

AEAD under **EtE** Paradigm	**Message Encoding** ($\mu > 0$)	**Tweakable Wide Block Cipher Scheme (Message Enciphering)**	**CMT-4 Attack**
EtE-HBSH	$((0^{\mu} \parallel P_L) \parallel P_R) \leftarrow P$	HBSH	Section V-A [14], [43]
			Time Complexity $= O(1)$
EtE-HCTR2	$(M \parallel (0^{\mu} \parallel N)) \leftarrow P$	HCTR2	Section V-B [14], [43]
			Time Complexity $= O(1)$
EtE-Double-Decker	$(U_L \parallel (0^{\mu} \parallel U_R) \parallel V_L \parallel V_R) \leftarrow P$	Double-decker	Section V-C1
			Time Complexity $= O(1)$
EtE-Docked-Double-Decker	$(T \parallel (0^{\mu} \parallel U) \parallel V) \leftarrow P$	Docked-double-decker	Section V-C2
			Time Complexity $= O(1)$

double-decker and docked-double-decker. New results of CMT-4 attack is shown in red color, while in case of EtE-HBSH and EtE-HCTR2, we provide detailed analysis and algorithm for CMT-4 attack.

VI. HBtSH, HtS, tS-DOUBLE-DECKER AND tS-DOCKED-DOUBLE-DECKER

In this section, we present four new designs of tweakable wide block cipher schemes that utilizes tweakable stream cipher (tS). In section VI-A, we introduce HBtSH, which is based on HBSH. In section VI-B, we present HtS based on HCTR2. We present tS-double-decker based on double-decker, and tS-docked-double-decker based on docked-double-decker in section VI-C. In this paper, we use eXtendable-Output Function (or XOF in short) [44] for instantiating the tweakable stream cipher.

A. HBtSH

We present a new tweakable wide block cipher scheme called HBtSH (**H**ash **B**lock cipher **t**weakable **S**tream cipher **H**ash). The architecture of the scheme is shown in figure 5, and we present the pseudocode of the scheme in algorithm 5. HBtSH based on a tweakable stream cipher (refer definition 3), and uses the structural design of HBSH (refer section IV-A for HBSH), but replaces its one of the underlying primitive .i.e., instead of using a stream cipher, HBtSH uses a tweakable stream cipher. We now describe the construction of HBtSH.

Let $\mathcal{E}$ and $\mathcal{D}$ be the encryption sub-routine and the decryption sub-routine of HBtSH. $\mathcal{E}$ takes in three inputs, a key K, a tweak T and a plaintext P and produces a ciphertext C as the output. $\mathcal{D}$ takes input as a key K, a tweak T and a ciphertext C and outputs the corresponding plaintext P. The tweakable stream cipher is invoked two times, once for generating the ciphertext C_L, where $(C_L \parallel C_R) \leftarrow C$, and once for the key derivation function.

- **Generating Ciphertext:** S is used to generate the left part of the ciphertext .i.e., C_L. S takes in three inputs: a key K, a fixed size nonce C_M and a tweak T, and a maximum output length denoted by l_S, such that $|P_L| \leq l_S$. Hence, the invocation of S can be represented by following expression:

$$S(K, C_M, T)[1; |P_L|]. \tag{20}$$

The key stream generated by equation (20) is *xored* with P_L for generating C_L. When a XOF-based S is used, a reversible encoding denoted by ENC is utilized for combining C_M and T .i.e., ENC(C_M, T). For a nonce with size $< \eta$, C_M in this case, ENC can be written as $(\text{pad}_{\eta}(C_M) \parallel T)$, where padding function is defined in equation (1).

- **Key Derivation Function:** The key K_E (used for the block cipher) and the key K_H (used for the hash function) is derived using the tweakable stream cipher with input as the key K and empty nonce and empty tweak, with a maximum output length given by l_S such that $|K_E| + |K_H| \leq l_S$. The invocation of S can be represented by

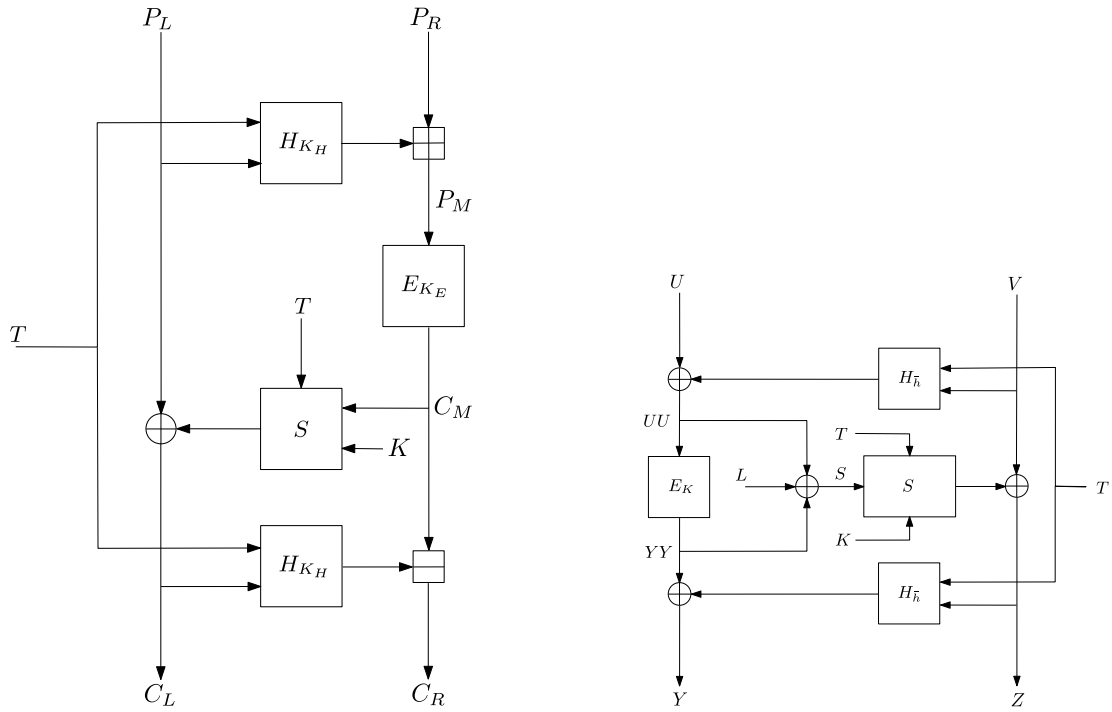
Algorithm 5 HBtSH

$\mathcal{E}(K, T, P)$

1: $(K_E \parallel K_H) \leftarrow S(K, \epsilon, \epsilon)[1; |K_H| + |K_E|]$ ▷ *Derive* K_H *and* K_E *from* K
2: $(P_R \parallel P_L) \leftarrow P, |P_L| = n$
3: $P_M \leftarrow P_R \boxplus H_{K_H}(T, P_L)$
4: $C_M \leftarrow E_{K_E}(P_M)$
5: $C_L \leftarrow P_L \oplus S(K_S, C_M, T)[1; |P_L|]$
6: $C_R \leftarrow C_M \boxminus H_{K_H}(T, C_L)$
7: $C \leftarrow (C_L \parallel C_R)$
8: **Return** C

