IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 3, pp. 1–35. DOI:10.46586/tosc.v2021.i3.1-35

1, 2, 3, Fork: Counter Mode Variants based on a Generalized Forkcipher

Elena Andreeva[1], Amit Singh Bhati[2], Bart Preneel[2] and Damian Vizár[3]

[1] Technische Universität Wien, Vienna, Austria, elena.andreeva@tuwien.ac.at
[2] imec-COSIC, Katholieke Universiteit Leuven, Leuven, Belgium
{amitsingh.bhati,bart.preneel}@esat.kuleuven.be
[3] Swiss Center for Electronics and Microtechnology (CSEM), Neuchâtel, Switzerland,
damian.vizar@csem.ch

Abstract. A multi-forkcipher (MFC) is a generalization of the forkcipher (FC) primitive introduced by Andreeva et al. at ASIACRYPT'19. An MFC is a tweakable cipher that computes s output blocks for a single input block, with s arbitrary but fixed. We define the MFC security in the ind-prtmfp notion as indistinguishability from s tweaked permutations. Generalizing tweakable block ciphers (TBCs, $s = 1$), as well as forkciphers ($s = 2$), MFC lends itself well to building simple-to-analyze modes of operation that support any number of cipher output blocks.

Our main contribution is the *generic* CTR encryption mode GCTR that makes parallel calls to an MFC to encrypt a message M. We analyze the set of all 36 "simple and natural" GCTR variants under the nivE security notion by Peyrin and Seurin from CRYPTO'16. Our proof method makes use of an intermediate abstraction called tweakable CTR (TCTR) that captures the core security properties of GCTR common to all variants, making their analyses easier. Our results show that many of the schemes achieve from well beyond birthday bound (BBB) to *full* n-bit security under nonce respecting adversaries and some even BBB and close to *full* n-bit security in the face of realistic nonce misuse conditions.

We finally present an efficiency comparison of GCTR using ForkSkinny (an MFC with $s = 2$) with the traditional CTR and the more recent CTRT modes, both are instantiated with the SKINNY TBC. Our estimations show that any GCTR variant with ForkSkinny can achieve an efficiency advantage of over 20% for moderately long messages, illustrating that the use of an *efficient* MFC with $s \geq 2$ brings a clear speed-up.

Keywords: Forkcipher · CTR mode · Encryption · Nonce · Tweak · MFC

1 Introduction

Forkcipher (FC) [ALP+19b] is a novel symmetric primitive, originally conceived for efficient authenticated encryption (AE) of short messages. It transforms a fixed length (n-bit) plaintext input X into a larger ($2n$-bit) fixed length output Y via a secret key K and an (optional) public tweak T. The security notion of an FC is given as indistinguishability from two pseudorandom tweakable permutations (ind-prtfp) [ALP+19b]. An FC is then used to build secure nonce-based AE schemes that require strictly one primitive (FC) call per message block. The FC modes for nonce-based AE (r)PAEF and SAEF are the first examples of such one-primitive-call-per-block constructions. Other nonce-based AE schemes, such as TAE [LRW02], ΘCB [RK11], GCM [MV04], CCM [WHF03], and OCB [RBB03], incur at the very least one additional primitive call. Combined with an *efficient* FC, the FC-based AE modes evidently minimize computational cost for short messages.

Received: 2021-06-01 Accepted: 2021-08-01 Published: 2021-09-17

Forkcipher applications beyond efficient short-message AE are still to be explored, especially their possible efficiency and security advantages over regular and tweakable ciphers. For example, recently, a stronger security for the forkcipher based AE mode SAEF has been proved to support its defense in depth. Additional to the classical nAE [Rog02] security, SAEF has been shown secure [ABV20, ABV21, ABD+20] under both OAE [FFL12] and INT-RUP [ABL+14] security notions.

In this work we focus on the applications of forkciphers to counter(CTR)-mode-like encryption. CTR mode is one of the most deployed symmetric encryption schemes. Its applications include random number generator AES-CTR DRNG [BK07], asynchronous transfer modes, network security: TLS/SSL and low power protocols, and IP security, among others. Furthermore, many standardized AE modes, such as GCM [MV04], CCM [WHF03], and SIV [RS06] make use of the CTR mode internally.

CTR mode was introduced in 1979 [LRW00] and is part of the US National Institute of Standards and Technology NIST SP 800-38A Recommendation for block cipher modes of operation. In his survey, P. Rogaway says "*I regard CTR as the "best" choice among the classical confidentiality-only techniques. Its parallelizability and obvious correctness, when based on a good blockcipher, mean that the mode should be included in any modern portfolio of modes.*", and "*Overall, usually the best and most modern way to achieve privacy-only encryption*" [Rog11].

For an underlying block cipher E_K with a key K, a message M, an initialization vector IV (either a nonce as a non-repeating value, or a random value), and a counter j, the classical CTR mode is defined as $c_j = E_K(f(IV, j)) \oplus M_j$. The output of f is the *counter block* and is a unique input to each call of the block cipher. A secure and hence typical choice of f is the simple concatenation operation, or $f(IV, j) = IV \| j$, e.g., the IV takes the upper 64 bits and the counter takes the lower 64 bits of a 128-bit counter block. For a non-repeating nonce IV, CTR mode is indistinguishable from random bits under chosen plaintext attack (CPA). If the nonce is reused, the CTR security is completely compromised. When E_K is a 128-bit block cipher, such as the AES-128, the CTR mode achieves confidentiality under CPA up to the birthday bound (BB), that is up to 2^{64} encrypted data blocks, assuming that E_K is a secure pseudorandom permutation (PRP). The CTR mode's most desirable features are the *forward-only* primitive operation in both encryption and decryption (known as the inverse-free property), and the full parallelizability. These make CTR particularly efficient and well-suited for modern architectures with multiple cores and SIMD extensions where blocks can be encrypted (and decrypted) in parallel.

Although classical blockciphers, such as the AES, are still broadly used, a new class of tweakable blockciphers [LRW02] (TBC), such as CRAFT [BLMR19], Deoxys [JNPS16], SKINNY [BJK+16], etc., has proliferated in the last decade. A tweakable cipher takes an additional public input called tweak. The tweak is used to ensure both the cipher *"variability"* and increased resistance against precomputation attacks.[1] The variability factor is particularly useful when analyzing security of TBC-based modes: if two AES calls are made with the same plaintext block X, the result will be identical, yet, under distinct tweaks T_1 and T_2 a "good" (indistinguishable from a tweak-indexed collection of random permutations) TBC returns computationally independent ciphertexts $E_K^{T_1}(X)$ and $E_K^{T_2}(X)$.

The CounTeR in Tweak (CTRT) encryption mode was proposed by Peyrin and Seurin [PS16]. It is a TBC-based CTR-style encryption mode where the tweak value T is set to the counter block computed as the XOR of the random IV value and a counter ($IV \oplus j$), and the cipher input value X is set to a unique nonce value N (fixed per message). For a

[1] TBCs (including those based on the TWEAKEY framework thanks to near-independent key and tweak-related computations in the tweakey schedule [JNP14]) typically achieve these properties more efficiently than having to re-key a blockcipher.

block size of n bits and a tweak of t bits, the CTRT mode achieves *beyond birthday bound* (BBB)[2] $(n+t)/2$-bit security under the nivE notion (defined as indistinguishability from random bits with fresh nonces for each encryption), and a graceful security degradation when the nonce is repeated. The CTRT mode retains the features of the classical CTR mode and, in addition, brings in beyond birthday bound security and improved resistance against nonce misuse.

Nonce-misuse resistance (NMR) has been considered a theoretical abstraction, but recent attacks illustrated its severity in practice. In USENIX'16 Böck et al. [BZD+16] investigated NMR of the CTR-based AES-GCM deployed in TLS 1.3 and managed to completely break the authenticity of those connections where servers repeated nonces. The next year at CCS'17 [VP17] Vanhoef and Piesens introduced the key reinstallation attack which forces nonce repetitions and breaks the WPA2 wireless protocol.

Contributions

In this work, we explore the security, efficiency and resulting advantages of tweakable primitives in CTR-style encryption through the prism of multiforkciphers. Our main reference points are the classical CTR and the recent CTRT encryption modes.

Multiforkciphers. First, we extend the forkcipher definition to that of a generic multiforkcipher (MFC). MFC is a tweakable cipher with an arbitrary but fixed length output, which generalizes over and covers both TBCs and tweakable FCs. An MFC transforms a single input block into $s \geq 1$ output blocks. When $s = 1$, MFC becomes a TBC and when $s = 2$, it becomes an FC. We present the MFC security definition − ind-prtmfp (indistinguishable pseudorandom tweakable multi-fork permutation), which similarly to the ind-prtfp notion [ALP+19b], captures indistinguishability from *multiple* pseudorandom tweakable permutations. We use MFCs to tackle the security of CTR-style encryption schemes. When s is left as a fixed but arbitrary parameter in a security analysis, one obtains a result valid simultaneously for TBCs, FCs and (hypothetical) MFCs with $s > 2$. This means, for example, that our results are directly applicable to both TBC-based and FC-based instances of CTR-like modes, among others.

Generic CTR mode. Our main contribution is the novel *generic* counter GCTR structure that uses an MFC as its underlying primitive. GCTR makes parallel MFC calls where the MFC inputs X (the plaintext) and T (tweak) are determined via two *input generator functions* f_X and f_T, taking as input a nonce N, a random IV denoted by R, and a counter j. We focus on the simplest and most natural generator functions, defined as either the concatenation, XOR, or the copy operations of two (respectively one) out of these three inputs, or simply a constant function independent of its inputs. We identify 36 instances, (most of) which implement secure nonce based, or IV-based or nivE schemes. In the special case of MFC with $s = 1$ (TBC), our results include, and offer alternatives to CTRT, which coincides with GCTR-3. To the best of our knowledge, this is the first systematic treatment of the popular CTR-style encryption.

TCTR abstraction. We analyze the security of *all* our GCTR variants. To do so, we define the tweakable CTR (TCTR) as an intermediate abstraction. TCTR directly takes two sequences of tweaks and plaintexts and feeds the input (T, X) pairs into parallel calls to MFC to generate a key stream. The security of TCTR is defined via the concept of a sequence-builder, capturing the common properties of all GCTR variants. We then bound the generic distinguishing advantage between the TCTR output and a truly random key stream, and apply this result in the analysis of our GCTR variants.

[2] We understand *birthday-bound security* of modes as $n/2$ bit security in the "blocksize" n of the BC/TBC/FC/MFC, as does a majority of recent publications.

Security. Our results show interesting security advantages overall, and particularly improve over the classical CTR mode. We prove that some of our variants achieve security beyond the birthday bound of $n/2$ bit (BBB) and some even full n-bit security with n being the size of the input blocks. We provide a detailed interpretation of our security results in Sect. 5 and pick a selection of variants GCTR-3 (= CTRT when $s = 1$), and GCTR-7 that excel in security. For a total of σ MFC calls, GCTR-7 provides *perfect* information-theoretic security against nonce-respecting adversaries and BBB-security against nonce misusing adversaries (for $\ll \sigma$ nonce repetitions). GCTR-3 comes with BBB-security against nonce misusing adversaries (for $\ll \sigma$ nonce repetitions). Our security bound for GCTR-3 additionally *improves* over the original bound of CTRT. CTRT [PS16] was proven BBB secure with a degradation in the bound (nonce misuse and nonce respecting) as $\frac{2\sigma(x-1)}{2^t} + \frac{\sigma^2}{2^{n+t+1}}$ where q and x are the number of total CTRT queries and the maximum number of nonce repetitions over CTRT queries, respectively. In this work, we reduce the gap to $\frac{(2\sigma-q)(x-1)}{2^{t+1}} + \frac{(q-1)(2\sigma-q)}{2^{n+t+1}}$. Our results also show that GCTR-3 with larger tweaks of $2n$ bits provide $\approx n$-bit security for all adversary (nonce-misuse and respecting) types (see Sect. 5). To achieve these security improvements, however, we may need to pay with an increase in the communication bandwidth with the nonce/IV size as compared to regular CTR mode.

Revisiting Tweakable HCTR. We reanalyze Tweakable HCTR (or THCTR; a VIL enciphering scheme [DN18]) that uses as an internal building block a CTR-like encryption mode that is in fact equal to our GCTR-4. We invalidate its existing security bound (claiming beyond birthday security with respective TSPRP [DN18] security notion) by identifying a flaw in its existing proof. Further, we provide a birthday attack confirming that THCTR does not achieve TSPRP-security beyond the birthday bound in its present form and recommend to replace its internal GCTR-4 component by either of our preferred variants (namely, GCTR-3 or GCTR-7) to achieve the intended BBB-security.

Efficiency. A large part of our motivation for the study of GCTR variants is the idea that an MFC with a large s, that is more efficient than s TBC calls, results in more efficient encryption, with the advantage accumulating as the message grows. Our findings in Fig. 4 confirm that the only existing MFC with $s > 1$ ForkSkinny [ALP+19b, ALP+19a] yields a more efficient encryption scheme than its TBC counterpart (with identical round and tweakey functions) SKINNY [BJK+16] ($s = 1$) when plugged in GCTR. For example, ForkSkinny in any of our GCTR modes achieves an efficiency improvement of over 20% over SKINNY in GCTR modes for the same tweak and nonce sizes.

2 Preliminaries

All strings used in this paper are binary strings. Strings of length $n > 0$ are referred to n-bit strings. The set of all n-bit strings is denoted as $\{0,1\}^n$. Any sequence of n-bit strings is denoted by $(\{0,1\}^n)^+$. We denote the set of all permutations of $\{0,1\}^n$ by $Perm(n)$. For any string A, $|A|$ represents the bit-length of A and $\mathsf{trunc}_c(A)$ represents the string defined by the first c bits of A. For a set $\mathcal{S}$, the notation $2^{\mathcal{S}}$ denotes the power set of $\mathcal{S}$ and $|\mathcal{S}|$ denotes the size of $\mathcal{S}$. For any real number r, $\lceil r \rceil$ denotes the least integer which is greater than r (the ceiling function). We denote any vector B with components $B_1, B_2, \ldots, B_i$ as $\langle B_1, B_2, \ldots, B_i \rangle$. For any two numbers a and b, $a \cdot b$ or ab represents their scalar multiplication.

Given a string A and an integer n with $|A| = cn + d$ for some $0 < d \leq n$, we use the notation $A_1, A_2, \ldots, A_{c+1} \xleftarrow{n} A$ to represent the partitioning of A into a maximum number of n-bit blocks, such that $|A_i| = n$ for $1 \leq i \leq c$ and $|A_{c+1}| = d$. The symbol $\perp$ represents an undefined value or error. We let $r \xleftarrow{\$} \mathcal{R}$ denote the random sampling of an element

r from a finite set $\mathcal{R}$ considering the uniform distribution. We let $\mathbb{N}$ denote the set of natural numbers.

The notation $(p)_q$ denotes the falling factorial $p \cdot (p-1) \cdot (p-2) \cdot \ldots \cdot (p-q+1)$ where $(p)_0 = 1$. A predicate $\mathsf{P}(x)$ is defined as $\mathsf{P}(x) = 1$ if it is true and $\mathsf{P}(x) = 0$ if it is false. All comparisons that are used in the work for integer tuples are lexicographic comparisons (to exemplify, $(i', j') < (i, j)$ iff $i' < i$ or $i' = i$ and $j' < j$).

2.1 Nonce- and IV-based Encryption

We target the syntax and security of nonce- and IV- based encryption schemes (nivE) [PS16]. An nivE scheme is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where $\mathcal{K}$ is a key distribution (typically a finite key space with uniform distribution), $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{R} \times \mathcal{M} \to \{0,1\}^*$ is the deterministic encryption algorithm, and $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{R} \times \{0,1\}^* \to \mathcal{M}$ is the deterministic decryption algorithm with $\mathcal{N}$ and $\mathcal{R}$ representing the sets of nonces and IVs, respectively. The encryption algorithm maps a key K, a nonce N, an IV R and a message M with $(K, N, R, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{R} \times \mathcal{M}$ to a ciphertext $C = \mathcal{E}(K, N, R, M)$ and the decryption maps key, nonce, IV and a ciphertext to a message $M = \mathcal{D}(K, N, R, C)$.