$\mathcal{D}(K, T, C)$

1: $(K_E \parallel K_H) \leftarrow S(K, \epsilon, \epsilon)[1; |K_H| + |K_E|]$ ▷ *Derive* K_H *and* K_E *from* K
2: $(C_R \parallel C_L) \leftarrow C, |C_L| = n$
3: $C_M \leftarrow C_R \boxplus H_{K_H}(T, C_L)$
4: $P_L \leftarrow C_L \oplus S(K_S, C_M, T)[1; |C_L|]$
5: $P_M \leftarrow E^{-1}_{K_E}(C_M)$
6: $P_R \leftarrow P_M \boxminus H_{K_H}(T, P_L)$
7: $P \leftarrow (P_L \parallel P_R)$
8: **Return** P



(a) HBtSH (b) HtS

FIGURE 5. HBtSH and HtS: Tweakable wide block cipher scheme based on tweakable stream cipher. HBtSH is based on HBSH and HtS is based on HCTR2. The construction of HtS is similar to HBtSH and hence, we omit the CMT-4 security proof of HtS.

following expression:

$$S(K, \epsilon, \epsilon)[1; |K_E| + |K_H|]. \tag{21}$$

The key stream generated by equation (21) is used as the two keys K_E and K_H. We enforce the condition that when using XOF, equation (20) and equation (21) must be unrelated. This will lead to an unrelated output stream. HBtSH from algorithm 5 can be used to describe the Adiantum specification. We present CMT-4 security proof of HBtSH in section VIII-A under encode-then-encipher paradigm.

B. HtS

HtS is based on HCTR2. The tweakable stream cipher S is used instead of the stream encryption $XCTR_K$ (refer figure 1), where K is the key. The invocation of S is given by the following expression:

$$S(K, S, T)[1; |V|]. \tag{22}$$

The key stream generated by equation (22) is *xored* with V to obtain Z. We present the construction of HtS in figure 5b and the pseudocode of HtS in presented the algorithm 6.

At this stage, we want to point out that the construction of HtS eliminates the use of stream encryption .i.e., $XCTR_K(S) = E_K(S \oplus \text{bin}(1)) \parallel E_K(S \oplus \text{bin}(2)) \parallel \ldots$, which makes the construction similar to HBtSH, and hence we omit the CMT-4 security proof of HtS under encode-then-encipher paradigm.

Algorithm 6 HtS

$\mathcal{E}(K, T, P)$

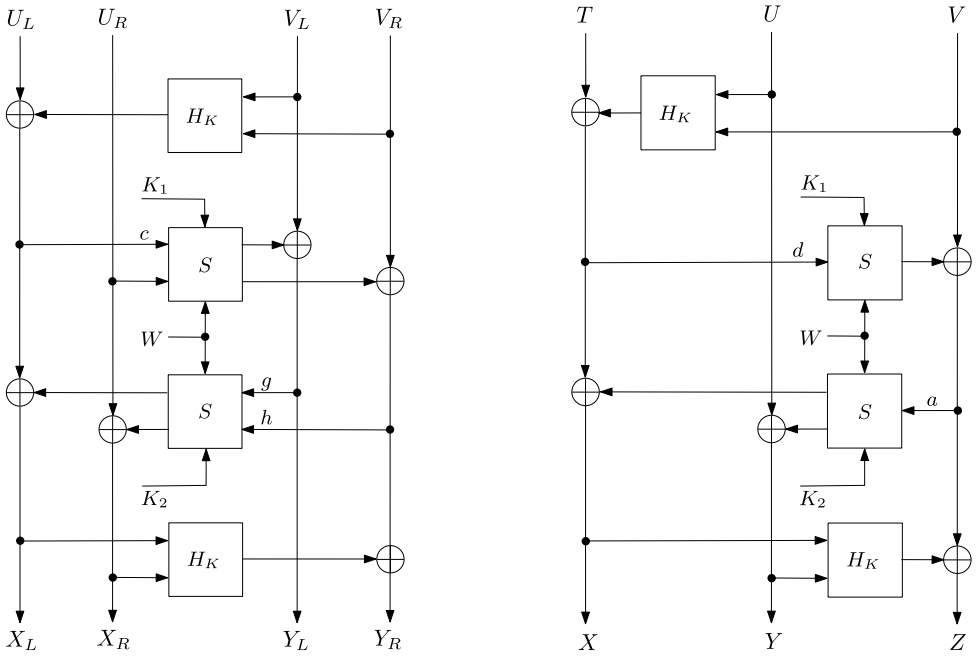
$\bar{h} \leftarrow E_K(\text{bin}(0))$
$L \leftarrow E_K(\text{bin}(1))$
$(U \parallel V) \leftarrow P$, where $|U| = n$
$UU \leftarrow U \oplus H_{\bar{h}}(T, V)$
$YY \leftarrow E_K(UU)$
$S \leftarrow (UU \oplus YY \oplus L)$
$Z \leftarrow V \oplus S(K, S, T)[1; |V|]$
$Y \leftarrow YY \oplus H_{\bar{h}}(T, Z)$
$C \leftarrow (Y \parallel Z)$
Return C

$\mathcal{D}(K, T, C)$

$\bar{h} \leftarrow E_K(\text{bin}(0))$
$L \leftarrow E_K(\text{bin}(1))$
$(Y \parallel Z) \leftarrow C$, where $|Y| = n$
$YY \leftarrow Y \oplus H_{\bar{h}}(T, Z)$
$UU \leftarrow E_K^{-1}(YY)$
$S \leftarrow (UU \oplus YY \oplus L)$
$V \leftarrow Z \oplus S(K, S, T)[1; |Z|]$
$U \leftarrow UU \oplus H_{\bar{h}}(T, V)$
$P \leftarrow (U \parallel V)$
Return P

C. tS-DOUBLE-DECKER AND tS-DOCKED-DOUBLE-DECKER

In this section, we present two new constructions of tweakable wide block cipher scheme, tS-double-decker based



(a) tS-Double-Decker (b) tS-Docked-Double-Decker

FIGURE 6. tS-Double-decker and tS-docked-double-decker: Tweakable wide block cipher scheme based on a tweakable stream cipher.

on double-decker, and tS-docked-double-decker based on the docked-double-decker. Similar to HBtSH, both of these constructions are based on a tweakable stream cipher S. We describe each of the schemes in section VI-C1 and section VI-C2. We start with the description of the key derivation function for tS-double-decker and tS-docked-double-decker.