We require that $\mathcal{D}(K, N, R, \mathcal{E}(K, N, R, M)) = M$ for all $K, N, R, M \in \mathcal{K} \times \mathcal{N} \times \mathcal{R} \times \mathcal{M}$ and we assume that both $\mathcal{E}$ and $\mathcal{D}$ return $\perp$ if any of the inputs is not in its intended domain. In this paper, we further require that $\mathcal{R}$ is a finite set, $\mathcal{M} \subset \{0,1\}^*$ such that $M \in \mathcal{M}$ and $|M| = m \Rightarrow \{0,1\}^m \subseteq \mathcal{M}$ and that $|\mathcal{E}(K, N, R, M)| = |M|$. We let $\mathcal{E}^\$: \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{R} \times \{0,1\}^*$ denote the randomized encryption algorithm, which internally samples an $R \leftarrow^\$ \mathcal{R}$ with uniform distribution, computes $C \leftarrow \mathcal{E}(K, N, R, M)$ and returns R, C. We further let $\mathcal{E}_K(N, R, M) = \mathcal{E}(K, N, R, M)$ and $\mathcal{E}^\$_K(N, M) = \mathcal{E}^\$(K, N, M)$.

nivE Security and ivE Security. We define the security of nivE through indistinguishability of ciphertexts from random strings in a chosen plaintext attack. More precisely, given an nivE scheme Π and a nonce respecting (i.e., using a fresh nonce for each encryption query) adversary $\mathcal{A}$, we define $\mathcal{A}$'s advantage at breaking the security of Π as $\mathbf{Adv}^{\mathrm{nivE}}_\Pi = \left| \Pr\left[K \leftarrow^\$ \mathcal{K} : \mathcal{A}^{\mathcal{E}^\$_K(\cdot,\cdot)} \Rightarrow 1 \right] - \Pr\left[\mathcal{A}^{\mathsf{Rand}^\$(\cdot,\cdot)} \Rightarrow 1 \right] \right|$, where $\mathsf{Rand}^\$(N, M)$ internally samples an $R \leftarrow^\$ \mathcal{R}$ and returns R with an independent random string of $|M|$ bits *upon every query.* If $\mathcal{A}$ is not nonce-respecting (may reuse a nonce), we define its advantage with the same experiment, but denote it as $\mathbf{Adv}^{\mathrm{ivE}}_\Pi$.

Relation to Nonce-based and IV-based Encryption. The syntax and notion of nivE schemes capture both nonce-based encryption (with $\mathcal{R} = \{\varepsilon\}$ which makes $\mathcal{E}^\$$ deterministic) and random initialization vector-based encryption (with $\mathcal{N} = \{\varepsilon\}$). Beyond these two basic types of encryption, nivE also captures a generalized type of symmetric encryption that uses a nonce and an IV simultaneously, previously shown useful to achieve high security levels [PS16].

On the use of nivE notion and schemes. At the first glance, the reader may question the usefulness of the "true" nivE schemes. Practice-wise, an encryption scheme requiring a nonce and an IV to be transmitted would indeed not be the first choice for mainstream applications. There are nevertheless scenarios where nivE schemes are useful as-is. For example, in an encryption-only scenario where two-pass processing is unacceptable and nonce-misuse resistance is desirable (e.g., a micro controller with embedded TRNG and constrained RAM, streaming sensor-data and having a reset-related possibility of nonce-repetition), a nonce-IV scheme seems the only viable option.

In addition, nivE schemes are useful building blocks for higher-level constructions, where the nonce, the IV, or both can be implicit, as exemplified by SCT [PS16] (AEAD) and HCTR [DN18] (enciphering). Generally speaking, the benefits of nivE can be leveraged wherever an implicit nonce exist or some form of a synthetic IV can be computed. We conjecture that this is the case in many constructions and communication protocols (think about TLS, where each frame has a sequence number). Our results can then be used as a blackbox in the analyses of such constructions, as seen on the example of HCTR.

Finally, the nivE *definition* is an umbrella notion, that captures nonce-based, IV-based and nonce-IV-based symmetric encryption schemes. This lets us characterize and study interesting constructions of all three types simultaneously, dispensing with the need for a dedicated treatment for each type.

2.2 Coefficient-H Technique

The coefficient-H technique is a simple but powerful proof technique by Patarin [Pat09] which is often used to prove indistinguishability of a given construction from an idealized object by an information-theoretic adversary. The Coefficient-H technique characterizes an indistinguishability experiment, in which an information-theoretic adversary $\mathcal{A}$ tries to distinguish two sets of oracles $\mathcal{O}_{\mathsf{real}}$ (the "real world") and $\mathcal{O}_{\mathsf{ideal}}$ (the "ideal world"), in the form of *transcripts.* A transcript is defined as a complete record of the interaction of an adversary $\mathcal{A}$ with its oracles. To exemplify, if $\mathcal{A}$ has a single oracle, (M_i, C_i) representing the input and output of the i-th query to this oracle and q is the total number of queries made by $\mathcal{A}$ then the corresponding transcript (denoted by τ) is defined as $\tau = \langle (M_1, C_1), \ldots, (M_q, C_q) \rangle$. The goal of $\mathcal{A}$ here is to distinguish interactions in the real world $\mathcal{O}_{\mathsf{real}}$ from the ones in the ideal world $\mathcal{O}_{\mathsf{ideal}}$.

Let us denote the distribution of transcripts in the real and in the ideal world by Θ_{real} and Θ_{ideal}, respectively. We call a transcript τ *attainable* if the probability of achieving τ in the ideal world is non-zero. Further, we also assume w.l.o.g. that $\mathcal{A}$ does not make any duplicate or prohibited queries. We can now state the fundamental lemma of the coefficient-H technique. We refer the reader to an excellent tutorial on the coefficient-H technique by Chen and Steinberger [CS14].

Lemma 1 (Fundamental Lemma of the coefficient H Technique [Pat09])**.** *Consider that the set of attainable transcripts is partitioned into two disjoint sets $\mathcal{T}_{\mathsf{good}}$ and $\mathcal{T}_{\mathsf{bad}}$. Also, assume there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in \mathcal{T}_{\mathsf{good}}$, we have $\frac{\Pr[\Theta_{\mathsf{real}}=\tau]}{\Pr[\Theta_{\mathsf{ideal}}=\tau]} \geq 1 - \epsilon_1$, and $\Pr[\Theta_{\mathsf{ideal}} \in \mathcal{T}_{\mathsf{bad}}] \leq \epsilon_2$. Then, for all adversaries $\mathcal{A}$, it holds that $|\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{real}}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{ideal}}} \Rightarrow 1]| \leq \epsilon_1 + \epsilon_2$.*

3 Multi-Fork Cipher (MFC)

In this section, we define the syntax and security of symmetric key primitive we name *multi-forkcipher* (MFC). MFC generalizes the primitive called *forkcipher* [ALP$^+$19b]. Informally, a forkcipher takes as input a secret key, a public tweak and an input block, and evaluates *two* independent permutations of the input block at the same time.[3] An MFC generalizes this concept to an arbitrary (but fixed) number of encryption branches (i.e., arbitrary but fixed number of output blocks). More precisely, a multi-forkcipher takes a secret key, a public tweak, and an n-bit plaintext block as input and produces s n-bit output blocks. Additionally, the input X should be computable backwards from any of the output blocks, and any of the output blocks should be *reconstructible* from any other output block.

Ideal MFC. With a random key as input, an ideal s-MFC implements an s-tuple of independent random permutations $\pi_{T,1}, \pi_{T,2}, \ldots, \pi_{T,s}$ for every tweak T, which for input X and a set $\alpha \subseteq \{1, 2, \ldots, s\}$ provides $\{v_i \mid v_i = \pi_{T,i}(X) \text{ for } i \in \alpha\}$ i.e. $|\alpha|$ many indexed but independent outputs. We define a secure multi-forkcipher to be computationally indistiguishable from such an ideal MFC. The ideal MFC is equivalent to a tuple of s ideal tweakable block ciphers used in parallel.

[3] A trivial forkcipher can be constructed by concatenating the output of two tweakable blockciphers, however a dedicated forkcipher is typically more efficient than a pair of TBCs.

Note that this formalism captures both conventional TBCs (MFC with a single output block), the original forkcipher, and any future generalized constructions with three or more branches. Results using the notion of MFC then have the advantage of being automatically applicable to instantiations based on any of the aforementioned primitives. Furthermore, forking primitives that can output a practically unlimited number of branches (akin to Farfalle [BDH+17]) can be viewed as an MFC with the (maximum) number of branches set to their operational (or security) limits.

3.1 Syntax

A multi-forkcipher F_s is a pair of deterministic algorithms, the forward F_s:

$$\mathsf{F}_s : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \times 2^{\{1,2,\ldots,s\}} \rightarrow \bigcup_{e=1}^{s} \{0,1\}^{en}$$

and the backward (or the inverse) F_s^{-1}:

$$\mathsf{F}_s^{-1} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \times \{1,2,\ldots,s\} \times 2^{\{\mathsf{i},1,2,\ldots,s\}} \rightarrow \bigcup_{e=1}^{s} \{0,1\}^{en} \,.$$

The forward algorithm F_s takes in a key K, a tweak T, an input block X and an output selector set α. It then outputs the output blocks $Y_{a_1}, \ldots, Y_{a_z}$ indicated by the output selector $\alpha = \{a_1, a_2, \ldots, a_z\}$, such that $a_1 < a_2 < \ldots < a_z$. We let $F_s(K,T,X,\alpha) = F_{s,K}(T,X,\alpha) = F_{s,K}^{T}(X,\alpha)$.

Similarly, the backward computing algorithm F_s^{-1} takes in a key K, a tweak T, a block Y, an input indicator β and an output selector set α. It then outputs the blocks $Y_{a_1}, \ldots, Y_{a_z}$ indicated by $\alpha = \{a_1, a_2, \ldots, a_z\}$. If $a_1 = \mathsf{i}$ then the first block is X (the corresponding input block of F_s) and $a_2 < \ldots < a_z$, otherwise $a_1 < a_2 < \ldots < a_z$. If $\beta \in \alpha$, then $F_s^{-1}(K,T,Y,\beta,\alpha) = \perp$. We let $F_s^{-1}(K,T,Y,\beta,\alpha) = F_{s,K}^{-1}(T,Y,\beta,\alpha)$. We call k, n and t the keysize, inputsize and tweaksize of F_s, respectively.

A multi-forkcipher is said to be correct if for every $K \in \{0,1\}^k, T \in \{0,1\}^t, X, Y \in \{0,1\}^n$ and $\beta \in \{1,2,\ldots,s\}$ it satisfies the following conditions:

1. $\mathsf{F}_{s,K}^{-1}(T, \mathsf{F}_{s,K}(T,X,\beta), \beta, \mathsf{i}) = X$, i.e., decrypting a ciphertext block with the same key, the same tweak and using the same output index gives the correct plaintext,

2. $\mathsf{F}_{s,K}^{-1}(T, \mathsf{F}_{s,K}(T,X,\beta), \beta, \alpha) = \mathsf{F}_{s,K}(T,X,\alpha)$ for all $\alpha \in 2^{\{1,2,\ldots,s\} \setminus \{\beta\}}$, i.e., fixing the key and the tweak and given a ciphertext block produced with output index β, reconstructing the ciphertext block for output index α always gives the same value as encrypting the same plaintext directly with the output index α,

3. $(\mathsf{F}_{s,K}(T,X,a_1), \ldots, \mathsf{F}_{s,K}(T,X,a_z)) = \mathsf{F}_{s,K}(T,X,\{a_1, a_2 \ldots, a_z\})$ for each set $\{a_1, a_2, \ldots, a_z\} \in 2^{\{1,2,\ldots,s\}}$, i.e., fixing the key and the tweak, encrypting a plaintext with a certain set of output indexes always produces the same output blocks as encrypting the same plaintext with each of the output indexes individually,

4. $(\mathsf{F}_{s,K}^{-1}(T,Y,\beta,a_1), \ldots, \mathsf{F}_{s,K}^{-1}(T,Y,\beta,a_z)) = \mathsf{F}_{s,K}^{-1}(T,Y,\beta,\{a_1,a_2,\ldots,a_z\})$ for each set $\{a_1, a_2, \ldots, a_z\} \in 2^{\{\mathsf{i},1,2,\ldots,s\} \setminus \{\beta\}}$, i.e., fixing the key and the tweak and given a ciphertext block, reconstructing/decrypting with a certain set of output indexes always produces the same output blocks as reconstructing/decrypting the same ciphertext blocks with each of the output indexes individually.

3.2 Security of MFC

We define the security of a multi-forkcipher with the help of security games **prtmfp-real** and **prtmfp-ideal** in Fig. 1.[4] An adversary $\mathcal{A}$ who wants to break the multi-forkcipher F_s plays games **prtmfp-real** or **prtmfp-ideal**. In either game, $\mathcal{A}$ makes q queries in total, of the form (T^i, X^i, α^i) to the encryption oracle, or $(T^i, Y^i, \beta^i, \alpha^i)$ to the decryption oracle for $1 \leq i \leq q$. The oracle either processes the inputs with the real F_s used with a random key, or with a random "multi-forked permutation" P. A multi-forked permutation is an s-tuple of tweakable permutations, s.t. these s permutations are always used with the same plaintext block (even when queried in the backward direction). The selection of the tweakable permutations to be applied is based on the selector α in the natural way. We define the advantage of $\mathcal{A}$ at distinguishing F_s from a random multi-forked permutation P of $|\alpha| \cdot n$ bits in a chosen ciphertext attack as $\mathbf{Adv}^{\text{prtmfp}}_{\mathsf{F}_s}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathbf{prtmfp\text{-}real}_{\mathsf{F}_s}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{prtmfp\text{-}ideal}_{\mathsf{F}_s}} \Rightarrow 1]\right|$.

We will use a shorthand $[s]$ to denote the set $\{1, 2, \ldots, s\}$. In the rest of this paper, we only use the forward direction of an MFC, with $\alpha = [s]$. Thus, we fix $\alpha = [s]$, drop the output selector from the input list, and use the notation $F_s(K, \cdot, \cdot) = F_{s,K}(\cdot, \cdot) = \mathsf{F}_{s,K}(\cdot, \cdot, [s])$. One can see this F as a multi-forkcipher with α hardwired to "all".

MFC vs TPRI. An n-bit MFC with s branches syntactically resembles an n-bit input tweakable pseudorandom injection PRI with sn-bit output, yet they differ in their probability distributions. While a TPRI simply samples sn-bit images w/o replacement, the MFC concatenates s random permutations, resulting in a birthday gap between the two objects.

```
Game prtmfp-real_{F_s}
    $K \xleftarrow{\$} \mathcal{K}$
    $b \leftarrow \mathcal{A}^{\mathcal{E},\mathcal{D}}$
    return $b$

    Oracle $\mathcal{E}(T, X, \alpha)$
    return $\mathsf{F}_{s,K}(T, X, \alpha)$

    Oracle $\mathcal{D}(T, V_\beta, \beta, \alpha)$
    return $\mathsf{F}^{-1}_{s,K}(T, V_\beta, \beta, \alpha)$
```

```
Game prtmfp-ideal_{F_s}
    for $T \in \{0,1\}^t$ do $\pi_{T,1}, \pi_{T,2}, \ldots, \pi_{T,s} \xleftarrow{\$} Perm(n)$
    $b \leftarrow \mathcal{A}^{\mathcal{E},\mathcal{D}}$
    return $b$

    Oracle $\mathcal{E}(T, X, \alpha)$        //$\alpha = \{a_1, a_2, \ldots, a_z\}$
    return $\pi_{T,a_1}(X), \ldots, \pi_{T,a_z}(X)$

    Oracle $\mathcal{D}(T, Y, \beta, \alpha)$     //$\alpha = \{a_1, a_2, \ldots, a_z\}$
    if $\beta \in \alpha$ then return $\perp$
    $X \leftarrow \pi^{-1}_{T,\beta}(Y)$
    if $a_1 = \mathsf{i}$ then return $X, \pi_{T,a_2}(X), \ldots, \pi_{T,a_z}(X)$
    else return $\pi_{T,a_1}(X), \ldots, \pi_{T,a_z}(X)$
```

Figure 1: Games $\mathbf{prtmfp\text{-}real}_{\mathsf{F}_s}$ and $\mathbf{prtmfp\text{-}ideal}_{\mathsf{F}_s}$ defining the security of the multi-forkcipher F_s.

4 MFC-based CTR Mode and its Variants

The Counter (CTR) [LRW00] mode of operation has been considered as one of the best choices among the set of block cipher modes for message confidentiality. The inverse-freeness and parallelism of the original CTR mode are simple but very powerful in confidentiality-only protocols. Yet, the classical CTR mode provides only $n/2$ security when used with an n-bit block cipher and fails completely in the face of nonce reuse (in the cases where IV is implemented as a nonce). In this section, we define a *generic* CTR (GCTR) with

[4] The CCA variant of this notion is easily obtained by giving $\mathcal{A}$ access to the decryption oracle, which is naturally defined in both games. We do not formally add this definition as it is not necessary for this paper.

the same design properties as the original CTR construction but aiming to achieve higher security levels in the spirit of the recent tweakable block-cipher-based CTRT mode [PS16] and aiming for the nivE security notion.

GCTR implements an nivE encryption scheme that uses an MFC as a lower-level primitive and similarly to CTRT takes as input both a nonce and a random value, as opposed to the classical CTR mode's single IV value. We then show that at the cost of an additional input we obtain encryption schemes with significantly improved security. More precisely, we present and investigate the security of 36 concrete instances which are distinguished by the way a nonce N, an IV (a random value R) and a counter j are combined as inputs to the MFC primitive. Our research is exhaustive regarding the simplest of operations (XOR, copy or concatenation), covering the classical CTR as well. Furthermore, via the abstraction of MFC we incorporate and enable the comparison of security and efficiency features of tweakable primitives with variable output sizes.

4.1 Generic CTR

There are many possible definitions of a CTR-like mode with an MFC by combining the MFC tweak and plaintext inputs: a nonce N, a block counter j, and a random IV value denoted by R. We formally capture the space of the MFC-based CTR-like modes through the *generic CTR mode* (GCTR), which uses placeholder functions $f_T(N, R, j) \mapsto T$ and $f_X(N, R, j) \mapsto X$ to compute the tweak T and the MFC plaintext X. An instance, or else a GCTR *variant*, is obtained by fixing those functions.

For a fixed multi-forkcipher $F_s : \mathcal{K} \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^{sn}$ and two functions $f_T : \{0,1\}^\nu \times \{0,1\}^r \times \mathbb{N} \to \{0,1\}^t$ and $f_X : \{0,1\}^\nu \times \{0,1\}^r \times \mathbb{N} \to \{0,1\}^n$, GCTR$[F_s, f_T, f_X]$ mode implements an nivE scheme with key space $\mathcal{K}$, nonce space $\mathcal{N} = \{0,1\}^\nu$, IV space $\mathcal{R} = \{0,1\}^r$, message space $\mathcal{M} = \bigcup_{i=0}^{\ell} \{0,1\}^i$ and the encryption and decryption algorithm defined in Fig. 2. The exact values of ν, r and ℓ depend on the concrete instantiation. We also use the shorthand GCTR, leaving F_s, f_T and f_X implicit.

```
GCTR$_s[F_s, f_T, f_X]$ Encryption: $\mathcal{E}_K(N, M)$
$R \xleftarrow{\$} \mathcal{R}$
$M_1, \ldots, M_\ell \xleftarrow{n} M$
for $j \leftarrow 1$ to $\lceil \ell/s \rceil$ do
    $X_j = f_X(N, R, j)$
    $T_j = f_T(N, R, j)$
    $S_j = \mathsf{trunc}_{|M_{(j-1)s+1}\|\cdots\|M_{\min(js,\ell)}|}(F_{s,K}(T_j, X_j))$
$S \leftarrow S_1 \| \cdots \| S_{\lceil \ell/s \rceil}$
$C \leftarrow M \oplus S$
return $R, C$
```

Figure 2: Encryption algorithm of the generic CTR mode instantiated with a MFC $F_s : \mathcal{K} \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^{sn}$. Different definitions of the functions f_X and f_T yield concrete variants of the GCTR (see Table 2).

Similar to the conventional CTR mode, GCTR is inverse-free, i.e., the inverse direction of the underlying multi-forkcipher F_s is never used. We note that the security of GCTR mode depends not only on the security of the underlying multi-forkcipher but also on the functions f_X and f_T that compute MFC inputs and tweaks. In the next section, we exhaustively investigate a well-defined subset of the GCTR variants' space.

4.2 GCTR Variants: CTR Mode of Encryption using MFC

The space of all possible GCTR instances is huge (there are $2^{(t+n)\cdot(2^{\nu}\cdot 2^{r}\cdot j_{\max})}$ of them with $\nu = |N|$, $r = |IV|$ and $j_{\max}$ being the maximal allowed counter value) but only a significantly smaller subset of those is of practical interest. The main criterion is computational complexity of the functions f_X and f_T; they must be computed efficiently for the instance to make any sense. In this section, we exhaustively investigate the set of arguably most efficient GCTR variants, with f_X and f_T defined using the simplest operations.

Simple variants. The class of GCTR variants we investigate are what we call "simple and natural". The class is induced by imposing the following restrictions on the functions f_T, f_X:

Simple operations: $f_T(N, R, j)$ (resp. $f_X(N, R, j)$) can only be (1) a concatenation of two out of the three input arguments, or (2) an xor of two out of the three input arguments, or (3) a simple copy of one of the three input arguments, or (4) a constant function independent of the input arguments.

No argument reuse: No input argument can be used by f_T and f_X at the same time (e.g if $f_T = N \oplus R$ then $f_X = N\|\langle j\rangle$ is invalid due to the use of N).

We put no restrictions on the integer parameters $\nu, r, j_{\max}$ defining the domains $\{0,1\}^{\nu}$, $\{0,1\}^{r}$ and $\{1, 2, \ldots, j_{\max}\}$ of respectively the nonce N, the random IV R and the counter j insofar the functions f_X and f_T are well-defined. For example, for $f_X = N \oplus R$ we must have $\nu = r = n$ while for $f_T = N\|j$ we must have $\nu < t$ and $j_{max} = 2^{t-\nu} - 1$. Note that we assume that for an evaluation of the functions f_T and f_X the counter is suitably encoded as a fixed-size binary string $\langle j\rangle$. The restriction to simple operations leaves 10 choices for each f_T and f_X (three possibilities for the xor, three for the concatenation, three for the copy plus the constant function), yielding a set of 100 GCTR variants. Further filtering this set by prohibiting the reuse of arguments leaves 36 variants:

- for a constant f_T, we are free to use any of the nine non-trivial possibilities for f_X (9 variants in total);
- for an f_T that is a copy of one of the three input arguments, f_X can be a binary operation of the remaining arguments or a copy of one of the remaining arguments or a constant function (15 variants in total);
- for an f_T that is a binary operation, f_X can be a copy of the remaining argument or a constant functions (12 variants).

The individual 36 variants are listed in the remaining paragraphs of this section.

Table 1: Trivially insecure GCTR variants.

No.	23	24	25	26	27	28	29	30	31	32	33	34	35	36
f_T	γ	$N\|R$	γ	$N \oplus R$	γ	N	γ	R	R	N	γ	$\langle j\rangle$	$N \oplus \langle j\rangle$	γ
f_X	$N\|R$	γ	$N \oplus R$	γ	N	γ	R	γ	N	R	$\langle j\rangle$	γ	γ	$N \oplus \langle j\rangle$

Trivially insecure variants. As the first step of our investigation, we immediately identify three sets of trivially insecure simple variants:

Counter only: If one of the functions f_T and f_X is a copy of the counter j, and the other is a constant γ, the GCTR variant is trivially insecure, as the key stream blocks it generates repeat in each query. This set consists of variants 33 and 34 in Table 1.

No counter: If none of the functions f_T and f_X uses the counter j, the GCTR variant is trivially insecure, all key stream blocks in a query have the same value. This set consists of variants 23 to 32 in Table 1.

Nonce XORed with counter only: If one of the functions f_T and f_X is $N \oplus \langle j \rangle$, and the other is a constant γ, the GCTR variant is trivially insecure, as the key stream blocks it generates will always have repetitions among queries where the adversary chooses nonces with ensuring that some of the $N \oplus \langle j \rangle$ inputs are the same among these queries. Note that such repetitions of block outputs are unavoidable here even when we restrict the adversary to be nonce-respecting. This set consists of variants 35 and 36 in Table 1.

We refer the reader to the full version of the paper (see Appendix C) for a more formal treatment of nivE attacks on the trivially insecure variants.

Interesting variants. We investigate the 22 variants that remain after the previous filtering. All of them are named and listed in the Table 2 as GCTR-1 to GCTR-22. We give a formal statement of security for these 22 secure GCTR variants and support them with security proofs. The formal claim about the security of these 22 secure GCTR variants is stated in Theorem 1 (see Table 2 for the definition of adversarial resources).

Theorem 1. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$. Then for any adversary $\mathcal{A}$ who makes at most q encryption queries such that the total number of multi-forkcipher calls induced by all the queries is at most $\sigma = \sum_{i=1}^{q} \lceil \frac{\ell_i}{s} \rceil$, we have (here ℓ_i represents the length of i^{th} queried message in terms of n-bit blocks and $\ell = \max_i\{\ell_i\}$)*

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-z}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \mathbf{deg}^{\text{(n)ivE}}_{\text{GCTR-z}[F_s]};\ z \in \{1, 2, \ldots, 22\} \tag{1}$$

for some adversary $\mathcal{B}$ who makes at most σ queries, and runs in time given by the running time of $\mathcal{A}$ plus $\gamma_0 \cdot \sigma$ for some constant γ_0. Here $\mathbf{deg}^{\text{(n)ivE}}_{\text{GCTR-z}[F_s]}$ represents the corresponding degradation in the (n)ivE *security of the variant* GCTR-*z as given in the Table 2 for all values of z.*

Table 2: GCTR$[F_s]$ variants for constructing a nivE scheme using a multiforkcipher. The columns "f_T" and "f_X" respectively show computation of the j^{th} t-bit tweak and n-bit plaintext to the MFC F_s. Here, q and σ are the total number of plaintext queries and MFC calls, respectively, R is an r-bit random value, N is a ν-bit nonce, j is a counter, γ is a constant, $\langle j \rangle$ is a constant-size encoding of an integer j, ℓ is the maximum of query lengths ℓ_is with $1 \leq i \leq q$, and $1 \leq x \leq q$ is the upper bound on number of reuses (repetitions) for any nonce N_i ($x = 1$ means no nonce repeats). The column "ℓ_{max}" contains the maximum number of possible n-bit blocks in a query.

z (No.)	f_T	f_X	ν	r	ℓ_{max}	$\mathbf{deg}^{\text{nivE}}_{\text{GCTR-z}[F_s]}$ $(x = 1)$	$\mathbf{deg}^{\text{ivE}}_{\text{GCTR-z}[F_s]}$ $(x > 1)$
1	$R \Vert \langle j \rangle$	N	n	$< t$	$s2^{t-r}$	$\frac{sq\sigma}{2^{n+r+1}}$	$\frac{qx}{2^{r+1}} + \frac{sq\sigma}{2^{n+r+1}}$
2	N	$R \Vert \langle j \rangle$	t	$< n$	$s2^{n-r}$	$\frac{s\sigma\ell}{2^{n+1}}$	$\frac{qx}{2^{r+1}} + \frac{s\sigma x\ell}{2^{n+1}}$
3	$R \oplus \langle j \rangle$	N	n	t	$s2^t$	$\frac{sq\sigma}{2^{n+t}}$	$\frac{x\sigma}{2^t} + \frac{sq\sigma}{2^{n+t}}$
4	N	$R \oplus \langle j \rangle$	t	n	$s2^n$	$\frac{s\sigma\ell}{2^{n+1}}$	$\frac{x\sigma}{2^n} + \frac{s\sigma x\ell}{2^{n+1}}$
5	$N \Vert R$	$\langle j \rangle$	$< t$	$t - \nu$	$s2^n$	$\frac{s\sigma\ell}{2^n}$	$\frac{qx}{2^{r+1}} + \frac{s\sigma\ell}{2^n};\ x \leq 2^r$
6	$\langle j \rangle$	$N \Vert R$	$< n$	$n - \nu$	$s2^t$	$\frac{sq\sigma}{2^{n+1}}$	$\frac{qx}{2^{r+1}} + \frac{sq\sigma}{2^{n+1}}$
7	$N \Vert \langle j \rangle$	R	$< t$	n	$s2^{t-\nu}$	0	$\frac{(s\sigma+q)x}{2^{n+1}}$
8	R	$N \Vert \langle j \rangle$	$< n$	t	$s2^{n-\nu}$	$\frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$	$\frac{qx}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$
9	$N \oplus R$	$\langle j \rangle$	t	t	$s2^n$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$
10	$\langle j \rangle$	$N \oplus R$	n	n	$s2^t$	$\frac{(s\sigma+q)q}{2^{n+1}}$	$\frac{(s\sigma+q)q}{2^{n+1}}$
11	$R \Vert \langle j \rangle$	γ	0	$< t$	$s2^{t-r}$	$\frac{q^2}{2^{r+1}}$	$\frac{q^2}{2^{r+1}}$
12	γ	$R \Vert \langle j \rangle$	0	$< n$	$s2^{n-r}$	$\frac{q^2}{2^{r+1}} + \frac{s\sigma^2}{2^{n+1}}$	$\frac{q^2}{2^{r+1}} + \frac{s\sigma^2}{2^{n+1}}$
13	$R \oplus \langle j \rangle$	γ	0	t	$s2^t$	$\frac{q\sigma}{2^t}$	$\frac{q\sigma}{2^t}$
14	γ	$R \oplus \langle j \rangle$	0	n	$s2^n$	$\frac{q\sigma}{2^n} + \frac{s\sigma^2}{2^{n+1}}$	$\frac{q\sigma}{2^n} + \frac{s\sigma^2}{2^{n+1}}$

z (No.)	f_T	f_X	ν	r	ℓ_{max}	$\mathbf{deg}^{\text{nivE}}_{\text{GCTR-z}[F_s]}$ $(x = 1)$	$\mathbf{deg}^{\text{ivE}}_{\text{GCTR-z}[F_s]}$ $(x > 1)$
15	R	$\langle j \rangle$	0	t	$s2^n$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$
16	$\langle j \rangle$	R	0	n	$s2^t$	$\frac{(s\sigma+q)q}{2^{n+1}}$	$\frac{(s\sigma+q)q}{2^{n+1}}$
17	$N \oplus \langle j \rangle$	R	t	n	$s2^t$	$\frac{q^2}{2^{n+1}} + \frac{sq\sigma}{2^n}$	$\frac{q^2}{2^{n+1}} + \frac{sq\sigma}{2^n}$
18	R	$N \oplus \langle j \rangle$	n	t	$s2^n$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$	$\frac{q^2}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t$
19	$N\|\langle j \rangle$	γ	$< t$	0	$s2^{t-\nu}$	0	Insecure
20	γ	$N\|\langle j \rangle$	$< n$	0	$s2^{n-\nu}$	$\frac{s\sigma^2}{2^{n+1}}$	Insecure
21	N	$\langle j \rangle$	t	0	$s2^n$	$\frac{s\sigma\ell}{2^{n+1}}$	Insecure
22	$\langle j \rangle$	N	n	0	$s2^t$	$\frac{sq\sigma}{2^{n+1}}$	Insecure

We defer the proof of Theorem 1 to Sect. 6.2.

5 Discussion

In this section, we give an interpretation of the bounds in Theorem 1. We then discuss the performance benefits that can be gained from MFC-based GCTR.

5.1 Security

(Mis)use of Nonce and IV. The GCTR variants 1-18 that use the random IV R input remain secure under nonce reuse. Some of these do not use the nonce N at all, making the nivE and ivE bounds equal (variants 11-16). Most of the variants using both nonce and random IV have a better nivE bound than ivE (here most means all except GCTR variants 9, 10, 17 and 18 because despite of having both N and R as inputs these variants use the nonce N as XORed with either R or the counter which negates the benefits of the nonce).

(Beyond) Birthday-Secure Variants. The classical CTR mode, a.k.a. GCTR-20, is among our 22 secure variants. Interestingly, the bounds of all other 21 variant are superior to that of the CTR mode (variant 20), which becomes void at $\approx 2^{n/2}$ processed blocks. More specifically, all of these (n)ivE bounds are dominated by a quadratic term, but unlike the CTR mode, this term is not only in the number of blocks σ but has q or ℓ as well. (see Table 2). Recall that we (informally) consider a GCTR variant beyond birthday bound (BBB) secure when having a security bound that does not become void around $2^{n/2}$ queried blocks. With this definition, we have 6 variants in the 22 that are BBB-secure namely, variant 1, 3, 7, 11, 13 and 19.