We use the key K_2 for the second instance of the tweakable stream cipher and also use the key K_2 with the empty instantiation of tweakable stream cipher for generating the other two keys .i.e., K and K_1:

$$S(K_2, \epsilon, \epsilon)[1; |K| + |K_1|]. \tag{23}$$

We want to point out that, it is necessary to generate the two keys K and K_1 using the key derivation function of equation (23) with K_2 as one of the input, due to the fact that, if the keys are unrelated then it will become trivial to find two keys (K', K'') producing the same output from the polynomial hash function.

1) TS-DOUBLE-DECKER

In section IV-C1, we saw that the construction of double-decker uses deck function two times .i.e., F_{K_1} with key K_1 and F_{K_2} with key K_2. We replace these two instances of the deck function with a tweakable stream cipher. Figure 6 shows the construction of ts-double-decker. The two instances of S for ts-double-decker is show below:

$$\begin{aligned} &S(K_1, c, \text{ENC}(U_R, W))[1; |V_L| + |V_R|] \\ &S(K_2, h, \text{ENC}(g, W))[1; |V_L| + |V_R|]. \end{aligned} \tag{24}$$

where $\text{ENC}(X, Y) = (\text{bin}_\eta(|X|) \parallel X \parallel Y)$ and $|X| < 2^\eta$. The key stream generated by the first instance of S in equation (24) is *xored* with V_L and V_R. The key stream generated by the second instance S is *xored* with U_L and U_R.

2) TS-DOCKED-DOUBLE-DECKER

Similar to double-decker, docked-double-decker uses the deck function two times (refer section IV-C2). We replace these two instances of the deck function by the tweakable stream cipher, hence resulting into ts-docked-double-decker (refer figure 6b). The two instances of S for ts-docked-double-decker is show below:

$$\begin{aligned} &S(K_1, d, W))[1; |V|] \\ &S(K_2, a, W))[1; |T| + |U|]. \end{aligned} \tag{25}$$

The key stream generated by S in equation (25) during the first instance is *xored* with V, while, in case of the second instance S, the key stream generated is *xored* with T and U.

In table 4, we present summary of all the four proposed tweakable wide block cipher schemes.

VII. DESIGN RATIONALE

In this section, we describe the design rationale behind the four proposed tweakable wide block cipher schemes.

- **Tweakable Stream Cipher (tS):** In section V-A and section V-B, we saw that the CMT-4 attack on EtE-HBSH and EtE-HCTR2 was possible since the stream cipher in EtE-HBSH and the stream encryption in EtE-HCTR2 did not take the tweak as the input. Further, for ts-double-decker and ts-docked-double-decker, the two deck functions based on Farfalle [20]

TABLE 4. Summary of proposed tweakable wide block cipher schemes.

Proposed Scheme	CMT-4 Security Proof under EtE paradigm
HBtSH (Section VI-A)	Theorem 1.
HtS (Section VI-B)	Omitted for brevity (design is similar to HBtSH).
tS-double-decker (Section VI-C)	Theorem 3.
tS-docked-double-decker (Section VI-C)	Theorem 3.

construction was unable to counter CMT-4 attack. The tweakable stream cipher ensures partial collision resistance against a CMT-4 adversary under encode-then-encipher paradigm. In this paper, we use XOF [44] as an instantiation to the S.

XOF-based Tweakable Stream Cipher:

1) Security Claim: Let a sponge function be based on an ideal permutation and total bit length of data be at most σ, then the indifferentiability of the sponge function from a random oracle is given by the expression: $\leq \frac{\sigma^2}{2^{c+1}}$ [45], where c is the capacity of the sponge function. Now, for performing μ-bit partial collision on random oracle, its probability bound is given by the expression: $\leq \frac{q^2}{2^{\mu+1}}$, where q is the number of queries to the random oracle and μ is the number of bits for partial collision. Hence, the probability bound of the sponge function for performing μ-bit partial collision, where the underlying permutation is ideal, is given by the expression: $\leq \frac{\sigma^2}{2^{c+1}} + \frac{q^2}{2^{\mu+1}}$.
   For example, in case of SHAKE [46], the capacity c of SHAKE128 is 256, thereby making the probability bound for performing μ-bit partial collision on SHAKE128 to $\leq \frac{\sigma^2}{2^{257}} + \frac{q^2}{2^{\mu+1}}$, assuming underlying permutation of SHAKE128 is ideal, while in case of SHAKE256, $c = 512$, hence, the probability bound for performing μ-bit partial collision on SHAKE256 is $\leq \frac{\sigma^2}{2^{513}} + \frac{q^2}{2^{\mu+1}}$, where underlying permutation of SHAKE256 is ideal.
2) Implementation: TurboSHAKE [47] is a family of eXtendable Output Functions (XOF) with reduced round, that can be used for the applications requiring fast computation or hardware that have low computing power. Hence, TurboSHAKE may be suitable for application curated for IoT devices like disk encryption scenarios. Further, XOF is an active area of research and future advancements might result into more efficient designs.

- **Recycle Proofs:** The construction of HBtSH, HtS, tS-double-decker and tS-docked-double-decker ensures a minimal change in the original design .i.e., HBSH, HCTR2, double-decker and docked-double-decker respectively. The proposed designs are structurally similar to their original counter parts, hence verification of the proposed designs are intuitive. The security proofs of the original design can be recycled, giving us the leverage to provide only the CMT-4 security proof for the respective proposed designs under encode-then-encipher paradigm. The will further lead to easy verification by the research community.
- **Facilitate Existing Implementations:** Due to the structural similarity of all the four proposed designs, it fairly easy task to replace the existing implementations with the new proposed ones. This will further lead to easy benchmarking under standard metrics. Moreover, the property of structural similarity in the proposed designs will facilitate the replacement in the existing deployments.