Our pick. The variants GCTR-3 and GCTR-7 are the best two modes in terms of security. Table 2 shows that out of all 22 modes, GCTR-7 achieves the best quantitative security for $x \leq 1 + sq/D$ where $D = ((s\sigma + q)/(2\sigma - q))2^t - 2^n$ whereas for $x > 1 + sq/D$, GCTR-3 provides the best security. In other words, for the nonce respecting case GCTR-7 is the best choice whereas for the general nonce misuse case GCTR-3 with $t = 2n$ is the best choice (in practical cases σ and q are upper bounded by 2^n). The same can also be verified in Fig. 3 (plot F) which shows the security degradation of these variants with increasing number of nonce-repetitions.

We further illustrate the security gap between GCTR-3, GCTR-7 and the classical CTR mode for a fixed input sizes of $n = 128$ in Fig. 3 (plots A to C). For simplicity, q is replaced by its worst-case value, i.e. $q = \sigma$. Further, in Fig. 3 (B, D and E), we give the advantage for more realistic values of q to illustrate that the degradation slope is preserved also for those q values. We note that the advantage slopes for other choices of n have the same shape.

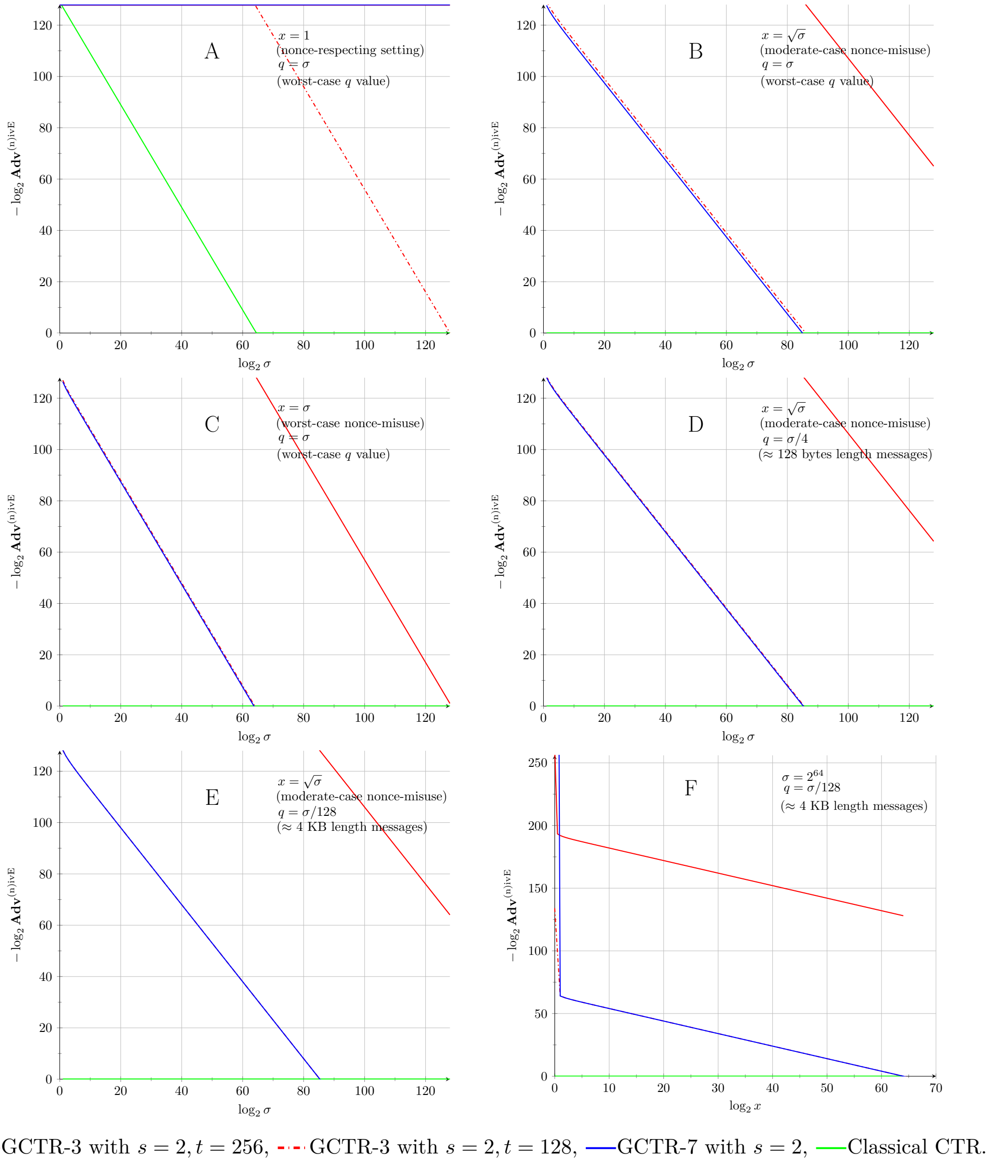


Figure 3: Security comparison (as logarithmic plots of the advantage functions) of GCTR-3 and GCTR-7 using an MFC with $s = 2$ with the classical CTR mode for $n = 128$. We plot the (n)ivE advantage of an adversary as a function of the total number of queried blocks σ (in plots A to E) and as a function of the total number of repeated nonces x (in plot F).

From Fig. 3, we infer for $s = 2$:

1. GCTR-3 with $t = 2n$ provides $\approx n$-bit security against all types of adversaries. GCTR-3 with $t = n$ provides $\approx n$-bit security against nonce-respecting adversaries and BBB-security against nonce misusing (with $x \ll \sigma$ nonce repetitions) adversaries.

2. GCTR-7 provides n-bit security against nonce-respecting adversaries and BBB-security against nonce misusing (with $x \ll \sigma$ nonce repetitions) adversaries.

3. Even though GCTR-3 with $t = n$ and GCTR-7 both provides $\approx n$-bit security for nonce-respecting adversaries, GCTR-7 has a comparatively slower/better security degradation with increasing σs. For example, a nonce-respecting adversary who wants to achieve an advantage of 2^{-120} (or more) requires at least 2^{128} 128-bit encrypted blocks against GCTR-7 whereas to achieve the same advantage for GCTR-3 with $t = n$, it only requires 2^{68} 128-bit encrypted blocks.

GCTR Modes and CTRT. In our GCTR framework CTRT coincides with the variant GCTR-3 while GCTR-4, GCTR-7 and GCTR-17 are just mentioned in [PS16] as other possible secure variants. The existing instantiation Deoxys-II [JNPS16] of CTRT is the same as the GCTR_1-3 (with $t = n$) mode with the TBC Deoxys-BC [JNPS16]. In [PS16], CTRT is shown BBB secure with (n)ivE degradation bound as $\frac{2\sigma(x-1)}{2^t} + \frac{\sigma^2}{2^{n+t+1}}$. In this work, we improve this CTRT security bound with updated degradation as $\frac{(2\sigma-q)(x-1)}{2^{t+1}} + \frac{(q-1)(2\sigma-q)}{2^{n+t+1}}$. This improved bound is of practical relevance and strengthens the security of CTRT in cases where average message length is longer.

Revisiting THCTR. Tweakable HCTR (THCTR) was proposed as a tweakable VIL enciphering scheme that turns an n-bit tweakable block cipher to a variable input length tweakable block cipher [DN18]. It uses a CTR-like encryption mode as an internal building block. In the original publication, THCTR is claimed to be BBB-secure under the TSPRP notion [DN18] and a security proof is provided to support this claim.

However, upon an inspection, the internal CTR-like component of THCTR can be seen to be equal to GCTR-4, for which our own analysis yields BB-security (see Table 2). An investigation of this discrepancy revealed that the claim of THCTR's BBB-security under the TSPRP notion is not correct. We give a BB-attack as Prop. 1 (included in App. D), disproving the claimed TSPRP-security beyond the birthday bound. We also point to the exact flaw in the security proof.

With GCTR-3 and GCTR-7 being better alternatives to GCTR-4 having gracefully degrading BBB-security under the ivE notion, we recommend to replace the GCTR-4-like component of THCTR by GCTR-3 or GCTR-7 to achieve the desired BBB security.

GCTR in AE schemes. There are existing known ways to construct AE schemes from encryption and authentication schemes such as the Encrypt-then-MAC generic composition [BN00] and SCT-style AE [PS16]. We recommend the later for GCTR as it allows message generated pseudo-random IVs and thus reduces the bandwidth as well as avoids the dedicated random sampling of IVs.

We note that the syntax of GCTR is a natural generalization of CTRT only in terms of number of outputs which means that any secure variant of GCTR would yield an AE if combined with an SCT-style overarching scheme.

5.2 Efficiency

A thorough performance evaluation of a GCTR instance would of course require a fixed HW setup and concrete implementations, which is out of scope of this paper. Nevertheless, we do provide an estimation of efficiency gain between GCTR, CTRT and basic CTR by comparing the total number of primitive rounds for instances based on ForkSkinny [PARV19]. Since all GCTR variants follow the same MFC-based GCTR framework, it is sufficient to analyze the efficiency of the generic GCTR mode.

In Fig. 4 we present an efficiency comparison of GCTR mode (instantiated with ForkSkinny, having $s = 2$) with the traditional CTR mode and the CTRT mode (both instantiated with SKINNY [BJK+16]). The estimation shows that the number of rounds required for GCTR[ForkSkinny] is smaller than CTR[SKINNY] and CTRT[SKINNY] for all values of queried bytes. In fact, any GCTR[ForkSkinny] variant with $t = 2n$ is still more efficient than the CTRT[SKINNY] mode with $t = n$.

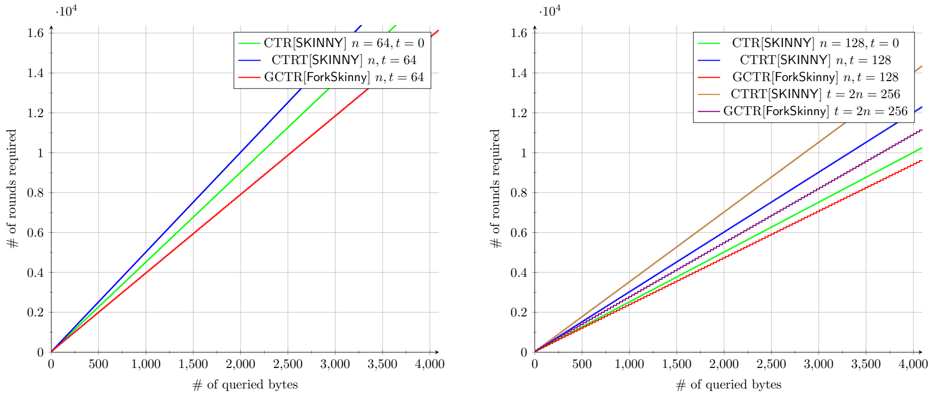


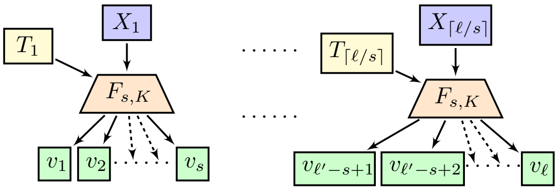
Figure 4: Efficiency comparison of any GCTR[ForkSkinny] (with $s = 2$) with CTR[SKINNY] and CTRT[SKINNY] modes. These plots show the number of rounds required to process the queried bytes ($sn\sigma/8$). The left figure corresponds to the input size $n = 64$ bits whereas the right figure corresponds to the input size $n = 128$ bits.

6 Security

This section is dedicated to security analysis backing up Theorem 1. We first define the Tweakable CTR (TCTR) construction and security notion. Rather than a full-fledged security notion, TCTR is to be seen as intermediate abstraction layer, albeit powerful one. It captures the core security aspects common to all GCTR variants in Lemma 2, thus simplifying the security analysis. We then proceed and prove all the bounds of Theorem 1, relying heavily on the aforementioned lemma.

6.1 Tweakable CTR framework

We define the tweakable CTR construction (TCTR), an algorithm that takes a *sequence* of tweak-input pairs and generates a key stream by applying an MFC to each pair. When paired with a security notion based on the concept of a *sequence-builder* defined below, TCTR is easily seen to be what is common for all GCTR variants. We upper-bound the distinguishing advantage for TCTR as a function of the properties of the used sequence-builder, and then apply this result in the analysis of GCTR variants.

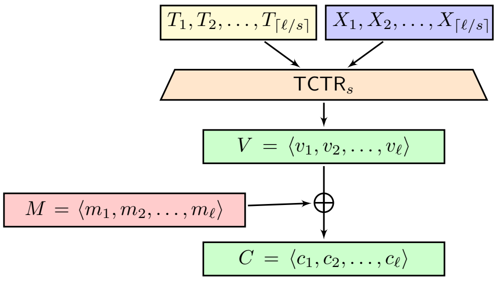


```
Tweakable CTR algorithm : $\mathsf{TCTR}_{s,K}(T, X)$
// $K \in \mathcal{K}$
$(X_1, \ldots, X_{\lceil \ell/s \rceil}) \xleftarrow{n} X$
$(T_1, \ldots, T_{\lceil \ell/s \rceil}) \xleftarrow{t} T$
for $j \leftarrow 1$ to $\lceil \ell/s \rceil$ do
    $(v_{sj-s+1}, \ldots, v_{sj}) \leftarrow F^{T_j}_{s,K}(X_j)$
$V \leftarrow \mathsf{trunc}_{\ell n}(v_1 || \cdots || v_{s\lceil \ell/s \rceil})$
return $V$
```

Figure 5: Tweakable CTR. This algorithm is parameterized by a multi-forkcipher $F_s : \mathcal{K} \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^{sn}$. An input vector $X = \{X_j | 1 \le j \le \lceil \ell/s \rceil\}$ and a tweak vector $T = \{T_j | 1 \le j \le \lceil \ell/s \rceil\}$ must be provided to generate the output vector V having ℓ n-bit blocks. Here ℓ' is defined as $s\lceil \ell/s \rceil$.

Definition of Tweakable CTR. For a fixed multi-forkcipher $F_s : \mathcal{K} \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^{sn}$, the tweakable CTR construction TCTR_s takes in a key $K \in \mathcal{K}$, a sequence of tweaks; $T \in (\{0,1\}^t)^+$ and a sequence of input blocks; $X \in (\{0,1\}^n)^+$. It uses the multi-forkcipher F_s as shown in the Fig. 5 and outputs the key stream vector V as $\langle v_1, v_2 \dots v_\ell \rangle$.

GCTR mode is obtained from the TCTR_s algorithm in a natural way, as shown in Fig. 6. Taking a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$ and plaintext $M \in \{0,1\}^*$, GCTR mode determines the number of components of the input sequence X and the tweak sequence T as $\lceil |M|/sn \rceil$, and computes them using the functions f_X and f_T. It then uses TCTR_s and outputs a ciphertext C.


$T_1, T_2, \dots, T_{\lceil \ell/s \rceil}$
$X_1, X_2, \dots, X_{\lceil \ell/s \rceil}$
TCTR_s
$V = \langle v_1, v_2, \dots, v_\ell \rangle$
$M = \langle m_1, m_2, \dots, m_\ell \rangle$
$C = \langle c_1, c_2, \dots, c_\ell \rangle$

```
GCTR_s Encryption: $\mathcal{E}_K(N, M)$
$R \xleftarrow{\$} \mathcal{R}$
for $j \leftarrow 1$ to $\lceil \ell/s \rceil$ do
    $X_j = f_X(N, R, j)$
    $T_j = f_T(N, R, j)$
$T, X \leftarrow \langle T_1, \dots, T_{\lceil \ell/s \rceil} \rangle, \langle X_1, \dots, X_{\lceil \ell/s \rceil} \rangle$
$V \leftarrow \mathsf{TCTR}_{s,K}(T, X)$
$C \leftarrow M \oplus V$
return $R, C$
```

Figure 6: GCTR$_s$ mode redefined with TCTR. This definition is equivalent with the definition presented in Sect. 4.1.

Security of Tweakable CTR. Defining the security of TCTR as indistinguishability from a random key stream generator, while giving the adversary the ability to directly query the input-and-tweak sequences would not be meaningful, as there are adversaries that would achieve an advantage close to one with constant resources. This also fails to capture how TCTR is used in GCTR.

```
Game tctr-real_{TCTR_s, s-build}
$K \xleftarrow{\$} \mathcal{K}$
$b \leftarrow \mathcal{A}^{\mathcal{O}}$
return $b$

Oracle $\mathcal{O}(N, \ell)$
$R, X, T \xleftarrow{\$} \mathsf{s\text{-}build}(N, \lceil \ell/s \rceil)$
$V \leftarrow \mathsf{trunc}_{\ell \cdot n}(\mathsf{TCTR}_{s,K}(T, X))$
return $R, V$
```

```
Game tctr-ideal_{TCTR_s, s-build}
$b \leftarrow \mathcal{A}^{\mathcal{O}}$
return $b$

Oracle $\mathcal{O}(N, \ell)$
$R, X, T \xleftarrow{\$} \mathsf{s\text{-}build}(N, \lceil \ell/s \rceil)$
$V \xleftarrow{\$} \{0,1\}^{\ell \cdot n}$
return $R, V$
```

Figure 7: Games **tctr-real**$_{\mathsf{TCTR}_s, \mathsf{s\text{-}build}}$ and **tctr-ideal**$_{\mathsf{TCTR}_s, \mathsf{s\text{-}build}}$ defining the security of the TCTR_s construction. Here $\mathsf{s\text{-}build} : \{0,1\}^\nu \times \mathbb{N}^+ \rightarrow \{0,1\}^r \times (\{0,1\}^n)^+ \times (\{0,1\}^t)^+$ is a sequence-builder, a possibly randomized algorithm that maps a nonce to a sequence of MFC inputs and tweaks.

To address the latter, we define the security TCTR by slotting a possibly randomized *query-builder* algorithm $\mathsf{s\text{-}build} : \{0,1\}^\nu \times \mathbb{N}^+ \rightarrow \{0,1\}^r \times (\{0,1\}^n)^+ \times (\{0,1\}^t)^+$ between an adversary and TCTR. The query builder takes as input adversarially chosen nonce N and sequence length ℓ, and outputs random coins R (if used), the sequence of MFC inputs $X \in (\{0,1\}^n)^\ell$ and the sequence of MFC tweaks $T \in (\{0,1\}^t)^\ell$, which are then fed to TCTR to produce the key stream V. The adversary gets R and V. The algorithm $\textsf{s-build}$ is a parameter of the security games **tctr-real** and **tctr-ideal** in Fig. 7, and is fixed throughout the experiment, and known to the adversary. The adversary can thus compute all MFC inputs and tweaks.

An adversary $\mathcal{A}$ who wants to break the $\textsf{TCTR}_s$ algorithm used in conjunction with a sequence builder $\textsf{s-build}$ plays the games **tctr-real** and **tctr-ideal**. $\mathcal{A}$ makes oracle queries of the form N, ℓ as explained above. The oracle returns random coins R (if any) and a string that is either the real $\textsf{TCTR}_s$ output for the inputs queried by $\mathcal{A}$, or a random string of the same length. We define $\mathbf{Adv}^{\text{ind-tctr}}_{\textsf{TCTR}_s,\textsf{s-build}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathbf{tctr\text{-}real}_{\textsf{TCTR}_s,\textsf{s-build}}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{tctr\text{-}ideal}_{\textsf{TCTR}_s,\textsf{s-build}}} \Rightarrow 1]\right|$. We say a $\textsf{TCTR}_s$ construction is secure with $\textsf{s-build}$ if the adversarial advantage $\mathbf{Adv}$ as described above is "small" for all adversaries with "reasonable resources".

The security notion for TCTR is intuitive. For the GCTR variants from Sect. 4, the algorithm $\textsf{s-build}$ consists of simply sampling the random IV R and applying the function f_X and f_T in a loop. In Lemma 2 and the corresponding analysis, we express the security of $\textsf{TCTR}_s$, i.e. the adversarial advantage $\mathbf{Adv}^{\text{ind-tctr}}_{\textsf{TCTR}_s[F_s],\textsf{s-build}}(\mathcal{A})$ as a function of the properties of $\textsf{s-build}$ and of the security of the MFC F_s.

For simplicity, the lemma uses a shorthand $\tilde{\Pr}$ which is defined as follows. Let $E(a)$ be an event that depends on an integer index $a \geq a_0$ where a_0 is a constant. Then $\tilde{\Pr}(E(a)) = \Pr(E(a) \wedge \bar{E}(a-1) \wedge \cdots \wedge \bar{E}(a_0)) \leq \Pr(E(a))$. Further, with this notation, it also holds that $\Pr(E(a) \vee E(a-1) \vee \cdots \vee E(a_0)) = \sum_{i=a_0}^{a} \tilde{\Pr}(E(i))$. This equality holds for any ordering of the indices $[a_0, a]$, however, we stick to the lexicographical ordering. Note that the equality also holds for events that are dependent on multiple indices such as $E(i,j)$. Further, with a slight abuse of notation, we will leave the number q of queries and the length of i^{th} query (in blocks) ℓ_i implicit when summing over all MFC calls, using $\sum_{i,i',j,j'}$ instead of $\sum_{\substack{1\leq i'\leq i\leq q,\ (i',j')<(i,j),\\ 1\leq j'\leq\lceil \ell_{i'}/s\rceil,\ 1\leq j\leq\lceil \ell_i/s\rceil}}$.

Lemma 2 (Security of $\textsf{TCTR}$)**.** *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and $\textsf{s-build}: \{0,1\}^\nu \times \mathbb{N}^+ \to \{0,1\}^r \times (\{0,1\}^n)^+ \times (\{0,1\}^t)^+$ a sequence-builder algorithm. Then for any adversary $\mathcal{A}$ who makes at most q $\textsf{TCTR}_s$ queries of the form (N_i, ℓ_i), such that the i^{th} query is internally mapped to $R_i, ((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s\rceil}, X^i_{\lceil \ell_i/s\rceil}))$ by $\textsf{s-build}$, we have*

$$\mathbf{Adv}^{\text{ind-tctr}}_{\textsf{TCTR}_s[F_s],\textsf{s-build}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \\ \sum_{i,i',j,j'} \left[\frac{s}{2^n}\tilde{\Pr}(X^{i'}_{j'} \neq X^i_j \wedge T^{i'}_{j'} = T^i_j) + \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)\right)\right]$$

for some adversary $\mathcal{B}$ who makes at most $\sigma = \sum_{i=1}^{q} \lceil \frac{\ell_i}{s} \rceil$ queries, and runs in time given by the running time of $\mathcal{A}$ plus $\gamma_0 \cdot \sigma$ for some constant γ_0.

Note that the distribution of the tweak-input pairs, and consequently the bound, is determined by fixing the sequence builder $\textsf{s-build}$.

Proof. [**Lemma 2**] We first replace the multi-forkcipher $F_s(\cdot,\cdot)$ with a tweak-indexed collection of s-tuples of independent random permutations $\pi^i_{T,1}(\cdot), \pi^i_{T,2}(\cdot), \ldots, \pi^i_{T,s}(\cdot) \xleftarrow{\$} \text{Perm}(n)$ for each $T \in \{0,1\}^t$. We let $\textsf{TCTR}_s[\pi]$ denote the $\textsf{TCTR}_s$ that uses these random permutations instead of $F_{s,K}(\cdot,\cdot)$. We have that $\mathbf{Adv}^{\text{ind-tctr}}_{\textsf{TCTR}_s[F_s],\textsf{s-build}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \mathbf{Adv}^{\text{ind-tctr}}_{\textsf{TCTR}_s[\pi]}(\mathcal{A})$. Now, the adversary $\mathcal{A}$ is left with the goal of distinguishing between the games $\mathbf{tctr\text{-}real}_{\textsf{TCTR}_s[\pi]}$ and $\mathbf{tctr\text{-}ideal}_{\textsf{TCTR}_s[\pi]}$. For simplicity, we denote these games by "real world" and "ideal world", respectively. Hence, we want to bound $\mathbf{Adv}^{\text{ind-tctr}}_{\mathsf{TCTR}_s[\pi],\mathsf{s\text{-}build}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathbf{tctr\text{-}real}_{\mathsf{TCTR}_s},\mathsf{s\text{-}build}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{tctr\text{-}ideal}_{\mathsf{TCTR}_s},\mathsf{s\text{-}build}} \Rightarrow 1]\right|$.

Transcripts. Following the coefficients-H technique [Pat09], we describe the interactions of $\mathcal{A}$ with its oracle in a transcript:

$$\tau = \langle((T_1^i, \ldots, T^i_{\lceil \ell_i/s \rceil}), (X_1^i, \ldots, X^i_{\lceil \ell_i/s \rceil}), (v_1^i, \ldots, v^i_{\ell_i}))_{i=1}^q \rangle \,.$$

Coefficient-H. Let us now represent the distribution of the transcripts in the real world and the ideal world by Θ_{re} and Θ_{id}, respectively. The proof relies on the fundamental lemma of the coefficient-H technique as defined above in Lemma 1. We say an *attainable* transcript τ is bad if one of the following conditions occurs:

BadT_1 (input collision) – There exists $(i', j') < (i, j)$ such that $(X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)$.

BadT_2 (output collision) – There exists $(i', j') < (i, j)$ such that $X^{i'}_{j'} \neq X^i_j, T^{i'}_{j'} = T^i_j$ and $v^{i'}_{(j'-1)s+p} = v^i_{(j-1)s+p}$ for at least one of the values of $1 \leq p \leq s$.

We use $\mathcal{T}_{bad}$ to denote the set of "bad" transcripts which is defined as the set of attainable transcripts for which the transcript predicate $\mathsf{BadT}(\tau) = \mathsf{BadT}_1(\tau) \vee \mathsf{BadT}_2(\tau) = 1$. Further, we use $\mathcal{T}_{good}$ to denote the set of attainable transcripts that are not in the set $\mathcal{T}_{bad}$. Transcripts of the set $\mathcal{T}_{good}$ are therefore called good transcripts.

Lemma 3. *For $\mathcal{T}_{bad}$ as defined above, we have*

$$\Pr(\Theta_{id} \in \mathcal{T}_{bad}) \leq \sum_{i,i',j,j'} \left[\frac{s}{2^n} \tilde{\Pr}(X^{i'}_{j'} \neq X^i_j \wedge T^{i'}_{j'} = T^i_j) + \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)\right)\right] .$$

Proof. [**Lemma 3**] For any transcript in $\mathcal{T}_{bad}$ with BadT_1 set to 1, we know that there exists at least one pair of block indices $(i', j') < (i, j)$ for which $(X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)$.

One can notice that there are in total q values of i and i', and for each such i and i', there are $\lceil \ell_i/s \rceil$ and $\lceil \ell_{i'}/s \rceil$ values for j and j', respectively. Now, since the values of Xs and Ts are independent of the corresponding world being real or ideal, we have

$$\Pr(\mathsf{BadT}_1(\Theta_{id}) = 1) = \sum_{i,i',j,j'} \tilde{\Pr}(X^{i'}_{j'} = X^i_j \wedge T^{i'}_{j'} = T^i_j) \,.$$

Similarly, for any transcript in $\mathcal{T}_{bad}$ with BadT_2 set to 1, we know that there exists at least one pair of block indices $(i', j') < (i, j)$ for which $X^{i'}_{j'} \neq X^i_j$, $T^{i'}_{j'} = T^i_j$ and $v^{i'}_{(j'-1)s+p} = v^i_{(j-1)s+p}$ for at least one of the values of $1 \leq p \leq s$.

Clearly, as the values of vs are uniformly and independently distributed in Θ_{id} and as p can take at most s values, the probability of $v^{i'}_{(j'-1)s+p} = v^i_{(j-1)s+p}$ is upper bounded by $s/2^n$. Now, since the values of Xs and Ts are independent of the corresponding world being real or ideal, with same bounds on i, i', j, j' as above, we get

$$\Pr(\mathsf{BadT}_2(\Theta_{id}) = 1) \leq \sum_{i,i',j,j'} \frac{s}{2^n} \cdot \tilde{\Pr}(X^{i'}_{j'} \neq X^i_j \wedge T^{i'}_{j'} = T^i_j)$$

and now using the union bound we obtain the claim of the lemma. □

Lemma 4. *For every good transcript $\tau \in \mathcal{T}_{good}$, $\frac{\Pr(\Theta_{re}=\tau)}{\Pr(\Theta_{id}=\tau)} \geq 1$.*

Proof. [**Lemma 4**] Note that any good transcript τ does not contain input or output collisions as described in the bad events above which means all inputs and outputs blocks (of n-bits) that correspond to the same tweak are distinct in τ. Also, one should keep in mind that the values of input Xs and Ts are not dependent on the corresponding world. We can now compute the probability to obtain a good transcript in the real and ideal worlds as follows. Let $\mathcal{S}_M$ denotes the multiset of all tweaks used during a session of q queries with query lengths $\lceil \ell_1/s \rceil, \ldots, \lceil \ell_q/s \rceil$ in terms of $(n+t)$-bit blocks (here an $n+t$-bit block denotes the corresponding input-tweak pair (T, X)). Let $\mathcal{S}_S$ be the largest subset of $\mathcal{S}_M$ (i.e. a set with all distinct elements of $\mathcal{S}_M$) and let η_a denote the multiplicity of $T_a \in \mathcal{S}_S$ in $\mathcal{S}_M$. In the real world, since the output vs are defined using a random permutation, we get $\Pr(\Theta_{re} = \tau) = \Pi_{a=1}^{|\mathcal{S}_S|} 1/(2^n)_{\eta_a}$. On the other hand, in the ideal world, the output vs are chosen uniformly and independently at random which gives us $\Pr(\Theta_{id} = \tau) = \Pi_{a=1}^{|\mathcal{S}_S|} (1/2^n)^{\eta_a}$. From these two expressions, we get

$$\frac{\Pr(\Theta_{re} = \tau)}{\Pr(\Theta_{id} = \tau)} = \frac{\Pi_{a=1}^{|\mathcal{S}_S|} (2^n)^{\eta_a}}{\Pi_{a=1}^{|\mathcal{S}_S|} (2^n)_{\eta_a}} \geq 1$$

and hence the claim. □

Combining the results of Lemma 3 and 4 (taking $\epsilon = 0$) into Lemma 1, we get the upper bound

$$\mathbf{Adv}^{\text{ind-tctr}}_{\mathsf{TCTR}_s[\pi],\mathsf{s\text{-}build}}(\mathcal{A}) \leq \sum_{i,i',j,j'} \left[\frac{s}{2^n} \tilde{\Pr}(X^{i'}_{j'} \neq X^i_j \wedge T^{i'}_{j'} = T^i_j) + \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j) \right) \right]$$

and hence the result of Lemma 2. □

In the following, we upper bound the probability terms in Lemma 2 for certain choices of s-build.

Lemma 5. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and let s-build(N_i, ℓ_i) output $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ where T^i_j is computed as $N_i \| \langle j \rangle$. Then for any adversary $\mathcal{A}$ who makes at most q TCTR_s queries, such that each nonce value repeats no more than x times and $\sigma = \sum_{i=1}^q \lceil \frac{\ell_i}{s} \rceil$, we have*

$$\sum_{i,i',j,j'} \frac{s}{2^n} \Pr(T^{i'}_{j'} = T^i_j) \leq \frac{s(x-1)\sigma}{2^{n+1}} .$$

The proof of Lemma 5 is straightforward from the fact that there are at most σ choices of $T^i_j = N_i \| \langle j \rangle$ and for each choice (as $\langle j \rangle$ gets fixed) there are at most $x-1$ choices of $T^{i'}_{j'} = N_{i'} \| \langle j' \rangle$ such that $T^i_j = T^{i'}_{j'}$ with non-zero probability. Further, we multiply by an extra $1/2$ as we are only interested in exactly half of these pairs due to the ordering of indices as defined in the sum expression.

Lemma 6. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and let s-build(N_i, ℓ_i) output $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ where T^i_j is computed as $N_i \| R_i$. Then for any adversary $\mathcal{A}$ who makes at most q TCTR_s queries, such that each nonce value repeats no more than x times, $\sigma = \sum_{i=1}^q \lceil \frac{\ell_i}{s} \rceil$ and $\ell = \max_i\{\ell_i\}$, we have*

$$\sum_{i,i',j,j'} \frac{s}{2^n} \Pr(T^{i'}_{j'} = T^i_j) \leq \min\left\{ \frac{s\sigma(\ell-1)}{2^{n+1}} + \frac{s\sigma}{2^{n+r+1}} \cdot \min\{(x-1)\ell, \sigma\}, \frac{s\sigma^2}{2^{n+1}} \right\} .$$

Proof. [**Lemma 6**] Since there are σ possible ways to choose the block index pair (i, j) and for each choice of (i, j), there are at most $(\ell - 1)$ choices for another block index pair $(i', j') \neq (i, j)$ such that $i' = i$, $N_i = N_{i'}$ and $R_i = R_{i'}$ with probability 1, the sum expression which corresponds to these collisions is bounded by the first term as shown above in Lemma 6. Here, we multiply by an extra $1/2$ as we are only interested in exactly half of these pairs due to the ordering of indices as defined in the sum expression.