VIII. SECURITY PROOFS

A. SECURITY PROOF OF HBtSH

We begin with some definitions. Let $\mathcal{K} \xleftarrow{\$} \{0,1\}^k$ denote the key space with key size equal to k, $\mathcal{M} \in \{0,1\}^*$ is the message space and $\mathcal{T} \in \{0,1\}^*$ denotes the space of the tweak. HBtSH (algorithm 5) takes in three inputs: a key $K \in \mathcal{K}$, a tweak $T \in \mathcal{T}$ and a message (or a plaintext) $P \in \mathcal{M}$, and outputs a ciphertext $C \in \mathcal{C}$, where $\mathcal{C}$ is the ciphertext space and $|M| = |C|$. In case of CMT-4 security of an authenticated encryption scheme, the task of the adversary $\mathcal{A}$, is to construct two distinct tuples (K_1, N_1, A_1, P_1) and (K_2, N_2, A_2, P_2) for the encryption sub-routine of EtE-HBtSH, such that $\tau = \{(K_1, N_1, A_1, P_1), (K_2, N_2, A_2, P_2)\}$, where $(K_1, N_1, A_1, P_1) \neq (K_2, N_2, A_2, P_2)$ and $\mathcal{E}(K_1, N_1, A_1, P_1) = \mathcal{E}(K_2, N_2, A_2, P_2)$. In case of EtE-HBtSH, $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$. CMT-4 security is defined in definition 9 and EtE-HBtSH is defined in definition 10. We now present the proof of CMT4 security of EtE-HBtSH in **theorem** 1.

Theorem 1: Let $\mathcal{A}$ be a CMT-4 adversary defined in algorithm 2 that has access to EtE-HBtSH with μ-bit zero prepending, and creates two distinct tuples, $\tau = \{(K_1, N_1, A_1, P_1), (K_2, N_2, A_2, P_2)\}$ s.t $C_1 \leftarrow \mathcal{E}(K_1, N_1, A_1, P_1)$, $C_2 \leftarrow \mathcal{E}(K_2, N_2, A_2, P_2)$, and $T_i = (N_i \parallel A_i)$, $i \in \{1, 2\}$, where $(K_1, N_1, A_1, P_1) \neq (K_2, N_2, A_2, P_2)$ and $C_1 = C_2$, then, we can construct a partial collision finding adversary $\mathcal{B}_\mathcal{A}$ on tS given by:

$$\mathrm{Adv}^{\text{CMT-4}}_{\text{EtE-HBtSH}}(\mathcal{A}) \leq \mathrm{Adv}^{\mu\text{-PartialColl}}_{\text{tS}}(\mathcal{B}_\mathcal{A}). \tag{26}$$

Proof: We start by presenting a algorithm (refer Algorithm 7) that transforms the query executed by the adversary $\mathcal{A}$ as a halting problem. The adversary $\mathcal{B}_\mathcal{A}$ takes input from the adversary $\mathcal{A}$ and executes the partial collision attack.

Note that in case of HBtSH, we combine the nonce and the associated data to form the tweak .i.e., $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$. We now proceed to calculate the CMT-4 bound of EtE-HBtSH using following two cases:

- **Case 1:** $(K_1, T_1) = (K_2, T_2)$
  For the two tuples generated by $\mathcal{A}$ .i.e., $\{(K_1, N_1, A_1, P_1),$

Algorithm 7 $\mathcal{B}_{\mathcal{A}}$

$\tau \leftarrow \phi$
Run $\mathcal{A}$**:**
  Generate $\{(K_1, N_1, A_1, P_1), (K_2, N_2, A_2, P_2)\} \xleftarrow{\$} \mathcal{A}$
  ▷ $\forall i\ T_i \leftarrow (N_i \parallel A_i)$
  Execute: $C_1 \leftarrow \mathcal{E}(K_1, N_1, A_1, P_1)$, $C_2 \leftarrow \mathcal{E}(K_2, N_2, A_2, P_2)$
Halt: $(C_1 \stackrel{?}{=} C_2) \wedge (K_1, N_1, A_1, P_1) \stackrel{?}{=} (K_2, N_2, A_2, P_2)$
  if $(C_1 = C_2) \wedge (K_1, N_1, A_1, P_1) \neq (K_2, N_2, A_2, P_2)$
    then $(P_{L_1} \parallel P_{R_1}) \leftarrow P_1$, $(P_{L_2} \parallel P_{R_2}) \leftarrow P_2$
  ▷ Generate inputs of tS
    $P'_{L_1} \leftarrow (0^{\mu} \parallel P_{L_1})$ and $P'_{L_2} \leftarrow (0^{\mu} \parallel P_{L_2})$
    $(K_{1_E}, K_{1_H}) \leftarrow S(K_1, \epsilon, \epsilon)[1; |K_{1_E}| + |K_{1_H}|]$
    $(K_{2_E}, K_{2_H}) \leftarrow S(K_2, \epsilon, \epsilon)[1; |K_{2_E}| + |K_{2_H}|]$
    $P_{M_1} \leftarrow P_{R_1} \boxplus H_{K_{1_H}}(T_1, P'_{L_1})$
    $P_{M_2} \leftarrow P_{R_2} \boxplus H_{K_{2_H}}(T_2, P'_{L_2})$
    $C_{M_1} \leftarrow E_{K_{1_E}}(P_{M_1})$, $C_{M_2} \leftarrow E_{K_{2_E}}(P_{M_2})$
    $\tau = \{(K_1, C_{M_1}, T_1), (K_2, C_{M_2}, T_2)\}$
return τ

$(K_2, N_2, A_2, P_2)\} \xleftarrow{\$} \mathcal{A}$, we have $(K_1, T_1) = (K_2, T_2)$, where $(N_1 \parallel A_1) \leftarrow T_1$ and $(N_2 \parallel A_2) \leftarrow T_2$. Now from the construction of HBtSH in section VI-A and figure 5, we know that the tweakable stream cipher S is defined in the following way: $S(K_1, C_{M_1}, T_1)$ and $S(K_2, C_{M_2}, T_2)$, where C_{M_1} and C_{M_2} are the two nonces. The key stream generated by the tweakable stream cipher S is used to generate the left part of the ciphertext .i.e., $C_{L_1} \leftarrow P'_{L_1} \oplus S(K_1, C_{M_1}, T_1)[1; |P'_{L_1}|]$ and $C_{L_2} \leftarrow P'_{L_2} \oplus S(K_2, C_{M_2}, T_2)[1; |P'_{L_2}|]$, where $P'_{L_1} \leftarrow (0^{\mu} \parallel P_{L_1})$, $P'_{L_2} \leftarrow (0^{\mu} \parallel P_{L_2})$, $(P_{L_1} \parallel P_{R_1}) \leftarrow P_1$ and $(P_{L_2} \parallel P_{R_2}) \leftarrow P_2$. Now, since S is a permutation, it is not possible to generate two different outputs .i.e., C_{L_1} and C_{L_2}, given the same inputs, since $(K_1, T_1) = (K_2, T_2)$. This leads to the fact that, in this case, it is not possible to generate two different outputs from the tweakable stream cipher, hence, in equation (26), we have $\text{Adv}^{\text{CMT-4}}_{\text{EtE-HBtSH}}(\mathcal{A}) = 0$.

- **Case 2:** $(K_1, T_1) \neq (K_2, T_2)$
  In this case, the tweakable stream cipher takes two distinct tuples .i.e., $S(K_1, C_{M_1}, T_1)$ and $S(K_2, C_{M_2}, T_2)$. Under encode-then-encipher paradigm, we know from definition 10 and equation (12), that in case of EtE-HBtSH, we have $P'_{L_1} \leftarrow (0^{\mu} \parallel P_{L_1})$ and $P'_{L_2} \leftarrow (0^{\mu} \parallel P_{L_2})$, where $(P_{L_1} \parallel P_{R_1}) \leftarrow P_1$, $(P_{L_2} \parallel P_{R_2}) \leftarrow P_2$ and $\mu > 0$. Hence, to generate the same ciphertext part $C_L = C_{L_1} = C_{L_2}$, the tweakable stream cipher must produce μ-bit partial collision in its output. This leads to the fact that the game in Algorithm 2 returns 1, making the game in Algorithm 1 to return 1 as well. Hence, using definition 9 and definition 8, we have $Pr[\mathbf{G}^{\text{CMT-4}}_{\text{EtE-HBtSH}}(\mathcal{A}) \mapsto 1] = Pr[\mathbf{G}^{\mu\text{-PartialColl}}_{\text{tS}}(\mathcal{B}_{\mathcal{A}}) \mapsto 1]$ .i.e., $\text{Adv}^{\text{CMT-4}}_{\text{EtE-HBtSH}}(\mathcal{A}) = \text{Adv}^{\mu\text{-PartialColl}}_{\text{tS}}(\mathcal{B}_{\mathcal{A}})$.

From case 1 and case 2, we conclude that $\text{Adv}^{\text{CMT-4}}_{\text{EtE-HBtSH}}(\mathcal{A}) \leq \text{Adv}^{\mu\text{-PartialColl}}_{\text{tS}}(\mathcal{B}_{\mathcal{A}})$, which proofs our claim in equation (26). Hence, we conclude our CMT-4 security proof of EtE-HBtSH. ■

We now present the security proof of HBtSH in theorem 2 below.

Theorem 2: Let HBtSH be the scheme defined in section VI-A, where H is ϵ-almost-Δ-universal for inputs, then we define:

$$\begin{aligned}&\text{Adv}^{\pm\widetilde{PRP}}_{\text{HBtSH}}(q, l_T, l_M, t)\\ &\leq (\epsilon + \frac{2}{2^n}) \cdot \binom{q}{2}\\ &\quad + \text{Adv}^{\text{tS}}_{S}(q+1, l_T, l_M + |K_E| + |K_H| - q \cdot n, t')\\ &\quad + \text{Adv}^{\pm\text{PRP}}_{E}(q, t'),\end{aligned}$$

where K_E is key for E, K_H is key for H, n is the output size of E and $t' = t + O(l_T + l_M)$.

Proof: The design of HBtSH is similar to HBSH with the exception of a tweakable stream cipher (definition 3), instead of a stream cipher, and therefore we leave the proof for brevity. We encourage the reader to refer theorem 1 in [2] for the proof. ■

B. SECURITY PROOF OF tS-DOUBLE-DECKER AND tS-DOCKED-DOUBLE-DECKER

We present the CMT-4 proof of EtE-tS-double-decker and EtE-tS-docked-double-decker in theorem 3. In case of EtE-ts-double-decker, μ bit zero prepending is done on U_R, while for EtE-ts-docked-double-decker, we use U for μ bit zero prepending. Further, please note that, we use key K_2 for the second instantiation of the tweakable stream cipher and generate the other two keys K and K_1 using key K_2 (refer equation (23)).

Theorem 3: Let Π be EtE-tS-Double-Decker or EtE-tS-Docked-Double-Decker created using tS-Double-Decker and tS-Docked-Double-Decker respectively, as defined in section VI-C1 and section VI-C2. Let $\mathcal{A}$ be a CMT-4 adversary defined in algorithm 2 that has access to Π with μ-bit zero prepending to U_R or U respectively, and creates two distinct tuples $\tau = \{(K_1, N_1, A_1, M_1), (K_2, N_2, A_2, M_2)\}$ s.t $C_1 \leftarrow \mathcal{E}(K_1, N_1, A_1, M_1)$, $C_2 \leftarrow \mathcal{E}(K_2, N_2, A_2, M_2)$, and $W_i = (N_i \parallel A_i)$, $i \in \{1, 2\}$, where $(K_1, N_1, A_1, M_1) \neq (K_2, N_2, A_2, M_2)$ and $C_1 = C_2$, then, we can construct a partial collision finding adversary $\mathcal{B}_{\mathcal{A}}$ on tS given by:

$$\text{Adv}^{\text{CMT-4}}_{\Pi}(\mathcal{A}) \leq \text{Adv}^{\mu\text{-PartialColl}}_{\text{tS}}(\mathcal{B}_{\mathcal{A}}). \tag{27}$$

Proof: The proof is similar to theorem 1, hence we omit it for brevity. ■

Theorem 4: Let tS-Double-Decker be the scheme defined in section VI-C1, where H is blinded keyed hash, then we define:

$$\begin{aligned}&\text{Adv}^{\pm\widetilde{PRP}}_{\text{tS-Double-Decker}}(q, l_W, l_M, t)\\ &\leq \text{Adv}^{\text{tS}}_{S}(q, l_W + l_{U_R}, l_M - l_{U_R} - q \cdot n, t')\\ &\quad + \text{Adv}^{\text{tS}}_{S}(q+1, l_W + l_{V_L}, l_M\\ &\quad + |K| + |K_1| - l_{V_L} - q \cdot n, t')\end{aligned}$$

$$+\sum_{W\in\mathcal{W}}\left(\mathrm{Adv}_{H}^{\mathrm{bhk}}(q_W,\sigma_{H_1,W})+\mathrm{Adv}_{H}^{\mathrm{bhk}}(q_W,\sigma_{H_2,W})+\binom{q_W}{2}\cdot 2^{-2\cdot n}\right),$$

where l_{U_R} and l_{V_L} are the total lengths of U_R and V_L respectively (refer figure 6), q_W is the total number of queries with tweak W and $\sigma_{H_1,W}$, $\sigma_{H_2,W}$ are the data complexities of the two instances of H .i.e., H_1 and H_2 respectively, K is key for H, K_1 is key for the first instance of S, n is the outside branch length .i.e., $|U_L| = |V_R| = n$ and $t' = t + O(l_T + l_M)$.