For all the remaining tweak pairs that are not counted in the above explanation (i.e. tweak pairs with tweaks from different queries), we can have a collision only if the tweak pair corresponds to same N and R. There are σ possible ways to choose the block index pair (i, j) and for each choice of (i, j), there are at most $\min\{(x-1)\ell, \sigma\}$ choices for another block index pair $(i', j') \neq (i, j)$ such that $i' \neq i$, $N_i = N_{i'}$ and $R_i = R_{i'}$ with probability $1/2^r$. Hence, the sum expression which corresponds to these collisions is bounded by the second term as shown above in Lemma 6. Here again, we multiply by an extra $1/2$ as we are only interested in exactly half of these pairs due to the ordering of indices as defined in the sum expression.

The third term can be understood from the fact that $\Pr(T^{i'}_{j'} = T^i_j) \leq 1$ for all $\binom{\sigma}{2}$ pairs of block indices $(i, j), (i', j')$. □

Lemma 7. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and let* s-build(N_i, ℓ_i) *output* $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ *where T^i_j is computed as $R_i \| \langle j \rangle$. Then for any adversary $\mathcal{A}$ who makes at most q TCTR_s queries, such that $\sigma = \sum_{i=1}^{q} \lceil \frac{\ell_i}{s} \rceil$, we have*

$$\sum_{i,i',j,j'} \frac{s}{2^n} \Pr(T^{i'}_{j'} = T^i_j) \leq \frac{s(q-1)\sigma}{2^{n+r+1}} \,.$$

The proof of Lemma 7 is straightforward from the fact that there are at most σ choices of $T^i_j = R_i \| \langle j \rangle$ and for each choice (as $\langle j \rangle$ gets fixed) there are at most $q - 1$ choices of $T^{i'}_{j'} = R_{i'} \| \langle j' \rangle$ such that $T^i_j = T^{i'}_{j'}$ with non-zero probability and since all R_i are uniformly chosen at random, this probability is equal to $1/2^r$. Further, we multiply by an extra $1/2$ as we are only interested in exactly half of these pairs due to the ordering of indices as defined in the sum expression.

Lemma 8. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and let* s-build(N_i, ℓ_i) *output* $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ *where T^i_j is computed as $R_i \oplus \langle j \rangle$ and the distributions of X^i_js and T^i_js are statistically independent over the coins of the adversary and* s-build*. Then for any adversary $\mathcal{A}$ who makes at most q TCTR_s queries, such that $\sigma = \sum_{i=1}^{q} \lceil \frac{\ell_i}{s} \rceil$, we have*

$$\sum_{i,i',j,j'} \frac{s}{2^n} \tilde{\Pr}(X^{i'}_{j'} \neq X^i_j \wedge T^{i'}_{j'} = T^i_j) \leq \sum_{i,i',j,j'} \frac{s}{2^n} \tilde{\Pr}(T^{i'}_{j'} = T^i_j) \leq \frac{s(q-1)(2\sigma - q)}{2^{n+r+1}} \,.$$

Proof. [**Lemma 8**] There are at most σ choices for a block index (i, j) and for each such choice there are at most $q - 1$ choices for another block index $(i', 1)$ with $i' \neq i$ such that $R_i \oplus R_{i'} = \langle j \rangle \oplus \langle 1 \rangle$ with probability $1/2^r$. Clearly, this counts all possible tweak collisions. However, one can notice that we have counted each pair of indices $((i, j), (i', 1))$ twice whenever $j = 1$ due to their ordering. Since we are only interested in unordered pairs of indices, we subtract these extra cases from the counted ones. Let us now count these extra cases. There are at most q choices for a block index (i, j) with $j = 1$ and for each such choice there are at most $q - 1$ choices for another block index $(i', 1) \neq (i, 1)$. Since we are only interested in exactly half of these pairs (i.e. the unordered pairs), we multiply by $1/2$. The final bound on tweak collision probability after subtracting these pairs becomes $(q-1)(\sigma - q/2)/2^r$. □

Lemma 9. *Let F_s be a tweakable multi-forkcipher with tweak space $= \{0,1\}^t$ and let* s-build(N_i, ℓ_i) *output* $((T_1^i, X_1^i), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ *where X_j^i is γ, a constant fixed for all block tuples (i,j). Then for any adversary $\mathcal{A}$ who makes at most q* TCTR_s *queries, we have*

$$\sum_{i,i',j,j'} \frac{s}{2^n} \tilde{\Pr}(X_{j'}^{i'} \neq X_j^i \wedge T_{j'}^{i'} = T_j^i) = 0\,.$$

The proof of Lemma 9 is straightforward from the fact that for $X_j^i = \gamma$, we have $X_{j'}^{i'} = X_j^i$ for all block index pairs $(i,j) > (i',j')$.

6.2 Security of GCTR

Table 2 lists 22 instantiations for the GCTR mode. For each of the 22 variants, the corresponding sequence-builder is defined in the natural way, using the functions f_X and f_T of the given variant. For the variant GCTR$_s$-i, we denote the corresponding sequence-builder $\mathsf{s\text{-}build}_i$. Additionally, we would like to emphasize that the plaintext M which is also fed as an input to the GCTR mode plays no role, and can be w.l.o.g. omitted in the rest of the analysis.

Proof. [**Theorem 1**] We use Lemma 2 and Lemmas 5-9 to prove this theorem. It is easy to see that for any (n)ivE adversary $\mathcal{A}$ against GCTR$_s$-i there exists an ind-tctr adversary $\mathcal{A}'$ against the underlying TCTR_s and $\mathsf{s\text{-}build}_i$ which uses $\mathcal{A}$ as a subroutine, and for which we have

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR}_s\text{-}i[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{ind}-\text{tctr}}_{\mathsf{TCTR}_s[F_s],\mathsf{s\text{-}build}_i}(\mathcal{A}')\,. \tag{2}$$

Let us now define Q as $\{(N_i, R_i) \mid 1 \leq i \leq q\}$; the set of q queries of $\mathcal{A}$ against GCTR with i^{th} query labeled as its corresponding pair (N_i, R_i). We define Q_N as $\{(N_i, R_i) \mid N_i = N$ and $1 \leq i \leq q\}$ i.e. a subset of Q with queries containing the same nonce N. By definition of x, any such subset of Q can have at most x elements. We now define the two possible generic events that are applicable against the defined 22 GCTR constructions namely event $\mathcal{U}$ and $\mathcal{V}$.

Event $\mathcal{U}$ – Let $\mathcal{U}$ be the event when in any $Q_N \subseteq Q$, we get $(N, R_{i_1}) = (N, R_{i_2})$ with $i_2 < i_1$ (i.e. one of the randomly chosen R_{i_1}s matches one of the previously chosen R_{i_2}s having the same nonce). Since for any Q_N we can have at most x such R_is and each one is of size r bits, therefore, following the details as explained in Appendix A.1, we obtain $\Pr(\mathcal{U}) \leq q(x-1)/2^{r+1}$. Note that the r here is also a variable and its value depends upon the underlying GCTR variant.

Event $\mathcal{V}$ – Let $\mathcal{V}$ be the event when in any $Q_N \subseteq Q$, for one of the randomly chosen R_{i_1}, an $R_{i_1} \oplus \langle j_1 \rangle$ matches to one of the previously used/defined $R_{i_2} \oplus \langle j_2 \rangle$s. For any Q_N we can have at most x such R_is and each one is of size r bits, therefore, following the details as explained in Appendix A.2, we obtain $\Pr(\mathcal{V}) \leq (2\sigma - q)(x-1)/2^{r+1}$. Note that the r here is also a variable and its value depends upon the underlying GCTR variant.

Note that the events $\mathcal{U}$ and $\mathcal{V}$ as defined are not dependent on the type of inputs of the GCTR mode. However, the fact that the occurrence of one of these events results into one or more input-tweak pair collisions (hereafter called *trivial* collisions) in the GCTR mode depends upon the type of inputs of GCTR (i.e. X and T). To make it more clear, we define event applicability for the GCTR mode.

Event Applicability – If the GCTR variant has $R \oplus \langle j \rangle$ as one of its inputs (i.e. either X or T) then we say that the applicable event for that variant is $\mathcal{V}$, if the variant has any other combination with R as one of its input then we say that the applicable event is $\mathcal{U}$ and if both inputs of the variant are independent of R then we say that none of the two events are applicable. In Table 3, we classify the 22 GCTR modes according to their event applicability.

Table 3: Classification of GCTR variants according to their event applicability.

Applicable event	$\mathcal{U}$	$\mathcal{V}$	None
GCTR variants	1, 2, 5 to 12, 15, 16, 17 and 18	3, 4, 13 and 14	19 to 22

Case Analysis – We now perform an exhaustive case analysis to proceed with the security proof of the GCTR mode. The motivation for doing this case analysis is to define/branch some simplified advantage expressions from the inequality of Lemma 2 (over the events $\mathcal{U}$ and $\mathcal{V}$) which are valid for disjoint sets/categories of GCTR variants. We can then further simplify one of these advantage expressions for individual variants that belong to the corresponding category.

Note that every (sub)case in the upcoming case analysis is defined with some conditions of event applicability and event occurrence of the events $\mathcal{U}$ and $\mathcal{V}$, and the types of the inputs that are fed to the GCTR mode. This shows that every (sub)case corresponds to the particular set of GCTR variants where its imposed conditions apply. Further, note that the length variables of N and R (ν and r) used in this analysis depend upon the variant itself. N is considered fixed to an empty string for GCTR variants that don't use N as one of the inputs i.e. $\nu = 0$. Similarly, R is considered fixed to an empty string for GCTR variants that don't use R as one of the inputs i.e. $r = 0$. We now define the cases as follows.

Case 1: When the event $\mathcal{U}$ is applicable to the given GCTR variant and –
Case 1.1: If the GCTR variant neither has $N \oplus \langle j \rangle$ nor $N \oplus R$ as one of its inputs (X or T) then there will be repetition of some input-tweak pairs (trivial collisions) only when $\mathcal{U}$ occurs. This is true as for any GCTR variant that belongs to this case we can map any input-tweak collision of (X, T) to a collision of (N, R). In expression, we have

$$\sum_{i,i',j,j'} \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)\right) \leq \ \Pr(\mathcal{U}) \leq \ \frac{q(x-1)}{2^{r+1}}\,, \tag{3}$$

where $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ denote the input-tweak pairs of the corresponding i^{th} query to the underlying TCTR_s construction of that GCTR variant.
Case 1.2: If the GCTR variant has $N \oplus \langle j \rangle$ (respectively $N \oplus R$) as one of its inputs (X or T) then there will be repetition of some input-tweak pairs (trivial collisions) only when the pairs are not from a same query and at least their corresponding values of Rs (respectively $N \oplus R$s) are the same. Since there are in total q queries of $\mathcal{A}$ against GCTR, we have

$$\sum_{i,i',j,j'} \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)\right) \leq \ \frac{q(q-1)}{2^{r+1}}\,, \tag{4}$$

where $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ denote the input-tweak pairs of the corresponding i^{th} query to the underlying TCTR_s construction of that GCTR variant.
Case 2: When the event $\mathcal{V}$ is applicable to the given GCTR variant then there will be a repetition of some input-tweak pairs (trivial collisions) only when $\mathcal{V}$ occurs. This is true

as for any GCTR variant that belongs to this case we can map any input-tweak collision of (X, T) to a collision of $(N, R \oplus \langle j \rangle)$ and vice versa. In expression, we have

$$\sum_{i,i',j,j'} \tilde{\Pr}\left((X^{i'}_{j'}, T^{i'}_{j'}) = (X^i_j, T^i_j)\right) = \Pr(\mathcal{V}) \leq \frac{(2\sigma - q)(x-1)}{2^{r+1}}, \tag{5}$$

where $((T^i_1, X^i_1), \ldots, (T^i_{\lceil \ell_i/s \rceil}, X^i_{\lceil \ell_i/s \rceil}))$ denote the input-tweak pairs of the corresponding i^{th} query to the underlying TCTR_s construction of that GCTR variant.

Case 3: When neither of the events $\mathcal{U}$ and $\mathcal{V}$ is applicable to the given GCTR variant and all MFC calls made during the q queries contain distinct input-tweak pairs (only 4 variants namely GCTR-19, 20, 21 and 22 fall into this category when conditioned under the nonce-respecting setting) then we know that there can not be a trivial collision here. However, there can be repetitions of the tweaks used in these calls which lead us to the Lemma 2 and one of the Lemmas 5-9.

Clearly, these generic cases are not only mutually exclusive but are also exhaustive for all GCTR variants of Table 2. For simplicity, let us summarize here the classification of 22 GCTR variants according to their corresponding case and applicable lemma (if multiple lemmas are applicable, the one with tightest bound is used).

Table 4: Classification of GCTR variants according to their corresponding case and applicable lemma.

	Case 1.1	Case 1.2	Case 2	Case 3
Lemma 5	7.			
Lemma 6	2, 5, 8, 12 and 15.	9 and 18.	4 and 14.	20 and 21.
Lemma 7	1, 6 and 16.	10.		22.
Lemma 8		17.	3.	
Lemma 9	11.		13.	19.

The remaining proof of the Theorem 1 relies on combining the results of Lemma 2 and Lemmas 5-9 with the bounds as defined above in the case analysis for each variant of Table 2. In the remaining proof, we use the following explanation for simplicity. If the underlying GCTR variant uses the tweaks defined as nonce with XOR i.e. in the format of $N_i \oplus \langle j \rangle$ or $N_i \oplus R_i$ (this includes the GCTR variants 9 and 17 from the Table 2) then we know that despite of the fact that N_is are distinct the corresponding values of tweaks $N_i \oplus \langle j \rangle$ (or $N_i \oplus R_i$) can be the same for different values of j (or R_i). Note that such collisions are unavoidable even in the case of a nonce-respecting adversary. However, we can at least argue the following:

- For GCTR variants with tweaks $T^i_j = N_i \oplus R_i$ (GCTR-9), the probability of a tweak collision is independent of the nonce repetition, therefore, the tweak collision probability can be computed using Lemma 6 in a similar manner to GCTR variants with tweaks $T^i_j = R_i$.

- For GCTR variants with tweaks $T^i_j = N_i \oplus \langle j \rangle$ (GCTR-17), the probability of a tweak collision is independent of the nonce repetition. However, what we still know is that for any of the query pairs of GCTR-17, there can only be at most one tweak collision due to some $N_i \oplus \langle j \rangle$ repetition. More specifically, for any index pair (i, i') with $1 \leq i' < i \leq q$, $N_i \oplus N_{i'} = \langle j \rangle \oplus \langle 1 \rangle$ can occur with probability at most 1. This implies that the tweak collision probability for GCTR-17 can be easily computed using Lemma 8 in a similar manner to GCTR variants with tweaks $T^i_j = R_i \oplus \langle j \rangle$ but with $|R_i| = 0$ (note that here $|R_i| = 0$ is equivalent to setting $1/2^r = 1$).