Proof: The design of tS-double-decker is similar to double-decker with the exception of a tweakable stream cipher (definition 3), instead of a deck function, and therefore we leave the proof for brevity. We encourage the reader to refer theorem 1 in [6] for the proof. ■

Theorem 5: Let tS-Docked-Double-Decker be the scheme defined in section VI-C2, where H is blinded keyed hash, then we define:

$$\begin{aligned}&\mathrm{Adv}^{\pm\widetilde{PRP}}_{\text{tS-Docked-Double-Decker}}(q,l_W,l_M,t)\\&\le \mathrm{Adv}^{\mathrm{tS}}_{S}(q,l_W,l_M-q\cdot n),t')\\&\quad+\mathrm{Adv}^{\mathrm{tS}}_{S}(q+1,l_W,l_M\\&\quad+|K|+|K_1|-q\cdot n,t')\\&\quad+\sum_{W\in\mathcal{W}}\left(\mathrm{Adv}_{H}^{\mathrm{bhk}}(q_W,\sigma_{H_1,W})+\mathrm{Adv}_{H}^{\mathrm{bhk}}(q_W,\sigma_{H_2,W})+\binom{q_W}{2}\cdot 2^{-2\cdot n}\right),\end{aligned}$$

where q_W is total number of queries with tweak W and $\sigma_{H_1,W}$, $\sigma_{H_2,W}$ are the data complexities of the two instances of H .i.e., H_1 and H_2 respectively, K is key for H, K_1 is key for the first instance of S, n is outside branch length .i.e., $|T| = |V| = n$ and $t' = t + O(l_T + l_M)$.

Proof: The design of tS-docked-double-decker is similar to docked-double-decker with the exception of a tweakable stream cipher (definition 3), instead of a deck function, and therefore we leave the proof for brevity. We encourage the reader to refer theorem 1 in [6] for the proof. ■

IX. CONCLUSION

In this paper, we provide detail analysis of CMT-4 security of four tweakable wide block cipher schemes under encode-then-encipher paradigm. We analyze the scenario of creating authentication encryption schemes by prepending zeros. We presented successful CMT-4 attack on EtE-HBSH in section V-A, EtE-HCTR2 in section V-B, and the two deck function based constructions in section V-C .i.e., EtE-double-decker in section V-C1 and EtE-docked-double in section V-C2 respectively. All the four attacks have time complexity $O(1)$ (refer table 3). We introduce the notion of tweakable stream cipher (or tS in short) in definition 3. We use tS to design four new tweakable wide block cipher schemes namely, HBtSH (refer section VI-A), HtS (refer section VI-B), tS-double-decker (refer section VI-C1) and tS-docked-double-decker (refer section VI-C2). All the four proposed schemes provide partial collision resistance against a CMT-4 adversary under encode-then-encipher paradigm (refer theorem 1 and theorem 3). Further, we provide security proof of HBtSH in theorem 2, ts-double-decker in theorem 4 and ts-docked-double-decker in theorem 5.

The notion of tweakable stream cipher opens a new direction. Several constructions can be analyzed for CMT-4 security under encode-then-encipher paradigm, and tweakable stream cipher can be explored to see the resilience against a CMT-4 adversary for the analyzed constructions. Further, in this paper, we use eXtendable-Output Function (or XOF) (refer definition 4) for creating a tweakable stream cipher. It can be of interest to explore other primitives that can be used as a tweakable stream cipher. From the implementation perspective, a tS based scheme may seem to be slower than a traditional stream cipher based construction or a deck function based construction, but on a positive note, XOF is an active area, where attempts are made to make it faster by reducing the number of rounds [47]. This also opens up a future research work in this area. Further, since, the committing security is very practical in the real-world setting, it is always good know the resilience of authenticated encryption schemes against such attacks.

APPENDIX A
FARFALLE

A. OVERVIEW

Farfalle [20] can be used for building a deck (doubly-extendable keyed) function. It takes bits of string and a key K as the input and returns an arbitrary-length output. From the architecture point of view, a Farfalle can be divided into two layers, a compression layer, followed by an expansion layer (refer figure 7). It can be instantiated using following notation:

$$\mathrm{Farfalle}[p_b,p_c,p_d,p_e,\mathrm{roll}_c,\mathrm{roll}_e],$$

where p_b, p_c, p_d and p_e are permutations and, roll_c and roll_e are rolling functions, such that:

- p_b: used for deriving the initial mask from the key K
- p_c: used in the compression layer
- p_d: used between the compression and expansion layer
- p_e: used in the expansion layer
- roll_c: used in the compression layer for generating masks that are added to the input blocks
- roll_e: used in the expansion layer to update the internal state

The rolling function, roll_c is a lightweight linear function with huge order [20], analogous to LFSR. As an example, XOOFFF [48] is based on Farfalle, with XOODOO [48] as the underlying permutation with six rounds for p_b, p_c, p_d and p_e and LFSR like rolling functions for roll_c and roll_e.

B. SPECIFICATION

Inside Farfalle, bits of string are processed in blocks of $\bar{b}$-bits or in multiple of $\bar{b}$ bits, where $\bar{b}$ denotes the width of

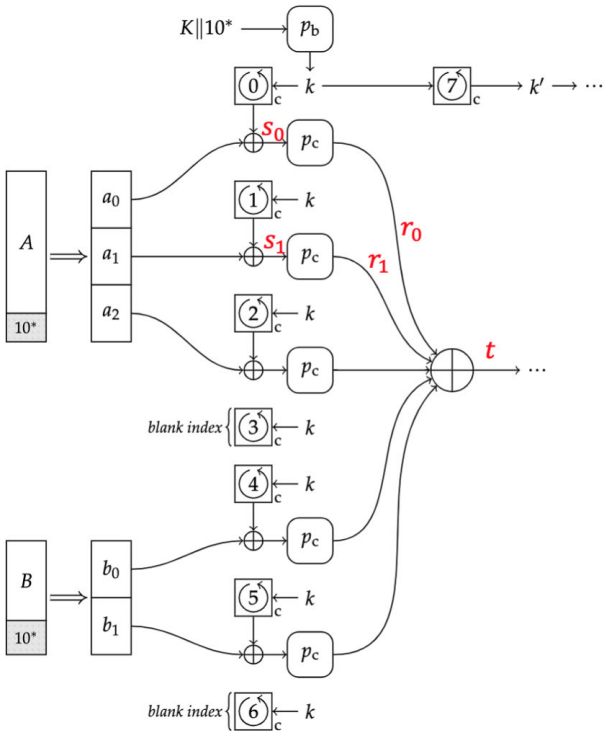


FIGURE 7. Compression layer of Farfalle (Adapted from [20]).

underlying permutation. Hence, a padding rule is needed to convert the input bits of string into multiple of $\bar{b}$-bits. For a given message $M \in \{0,1\}^*$, padding rule is defined as follows: $P = \text{pad10}^*(M)$, where $P \leftarrow (p_0 \parallel p_2 \parallel \cdots \parallel p_{n-1})$ and $\forall\, i \in \{0, n-1\}\ |p_i| = b$. Similarly, the key K is padded to $(K \parallel 10^*)$ before applying p_b. Two keys k and k' is derived from the input key K; k is used during the compression layer while k' is used during the expansion layer.

C. COLLISION ATTACK

For performing collision attack on Farfalle, the task the adversary $\mathcal{A}$ is to create two distinct inputs of Farfalle, such that the output generated by Farfalle for the two inputs are the same. From the construction, we know that if the output of the expansion layer, .i.e., 't' in figure 7, is same for the two inputs then the output produced by the expnasion layer (or Farfalle) is also same. Hence, we choose compression layer of Farfalle for demonstrating our collision attack. Our task is to construct two inputs: ENC(pad10*(A_1), pad10*(B)) and ENC(pad10*(A_2), pad10*(B)), where $A_1 \neq A_2$, such that the output of the expansion layer is t. We require that both A_1 and A_2 contains at least two full blocks .i.e., $|A_1| \geq 2 \bullet \bar{b}$ and $|A_2| \geq 2 \bullet \bar{b}$. We start our attack by fixing the key K, hence we obtain k and k' deterministically. Since, 't' is obtained by *xor* operation, it is trivial to find r'_0 and r'_1, that produces the same output 't'. Now, since p_c is a permutation, we can find values 's_0', 's_1', 's'_0' and 's'_1' such that $s_0 = p_c^{-1}(r_0)$, $s_1 = p_c^{-1}(r_1)$, $s'_0 = p_c^{-1}(r'_0)$ and $s'_1 = p_c^{-1}(r'_1)$. We know that the rolling function roll_c is linear, hence it is trivial to obtain two distinct $\bar{b}$-bit blocks a_{1_0} and a_{2_0}, such that a_{1_0} and a_{2_0} denotes the two $\bar{b}$-bit blocks of A_1; a'_{1_0} and a'_{2_0}, such that a'_{1_0} and a'_{2_0} denotes two $\bar{b}$-bit blocks of A_2. This leads to the fact that we have constructed two distinct inputs producing the same output. This concludes the collision attack on Farfalle.

REFERENCES

[1] S. Halevi and P. Rogaway, "A tweakable enciphering mode," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2003, pp. 482–499.

[2] P. Crowley and E. Biggers, "$*Adiantum*$: Length-preserving encryption for entry-level processors," *Cryptol. ePrint Arch.*, Dec. 2018.

[3] Android Open Source Project. (2024). *Enabling Adiantum on Android*. [Online]. Available: https://source.android.com/docs/security/features/encryption/adiantum

[4] P. Crowley, N. Huckleberry, and E. Biggers, "Length-preserving encryption with HCTR2," *Cryptol. ePrint Arch.*, Jan. 2021.

[5] Android Open Source Project. (2024). *File-Based Encryption*. [Online]. Available: https://source.android.com/docs/security/features/encryption/file-based

[6] A. Gunsing, J. Daemen, and B. Mennink, "Deck-based wide block cipher modes," *Cryptol. ePrint Arch.*, Jan. 2022.

[7] C. Dobraunig, K. Matusiewicz, B. Mennink, and A. Tereschenko, "Efficient instances of docked double decker with aes," *Cryptol. ePrint Arch.*, Jan. 2024.

[8] (2001). *Advanced Encryption Standard (AES)*. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=901427

[9] Mihir Bellare and Phillip Rogaway, "Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2000, pp. 317–330.

[10] (2024). *Proposal of Requirements for an Accordion Mode*. [Online]. Available: https://csrc.nist.gov/files/pubs/other/2024/04/10/proposal-of-requirements-for-an-accordion-mode-dis/iprd/docs/proposal-of-requirements-for-an-accordion-mode-discussion-draft.pdf

[11] V. T. Hoang, T. Krovetz, and P. Rogaway, "Robust authenticated-encryption AEZ and the problem that it solves," in *Proc. 34th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2015, pp. 15–44.

[12] *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. [Online]. Available: http://competitions.cr.yp.to/caesar.html

[13] (2023). *The Third NIST Workshop on Block Cipher Modes of Operation 2023*. [Online]. Available: https://csrc.nist.gov/events/2023/third-workshop-on-block-cipher-modes-of-operation

[14] (2023). *Key Committing Security of Aez and More*. [Online]. Available: https://csrc.nist.gov/csrc/media/Presentations/2023/key-committing-security-of-aez/images-media/sess-7-chen-bcm-workshop-2023.pd

[15] (2007). *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

[16] Y. Dodis, P. Grubbs, T. Ristenpart, and J. Woodage, "Fast message franking: From invisible salamanders to encryptment," in *Proc. Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA. Cham, Switzerland: Springer, Aug. 2018, pp. 155–186.

[17] P. Grubbs, J. Lu, and T. Ristenpart, "Message franking via committing authenticated encryption," in *Proc. 37th Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA. Cham, Switzerland: Springer, Aug. 2017, pp. 66–97.

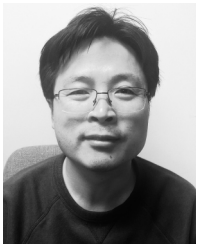
[18] A. Albertini, T. Duong, S. Gueron, S. Kölbl, A. Luykx, and S. Schmieg, "How to abuse and fix authenticated encryption without key commitment," in *Proc. 31st USENIX Secur. Symp. (USENIX Secur.)*, 2022, pp. 3291–3308.