We can now bound the (n)ivE advantage of $\mathcal{A}$ against each one of the 22 GCTR variants as follows:
GCTR-1: In this variant $R_i \| \langle j \rangle$ is used as the tweak and N_i as the input, hence applying the results from Lemma 2, Lemma 7 and Eqn.(3), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-1}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(x-1)}{2^{r+1}} + \frac{s(q-1)\sigma}{2^{n+r+1}} \, .$$

GCTR-2: In this variant, N_i is used as the tweak and $R_i \| \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $|R_i| = 0$ and Eqn.(3) with $|R_i| = r$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-2}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(x-1)}{2^{r+1}} + \frac{s\sigma(x\ell-1)}{2^{n+1}} \, .$$

GCTR-3 (CTRT mode): In this variant, $R_i \oplus \langle j \rangle$ is used as the tweak and N_i as the input, hence applying the results from Lemma 2, Lemma 8 with $|R_i| = t$ and Eqn.(5) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-3}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(2\sigma-q)(x-1)}{2^{t+1}} + \frac{s(q-1)(2\sigma-q)}{2^{n+t+1}} \, .$$

GCTR-4: In this variant, N_i is used as the tweak and $R_i \oplus \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $|R_i| = 0$ and Eqn.(5) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-4}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(2\sigma-q)(x-1)}{2^{n+1}} + \frac{s\sigma(x\ell-1)}{2^{n+1}} \, .$$

GCTR-5: Here $N_i \| R_i$ is used as the tweak and $\langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 and Eqn.(3), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-5}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(x-1)}{2^{r+1}} + \frac{s\sigma(2\ell-1)}{2^{n+1}}; \; x \leq 2^r \, .$$

GCTR-6: In this variant, $\langle j \rangle$ is used as the tweak and $N_i \| R_i$ as the input, hence applying the results from Lemma 2, Lemma 7 with $|R_i| = 0$ and Eqn.(3) with $|R_i| = r$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-6}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(x-1)}{2^{r+1}} + \frac{s(q-1)\sigma}{2^{n+1}} \, .$$

GCTR-7: Here $N_i \| \langle j \rangle$ is used as the tweak and R_i as the input, hence applying the results from Lemma 2, Lemma 5 and Eqn.(3) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-7}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(s\sigma+q)(x-1)}{2^{n+1}} \, .$$

GCTR-8: In this variant, R_i is used as the tweak and $N_i \| \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $(|R_i| = t, |N_i| = 0, x = q)$ and Eqn.(3) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-8}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(x-1)}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}}; \; \sigma \leq 2^t \, .$$

GCTR-9: Here $N_i \oplus R_i$ is used as the tweak and $\langle j \rangle$ as the input which means we can follow GCTR-8 and use Lemma 6 here as explained earlier. Now, applying the results from Lemma 2, Lemma 6 with $(|R_i| = t, |N_i| = 0, x = q)$ and Eqn.(4) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-9}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t\,.$$

GCTR-10: In this variant, $\langle j \rangle$ is used as the tweak and $N_i \oplus R_i$ as the input, hence applying the results from Lemma 2, Lemma 7 with $|R_i| = 0$ and Eqn.(4) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-10}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(s\sigma+q)(q-1)}{2^{n+1}}\,.$$

GCTR-11: In this variant, $R_i \| \langle j \rangle$ is used as the tweak and γ (can be treated here as a nonce that is same for all queries i.e. $x = q$) as the input, hence applying the results from Lemma 2, Lemma 9 and Eqn.(3), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-11}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{r+1}}\,.$$

GCTR-12: In this variant, γ (can be treated here as a nonce that is same for all queries i.e. $x = q$) is used as the tweak and $R_i \| \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $|R_i| = 0$ and Eqn.(3) with $|R_i| = r$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-12}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{r+1}} + \frac{s\sigma^2}{2^{n+1}}\,.$$

GCTR-13: In this variant, $R_i \oplus \langle j \rangle$ is used as the tweak and γ (can be treated here as a nonce that is same for all queries i.e. $x = q$) as the input, hence applying the results from Lemma 2, Lemma 9 and Eqn.(5) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-13}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(2\sigma-q)(q-1)}{2^{t+1}}\,.$$

GCTR-14: In this variant, γ (can be treated here as a nonce that is same for all queries i.e. $x = q$) is used as the tweak and $R_i \oplus \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $|R_i| = 0$ and Eqn.(5) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-14}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(2\sigma-q)(q-1)}{2^{n+1}} + \frac{s\sigma^2}{2^{n+1}}\,.$$

GCTR-15: Here R_i is used as the tweak and $\langle j \rangle$ as the input which is same as the inputs of GCTR-5 but with N_i fixed to $N =$ " "; an empty string for all i i.e. $\nu = 0$ and $x = q$. Hence, applying the results from Lemma 2, Lemma 6 with $|R_i| = t$ and Eqn.(3) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-15}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t\,.$$

GCTR-16: Here $\langle j \rangle$ is used as the tweak and R_i as the input which is same as the inputs of GCTR-6 but with N_i fixed to $N =$ " "; an empty string for all i i.e. $\nu = 0$ and $x = q$. Hence, applying the results from Lemma 2, Lemma 7 with $|R_i| = 0$ and Eqn.(3) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-16}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{(s\sigma + q)(q-1)}{2^{n+1}} .$$

GCTR-17: In this variant, $N_i \oplus \langle j \rangle$ is used as the tweak and R_i as the input which means we can follow GCTR-3 and use Lemma 8 here with $|R_i| = 0$ as explained earlier. Now, applying the results from Lemma 2, Lemma 8 with $|R_i| = 0$ and Eqn.(4) with $|R_i| = n$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-17}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{n+1}} + \frac{s(q-1)(2\sigma - q)}{2^{n+1}} .$$

GCTR-18: Here R_i is used as the tweak and $N_i \oplus \langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $(|R_i| = t, |N_i| = 0, x = q)$ and Eqn.(4) with $|R_i| = t$, we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-18}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{q(q-1)}{2^{t+1}} + \frac{s\sigma\ell}{2^{n+1}};\ \sigma \leq 2^t .$$

GCTR-19: Here $N_i \| \langle j \rangle$ is used as the tweak and γ as the input. Further, the adversary is assumed here to be nonce-respecting ($x = 1$) which allows us to apply the results from Lemma 2, Lemma 9 and Case 3 (as defined in the analysis above), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-19}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) .$$

GCTR-20 (CTR mode): In this variant, γ is used as the tweak and $N_i \| \langle j \rangle$ as the input. Further, the adversary is assumed here to be nonce-respecting which allows us to apply the results from Lemma 2, Lemma 6 with $|R_i| = 0, |N_i| = 0, x = q$ (note that this $x = q$ is specific to the Lemma 6 as the tweak γ here can be considered as a tweak $N \| R$ with $|N| = |R| = 0$ which gives $x = q$. It has nothing to do with the GCTR-20 adversary being nonce-respecting or not) and Case 3 (as defined in the analysis above), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-20}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{s\sigma^2}{2^{n+1}} .$$

GCTR-21: Here N_i is used as the tweak and $\langle j \rangle$ as the input, hence applying the results from Lemma 2, Lemma 6 with $|R_i| = 0$ and Case 3 (as defined in the analysis above), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-21}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{s\sigma(x\ell - 1)}{2^{n+1}} .$$

GCTR-22: In this variant, $\langle j \rangle$ is used as the tweak and N_i as the input, hence applying the results from Lemma 2, Lemma 7 with $|R_i| = 0$ and Case 3 (as defined in the analysis above), we get

$$\mathbf{Adv}^{\text{(n)ivE}}_{\text{GCTR-22}[F_s]}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prtmfp}}_{F_s}(\mathcal{B}) + \frac{s(q-1)\sigma}{2^{n+1}} . \qquad \square$$

7 Conclusion and Open Problems

We presented MFC, a generalization of the forkcipher, and used it to define and analyze a generic counter (GCTR) mode construction for tweakable primitives. Our results show that most variants of GCTR outperform the traditional CTR in terms of security and that use of a random IV can help to mitigate the impact of nonce reuse. Further, we also show that an efficient MFC instance can make any GCTR variant more efficient than the comparable CTRT instance. This work seems to be the first systematic investigation of CTR-style modes, which is surprising given its popularity.

In the security proof, we were able to rigorously analyse 22 GCTR variants. An appropriate choice of the abstraction layer combined with the unusual choice to express a bound as a function of elementary probabilities maximizes the common parts of the analyses. We obtained tight bounds that even improve on the state of the art in the case of CTRT (a.k.a. GCTR-3). We show that two variants stand out in terms of security. Our improvement of CTRT's bound illustrates that an investigation of tightness of these bounds is an interesting open problem. The result on GCTR-4 from the appendix D indicates that our security bounds could be tight and that one can define simple attacks satisfying these bounds. However, a full study of the tightness of these bounds is beyond the limit of this paper and we leave it to the future work.

An MFC is a resourceful primitive that boosts the security, and when supported by an efficient instance, can also improve the performance of applications that span beyond GCTR and AEAD for short messages. We leave to future research the questions of designing more efficient MFC instances, especially with $s > 2$, as well as the identification of novel applications benefiting from MFCs.

Acknowledgments

This work was supported by CyberSecurity Research Flanders with reference number VR20192203. This work was supported in part by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058 and by the Flemish Government through FWO Project G.0835.16 A security Architecture for IoT.

References

[ABD⁺20] Elena Andreeva, Amit Singh Bhati, Arne Deprez, Jowan Pittevils, Arnab Roy, and Damian Vizár. New Results and Insighs on ForkAE. In *NIST LWC workshop*, 2020.

[ABL⁺14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 105–125. Springer, 2014.

[ABV20] Elena Andreeva, Amit Singh Bhati, and Damian Vizár. Nonce-Misuse Security of the SAEF Authenticated Encryption mode. In *Selected Areas in Cryptography*, pages 512–534. Springer, 2020.

[ABV21] Elena Andreeva, Amit Singh Bhati, and Damian Vizár. RUP Security of the SAEF Authenticated Encryption mode. Cryptology ePrint Archive, Report 2021/103, 2021. https://ia.cr/2021/103.

[ALP$^+$19a] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. ForkAE v. *Submission to NIST LwC Standardization Process*, 2019.

[ALP$^+$19b] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: a New Primitive for Authenticated Encryption of Very Short Messages. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 153–182. Springer, 2019.

[BDH$^+$17] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Farfalle: parallel permutation-based cryptography. *IACR Transactions on Symmetric Cryptology (ToSC)*, pages 1–38, 2017. https://eprint.iacr.org/2016/1188.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Annual International Cryptology Conference*, pages 123–153. Springer, 2016.

[BK07] Elaine B Barker and John Michael Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised)*. US Department of Commerce, Technology Administration, NIST, 2007.

[BLMR19] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Transactions on Symmetric Cryptology (ToSC)*, 2019(1):5–45, 2019.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer, 2000.

[BZD$^+$16] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. In *10th USENIX Workshop on Offensive Technologies*, 2016.

[CS14] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.

[DN18] Avijit Dutta and Mridul Nandi. Tweakable HCTR: A BBB Secure Tweakable Enciphering Scheme. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology – INDOCRYPT 2018*, pages 47–69, Cham, 2018. Springer International Publishing.

[FFL12] Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 196–215. Springer, 2012.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

[JNPS16] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Submission to CAESAR : Deoxys v1.41, October 2016. http://competitions.cr.yp.to/round3/deoxysv141.pdf.

[LRW00] Helger Lipmaa, Phillip Rogaway, and David Wagner. Comments to NIST concerning AES modes of operations: CTR-mode encryption. In *National Institute of Standards and Technologies*. Citeseer, 2000.

[LRW02] Moses Liskov, Ronald L Rivest, and David Wagner. Tweakable block ciphers. In *Advances in Cryptology-Crypto 2002, Proceedings*, volume 2442, pages 31–46, 2002.

[MV04] David McGrew and John Viega. The Galois/counter mode of operation (GCM). *submission to NIST Modes of Operation Process*, 20, 2004. https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf.

[PARV19] Antoon Purnal, Elena Andreeva, Arnab Roy, and Damian Vizár. What the Fork: Implementation Aspects of a Forkcipher. In *NIST Lightweight Cryptography Workshop 2019*, 2019.

[Pat09] Jacques Patarin. *The "Coefficients H" Technique*, page 328–345. Springer-Verlag, Berlin, Heidelberg, 2009.

[PS16] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: authenticated encryption modes for tweakable block ciphers. In *Annual International Cryptology Conference*, pages 33–63. Springer, 2016.

[RBB03] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2003.

[RK11] Phillip Rogaway and T. Krovetz. The Software Performance of Authenticated-Encryption Modes. *FSE. LNCS*, 6733:306–327, Springer (2011).

[Rog02] Phillip Rogaway. Authenticated-Encryption with Associated-Data. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 98–107, 2002.

[Rog11] Phillip Rogaway. Evaluation of some blockcipher modes of operation. *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011. https://crossbowerbt.github.io/docs/crypto/rogaway_modes.pdf.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. *Advances in Cryptology-EUROCRYPT 2006*, pages 373–390, 2006.

[VP17] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328. ACM, 2017.

[WHF03] Doug Whiting, Russ Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). *submission to NIST Modes of Operation Process*, 2003. https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/ccm/ccm.pdf.

A Upper bounds on $\Pr(\mathcal{U})$ and $\Pr(\mathcal{V})$

A.1 Event $\mathcal{U}$

Let us consider $Q = \{(N_i, R_i) \mid 1 \leq i \leq q\}$ is the set of q queries of $\mathcal{A}$ against GCTR mode with i^{th} query labeled as its corresponding pair (N_i, R_i). We define Q_N as $\{(N_i, R_i) \mid N_i = N \text{ and } 1 \leq i \leq q\}$ i.e. a subset of Q with queries containing the same nonce N. By definition of x, any such subset of Q can have at most x elements. Now, let us recall that $\mathcal{U}$ is the event when in any $Q_N \subseteq Q$, we get $(N, R_{i_1}) = (N, R_{i_2})$ with $i_2 < i_1$ (i.e. one of the randomly chosen R_{i_1}s matches one of the previously chosen R_{i_2}s having the same nonce).

For any Q_N we know that we can have at most x such R_is and each one is of size r bits. Since each one of these R_is is chosen uniformly and independently at random we have $\Pr(\mathcal{U}) = B/2^r$ where B is the total number of query pairs in Q having same nonce. In other words, B is the total number of (unordered) pairs in Q of the form $((N_{i_1}, R_{i_1}), (N_{i_2}, R_{i_2}))$ with $N_{i_1} = N_{i_2}$.

Claim. *For Q, Q_N and x defined as above, $B \leq (x-1)q/2$.*

Proof. Let $Q_{N_1}, \ldots, Q_{N_d}$ be the mutually exclusive and spanning (exhaustive) sets of queries with nonce inputs as $N_1, \ldots, N_d$ respectively (i.e. $Q = \cup_{c=1}^{d} Q_{N_c}$).
Hence, we have,

$$\Pr(\mathcal{U}) = \sum_{c=1}^{d} \Pr(\mathcal{U}_c) \text{ and } B = \sum_{c=1}^{d} B_c\,, \tag{6}$$

where $\mathcal{U}_c$ is the event $\mathcal{U}$ conditioned on the set Q_{N_c} for queries and B_c is the total number of query pairs in Q_{N_c}. Let $|Q_{N_c}| = \theta_c$ for all c then for x as the maximum number of nonce-repetitions, we have

$$\sum_{c=1}^{d} \theta_c = q,\ B_c = \frac{\theta_c(\theta_c - 1)}{2} \text{ and } \theta_c \leq x \ \ \forall\, c\,. \tag{7}$$

Using Eqn.(7) in (6) gives us

$$B = \sum_{c=1}^{d} \binom{\theta_c}{2} = \frac{1}{2} \sum_{c=1}^{d} \theta_c^2 - \frac{q}{2}\,. \tag{8}$$

We now define the functions f and g of $\theta = (\theta_1, \ldots, \theta_d)$ as

$$f(\theta) = B = \sum_{c=1}^{d} \theta_c^2/2 - q/2 \text{ and } g(\theta) = \sum_{c=1}^{d} \theta_c - q = 0\,.$$

One can easily see that $\nabla_\theta(g) \neq 0$ and therefore for λ as a Lagrange multiplier we can define a Lagrange function $\mathcal{L}$ as

$$\mathcal{L}(\theta, \lambda) = f(\theta) - \lambda g(\theta)\,.$$

Clearly, for any "extrema" of f, we have $\nabla_\theta \mathcal{L} = \ 0$ which gives us

$$(\theta_1, \ldots, \theta_d) = (\lambda, \ldots, \lambda)\,. \tag{9}$$

Further, using the first result of Eqn.(7), we get

$$\theta_1 = \theta_2 = \cdots = \theta_d = q/d$$

and therefore in Eqn.(8), we get the extrema of B as

$$B|_{ext} = \sum_{c=1}^{d} \theta_c^2/2 - q/2 = (q/d - 1)q/2\,.$$

Calculating B for few inputs (other than this extrema) results into comparatively small values which means this only extremum of B is indeed a "maximum". Additionally, we know from the pigeonhole principle that the number of Q_Ns in Q can not be smaller than q/x (i.e. $d \geq q/x$) and therefore,

$$B \leq (x-1)q/2\,.$$

□

A.2 Event $\mathcal{V}$

Let us reconsider $Q = \{(N_i, R_i) \mid 1 \leq i \leq q\}$ as the set of q queries of $\mathcal{A}$ against GCTR mode with i^{th} query labeled as its corresponding pair (N_i, R_i). Let again Q_N be defined as $\{(N_i, R_i) \mid N_i = N \text{ and } 1 \leq i \leq q\}$ i.e. a subset of Q with queries containing same nonce N. By definition of x, any such subset of Q can have at most x elements. Now, let us recall that $\mathcal{V}$ is the event when in any $Q_N \subseteq Q$, for one of the randomly chosen R_{i_1}, an $R_{i_1} \oplus \langle j_1 \rangle$ matches one of the previously used/defined $R_{i_2} \oplus \langle j_2 \rangle$s, where j_1 and j_2 can take any value from the counters used in the i_1^{th} and i_2^{th} query, respectively.