[19] M. Bellare and V. T. Hoang, "Efficient schemes for committing authenticated encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer, 2022, pp. 845–875.

[20] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer, "Farfalle: Parallel permutation-based cryptography," *Cryptol. ePrint Arch.*, Dec. 2016.

[21] W. F. Ehrsam, C. H. Meyer, J. L. Smith, and W. L. Tuchman, "Message verification and transmission error detection by block chaining," U.S. Patent 4 074 066, Feb. 14, 1978.

[22] (2010). *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38e.pdf
[23] (2023). *Setting the Bitlocker Encryption Algorithm for Autopilot Devices*. [Online]. Available: https://learn.microsoft.com/en-us/autopilot/bitlocker
[24] R. Fujita, T. Isobe, and K. Minematsu, "ACE in chains: How risky is cbc encryption of binary executable files?" in *Proc. 18th Int. Conf. Appl. Cryptogr. Netw. Secur.*, Rome, Italy. Cham, Switzerland: Springer, Oct. 2020, pp. 187–207.
[25] (2018). *Lightweight Cryptography*. [Online]. Available: https://csrc.nist.gov/Projects/Lightweight-Cryptography
[26] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *J. Cryptol.*, vol. 34, no. 3, pp. 1–42, Jul. 2021.
[27] S. Banik, A. Chakraborti, T. Iwata, K. Minematsu, M. Nandi, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT-COFB," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 738, Jan. 2020.
[28] T. Iwata, M. Khairallah, K. Minematsu, and T. Peyrin, "Duel of the titans: The romulus and remus families of lightweight AEAD algorithms," *IACR Trans. Symmetric Cryptol.*, pp. 43–120, May 2020.
[29] M. Hasan and D. Chang, "Lynx: Family of lightweight authenticated encryption schemes based on tweakable blockcipher," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14357–14369, Apr. 2024.
[30] H. Wu and B. Preneel, "AEGIS: A fast authenticated encryption algorithm," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, Burnaby, BC, Canada. Cham, Switzerland: Springer, Aug. 2013, pp. 185–201.
[31] T. Isobe and M. Rahman, "Key Committing Security Analysis of AEGIS," *Cryptol. ePrint Arch.*, Jan. 2023.
[32] Y. Naito, Y. Sasaki, and T. Sugawara, "Committing security of ascon: Cryptanalysis on primitive and proof on mode," *IACR Trans. Symmetric Cryptol.*, vol. 2023, no. 4, pp. 420–451, Dec. 2023.
[33] Y. Naito, Y. Sasaki, and T. Sugawara, "KIVR: Context-committing authenticated encryption using plaintext redundancy and application to GCM and variants," in *Proc. NIST*, 2023.
[34] J. Krämer, P. Struck, and M. Weishäupl, "Committing Authenticated Encryption: Sponges vs. block-ciphers in the case of the nist lwc finalists," *Cryptol. ePrint Arch.*, Jan. 2023.
[35] S. Gueron, "Key committing AEADs," *Cryptol. ePrint Arch.*, Dec. 2020.
[36] P. Wang, D. Feng, and W. Wu, "HCTR: A variable-input-length enciphering mode," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Cham, Switzerland: Springer, 2005, pp. 175–188.
[37] H. Krawczyk, "LFSR-based hashing and authentication," in *Proc. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1994, pp. 129–139.
[38] B. Schneier and J. Kelsey, "Unbalanced feistel networks and block cipher design," in *Proc. Int. Workshop Fast Softw. Encryption*. Cham, Switzerland: Springer, 1996, pp. 121–144.
[39] D. J. Bernstein, "ChaCha, a variant of Salsa20," in *Proc. Workshop Rec. SASC*, Lausanne, Switzerland, vol. 8, no. 1, 2008, pp. 3–5.
[40] National Institute of Standards and Technology (NIST). (2001). *Advanced Encryption Standard (AES)*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf
[41] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *Proc. Annu. Int. Cryptol. Conf.*, Santa Barbara, CA, USA. Cham, Switzerland: Springer, Aug. 1999, pp. 216–233.
[42] D. J. Bernstein, "The Poly1305-AES message-authentication code," in *Proc. Int. Workshop Fast Softw. Encryption*. Cham, Switzerland: Springer, 2005, pp. 32–49.
[43] Y. L. Chen, A. Flórez-Gutiérrez, A. Inoue, R. Ito, T. Iwata, K. Minematsu, N. Mouha, Y. Naito, F. Sibleyras, and Y. Todo, "Key committing security of AEZ and more," *IACR Trans. Symmetric Cryptol.*, vol. 2023, no. 4, pp. 452–488, Dec. 2023.
[44] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," in *Proc. NIST*, 2015.
[45] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "On the indifferentiability of the sponge construction," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2008, pp. 181–197.
[46] (2015). *FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
[47] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, R. Van Keer, and B. Viguier, "TurboSHAKE," *Cryptol. ePrint Arch.*, Jan. 2023.
[48] J. Daemen, S. Hoffert, G. Van Assche, and R. Van Keer, "The design of xoodoo and xoofff," *IACR Trans. Symmetric Cryptol.*, pp. 1–38, Dec. 2018.



DONGHOON CHANG received the bachelor's degree in mathematics, the master's degree, and the Ph.D. degree in information security from Korea University, South Korea, in 2001, 2003, and 2008, respectively. In 2006, he was a Researcher with the University of Waterloo, Canada. Following this, he held a postdoctoral position with Columbia University, USA, from 2008 to 2009. Subsequently, he was a Guest Researcher with the Computer Security Division, National Institute of Standards and Technology (NIST), USA, from 2009 to 2012. Since 2012, he has been an Associate Professor with the Indraprastha Institute of Information Technology Delhi, India. From 2019 to 2021, he was with NIST, as a Guest Researcher. Since August 2021, he has been a Research Specialist with Strativia, USA.

MUNAWAR HASAN received the B.Tech. degree in computer science engineering from Dehradun Institute of Technology, Dehradun, India, in 2010, and the M.Tech. degree from Indraprastha Institute of Information Technology Delhi (IIIT-Delhi), New Delhi, India, in 2016, where he is currently pursuing the Ph.D. degree, under the supervision of Dr. Donghoon Chang. He was employed as a Software Engineer with DXC Technology (formerly Computer Sciences Corporation), from 2010 to 2013. He was a Senior Researcher with IRISYS Company Ltd., from 2016 to 2019. Since 2019, he has been with the National Institute of Standards and Technology (NIST), USA, as a Guest Researcher. His research interests include the design and analysis of authenticated encryption schemes, security proofs, and machine learning.