Since $R \oplus \langle j \rangle$ is a cyclic permutation in j, we can define the set A of $R \oplus \langle j \rangle$ pairs where such collisions can occur as $\mathsf{A} = \{(R_{i_1} \oplus \langle j_1 \rangle, R_{i_2} \oplus \langle j_2 \rangle) \mid 1 \leq i_2 < i_1 \leq q,\ 1 \leq j_1 \leq \ell_{i_1}^*,\ 1 \leq j_2 \leq \ell_{i_2}^*,\ N_{i_1} = N_{i_2} \text{ and } \min\{j_1, j_2\} = 1\}$. Here $\ell_i^* = \lceil \ell_i/s \rceil$ represents the number of MFC blocks calls used in the i^{th} query and thus can be treated as the maximum value for the corresponding counter of i^{th} query. Note that any other pair for possible collision of $R \oplus \langle j \rangle$ with $j_2 \geq j_1 > 1$ (w.l.o.g.) can be mapped back to the pair $(R_{i_1} \oplus \langle 1 \rangle, R_{i_2} \oplus \langle j_2 - j_1 + 1 \rangle)$ in the set A.

Now, for any Q_N we know that we can have at most x many of these R_is and each one is of size r bits. Since each one of these R_is is chosen uniformly and independently at random we have $\Pr(\mathcal{V}) = A/2^r$ where A is the size of the set A.

Claim. *For Q, Q_N and x defined as above, $A \leq (x-1)(\sigma - q/2)$.*

Proof. Let $Q_{N_1}, \ldots, Q_{N_d}$ be the mutually exclusive and spanning (exhaustive) sets of queries with nonce inputs as $N_1, \ldots, N_d$ respectively (i.e. $Q = \cup_{c=1}^{d} Q_{N_c}$).
Hence, we have,

$$\Pr(\mathcal{V}) = \sum_{c=1}^{d} \Pr(\mathcal{V}_c) \text{ and } A = \sum_{c=1}^{d} A_c\,, \tag{10}$$

where $\mathcal{V}_c$ is the event $\mathcal{V}$ conditioned on the set Q_{N_c} for queries and A_c is the number of elements in A with $N_{i_1} = N_{i_2} = N_c$. Let $|Q_{N_c}| = \theta_c$ for all c then for x as the maximum number of nonce-repetitions, we have

$$\sum_{c=1}^{d} \theta_c = q \quad \text{and} \quad \theta_c \leq \ x \ \ \forall \ c\,. \tag{11}$$

Further, as $\ell_i^* = \lceil \ell_i / s \rceil$ is the number of MFC calls made during the i^{th} query, we can write

$$\sum_{i=1}^{q} \ell_i^* = \sigma\,. \tag{12}$$

Now, since $R \oplus \langle j \rangle$ is a cyclic permutation, by definition of A_c, we have

$$\begin{aligned} A_c = & \sum_{1 \leq i_1 < i_2 \leq \theta_c} (\ell_{i_1}^* + \ell_{i_2}^* - 1) \\ = & \left\{ \sum_{i=1+\theta_{c-1}^*}^{\theta_c^*} \ell_i^* (\theta_c - 1) \right\} - \binom{\theta_c}{2}, \end{aligned} \tag{13}$$

where $\theta_c^* = \sum_{i=1}^{c} \theta_i$ and $\theta_0^* = 0$. Now, from Eq.(10) and (13), we have

$$\begin{aligned} A = & \sum_{c=1}^{d} \left[\left\{ \sum_{i=1+\theta_{c-1}^*}^{\theta_c^*} \ell_i^* (\theta_c - 1) \right\} - \binom{\theta_c}{2} \right] \\ = & (\theta_1 - 1) \left(\ell_1^* + \ldots + \ell_{\theta_1^*}^* - \frac{\theta_1}{2} \right) + \ldots + (\theta_d - 1) \left(\ell_{1+\theta_{d-1}^*}^* + \ldots + \ell_{\theta_d^*}^* - \frac{\theta_d}{2} \right). \end{aligned}$$

For L_c defined as $\left(\ell_{1+\theta_{c-1}^*}^* + \ldots + \ell_{\theta_c^*}^* - \frac{\theta_c}{2} \right)$, we have

$$A = \ \sum_{c=1}^{d} (\theta_c - 1) L_c \tag{14}$$

$$\text{and} \quad \sum_{c=1}^{d} L_c = \ \sigma - q/2\,. \tag{15}$$

We now define the functions f, g_1 and g_2 of $\theta = (\theta_1, \ldots, \theta_d)$ and $L = (L_1, \ldots, L_d)$ as

$$\begin{aligned} f(\theta, L) = & A = \sum_{c=1}^{d} \theta_c L_c - (\sigma - q/2)\,, \\ g_1(\theta, L) = & \sum_{c=1}^{d} \theta_c - q = 0\,, \\ g_2(\theta, L) = & \sum_{c=1}^{d} L_c - (\sigma - q/2) = 0\,. \end{aligned}$$

One can easily see that $\nabla_{\theta, L}(g_1, g_2, g_1 + g_2) \neq 0$ and therefore for λ_1 and λ_2 as Lagrange multipliers we can define a Lagrange function $\mathcal{L}$ as

$$\mathcal{L}(\theta, L, \lambda_1, \lambda_2) = f(\theta, L) - \lambda_1 g_1(\theta, L) - \lambda_2 g_2(\theta, L)\,.$$

Clearly, for any "extrema" of f, we have $\nabla_{\theta,L}\mathcal{L} = 0$ which gives us

$$(L_1, \ldots, L_d, \theta_1, \ldots, \theta_d) = (\lambda_1, \ldots, \lambda_1, \lambda_2, \ldots, \lambda_2) \,. \tag{16}$$

Further, using the results of Eqn.(11) and (15), we get

$$\begin{aligned} L_1 = L_2 = \cdots = L_d = (\sigma - q/2)/d \,, \\ \theta_1 = \theta_2 = \cdots = \theta_d = q/d \,, \end{aligned}$$

and therefore, in Eqn. (14), we get the extrema of A as

$$\begin{aligned} A|_{ext} &= \sum_{c=1}^{d} \theta_c L_c - (\sigma - q/2) \\ &= \sum_{c=1}^{d} q/d^2(\sigma - q/2) - (\sigma - q/2) \\ &= (q/d - 1)(\sigma - q/2) \,. \end{aligned}$$

Calculating A for few inputs (other than this extrema) results into comparatively small values which means this only extremum of A is indeed a "maximum". Additionally, we know from the pigeonhole principle that the number of Q_Ns in Q can not be smaller than q/x (i.e. $d \geq q/x$) and therefore,

$$A \leq (x - 1)(\sigma - q/2). \qquad \square$$

B Combinatorial explanation for the bounds of Appendix A

B.1 Event $\mathcal{U}$

For Q, Q_N, x and B as defined above in Appendix A.1, we know that there are at most q choices for a query index i and for each such choice there are at most $x - 1$ choices for another query index $i' \neq i$ such that $N_i = N_{i'}$. Now, since we are only interested in exactly half of these pairs (i.e. the unordered pairs), we multiply by 1/2 and get,

$$B \leq (x - 1)q/2 \,.$$

B.2 Event $\mathcal{V}$

For Q, Q_N, x and A as defined above in Appendix A.2, we know that there are at most σ choices for a block index (i, j) and for each such choice there are at most $x - 1$ choices for another block index $(i', 1)$ with $i' \neq i$ such that $N_i = N_{i'}$ and $R_i \oplus R_{i'} = \langle j \rangle \oplus \langle 1 \rangle$ with probability $1/2^r$. Clearly, this counts all possible elements of A. Further note that we have counted each pair of indices $((i, j), (i', 1))$ twice whenever $j = 1$ due to their ordering. Since we are only interested in unordered pairs of indices, we subtract these extra cases from the counted ones. Let us now count these extra cases. There are at most q choices for a block index (i, j) with $j = 1$ and for each such choice there are at most $x - 1$ choices for another block index $(i', 1) \neq (i, 1)$ such that $N_i = N_{i'}$. Since we are only interested in exactly half of these pairs (i.e. the unordered pairs), we multiply by 1/2. The final bound after subtracting these pairs becomes

$$A \leq (x - 1)(\sigma - q/2) \,.$$

C General Attack for the Insecure Variants

Since the security of all GCTR variants relies on the underlying TCTR construction, it suffices to show that the underlying TCTR is insecure for a variant to prove it insecure.

Let us recall that in a GCTR variant, the inputs X and T of the underlying TCTR are functions of N, R and $\langle j \rangle$ as $f_X(N, R, j)$ and $f_T(N, R, j)$. Let us now consider another function $f_{X,T}(N, R, j)$ as $f_{X,T} = f_X \| f_T$. From the security definition of TCTR, we know that TCTR is not secure if the input-tweak pairs of its underlying MFC have repetitions. In other words, a GCTR variant is insecure if any two of the queried MFC calls contain the same values for $f_{X,T}$.

We can now define a TCTR adversary $\mathcal{A}$ who makes 2 queries of size 2 $(n+t)$-bit blocks to this TCTR. In total, $\mathcal{A}$ queries the underlying MFC 4 times with inputs as (in the form of $f_{X,T}$)

$$f_{X,T}(N_1, R_1, j_1), \quad f_{X,T}(N_1, R_1, j_2), \quad f_{X,T}(N_2, R_2, j_1) \text{ and } f_{X,T}(N_2, R_2, j_2)$$

with ensuring that $N_1 \oplus N_2 = \langle j_1 \rangle \oplus \langle j_2 \rangle$. Clearly, if any two of these 4 outputs are the same for a variant then we know that $\mathcal{A}$ can easily distinguish the final outputs from random bits and hence the variant is insecure. One can now easily verify for all of the variants which are either mentioned as trivially insecure (all 14 variants of Table 1) or are labeled insecure for a particular setting in Table 2 (i.e. the nonce-reuse setting and thus for these variants $\mathcal{A}$ intentionally uses $N_1 = N_2$ to perform the attack), that they contain two same values in the first 3 entries of these 4 $f_{X,T}$s. To exemplify, let us consider the case when $X = R$ and $T = N$ (variant 32 in Table 1) then we have

$$f_{X,T}(N_1, R_1, j_1) = R_1 \| N_1 = f_{X,T}(N_1, R_1, j_2)\,.$$

D A BB attack on Tweakable HCTR

In this section, we propose a BB-attack (in n) as Prop. 1 that shows that THCTR construction can not achieve TSPRP-security beyond birthday in its existing form. Since our attack specifically targets the underlying CTR-like structure of THCTR and the other primitives of THCTR like the underlying hash functions don't play any role in it, we slightly abuse the notation and leave these other primitives and their input arguments implicit in Prop. 1 (and its proof). We refer the reader to [DN18] for the full definition of THCTR.

Proposition 1. *Let* Π *be the tweakable HCTR VIL enciphering scheme (as defined in [DN18]) that takes a plaintext* M *of size* m*, a tweak* T *of size* t *and a TBC key* K *of size* k *along with the other inputs and returns a ciphertext* C *of size* m *as output. Let* $\tilde{E} : \{0,1\}^k \times \{0,1\}^{t'} \times \{0,1\}^n \to \{0,1\}^n$ *be the underlying* n*-bit TBC (with* k*-bit key and* t'*-bit tweak) of tweakable HCTR. Then, there exists an explicit algorithm* $\mathcal{A}$ *that makes* $q = 1$ *query to the tweakable HCTR with* $\sigma \leq 2^{n/2-1}$ *queries to the underlying TBC* $\tilde{E}$*, so that*

$$\mathbf{Adv}_{\Pi}^{\mathsf{TSPRP}}(\mathcal{A}) \geq \frac{\sigma(\sigma-3)}{2^{n+2}}\,.$$

In other words, $\mathcal{A}$ *can distinguish THCTR from a true tweakable random permutation within* $\mathcal{O}(2^{n/2})$ *TBC queries with reasonably large probability.*

Proof. Let us consider the following algorithm $\mathcal{A}$ that makes a single THCTR query with plaintext input $M = 0^m$ of size m such that m is a multiple of n (i.e. $m = \sigma n$) and any arbitrary but valid tweak T. Later, when provided with the query response C, $\mathcal{A}$ defines $C_0, C_1, \ldots, C_{\sigma-1} \xleftarrow{n} C$ and returns 1 if there exists $i \neq 0, j \neq 0$ with $i \neq j$ such that $C_i = C_j$ and 0, otherwise.

Now, since we know that an output of a random permutation is information-theoretically indistinguishable from an output of a random function when only one forward query is allowed to these oracles, we have for $q = 1$,

$$\mathbf{Adv}_{\Pi}^{\mathsf{TSPRP}}(\mathcal{A}) \geq \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\Pi_K(T,M)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Rand}^{\$}(T,M)} \Rightarrow 1\right]\right| .$$

One can verify that by definition of THCTR, $\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\Pi_K(T,M)} \Rightarrow 1]$ is always 0 as in a single query all except the first underlying TBC block calls are fed with same tweak but distinct inputs which implies all output blocks C_is with $i \neq 0$ are always distinct. On the other hand, we can show for $\mathsf{Rand}^{\$}(T, M)$ (see Sec. 2.1 for definition) with basic probability calculations that

$$\begin{aligned}\Pr[\mathcal{A}^{\mathsf{Rand}^{\$}(T,M)} \Rightarrow 1] &= 0 + \frac{1}{2^n} + \frac{2}{2^n}\left(1 - \frac{1}{2^n}\right) + \cdots + \frac{\sigma-2}{2^n}\prod_{j=1}^{\sigma-3}\left(1 - \frac{j}{2^n}\right) \\ &= \sum_{i=1}^{\sigma-1}\frac{i-1}{2^n}\prod_{j=1}^{i-2}\left(1 - \frac{j}{2^n}\right) .\end{aligned}$$

This implies

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathsf{TSPRP}}(\mathcal{A}) &\geq \sum_{i=1}^{\sigma-1}\frac{i-1}{2^n}\prod_{j=1}^{i-2}\left(1 - \frac{j}{2^n}\right) \\ &\geq \sum_{i=3}^{\sigma-1}\frac{i-1}{2^n}\left(1 - \frac{(i-2)}{2^n}\right)^{i-2} + \frac{1}{2^n} .\end{aligned} \tag{17}$$

Now, for all i with $3 \leq i \leq \sigma \leq 2^{n/2-1}$, one can show with basic algebra that

$$\frac{(i-2)}{2^n} + \left(\frac{1}{2}\right)^{1/(i-2)} \leq \left(\frac{1}{2}\right)^{n/2+1} + \left(\frac{1}{2}\right)^{2^{(-n/2+1)}} \leq 1 .$$

Thus, we get

$$\begin{aligned}\mathbf{Adv}_{\Pi}^{\mathsf{TSPRP}}(\mathcal{A}) &\geq \sum_{i=3}^{\sigma-1}\frac{i-1}{2^n}\left(\left(\frac{1}{2}\right)^{1/(i-2)}\right)^{i-2} + \frac{1}{2^n} \\ &\geq \frac{\sigma(\sigma-3)}{2^{n+2}} .\end{aligned} \tag{18}$$

Further, it is clear from Eqn. 17 that Eqn. 18 is also valid for $\sigma = 1$ and 2 and hence, we get the claim of Proposition 1. □

We note that the exact flaw in the security proof of THCTR lies in the bad case analysis of case B.2 ($\sigma(\mu - 1)$ should be $\sigma\ell(\mu - 1)$) and case B.3 ($\sigma(\mu - 1)$ should be $\sigma(\mu\ell - 1)$). We refer the reader to our security analysis of GCTR for more details on these cases.

We would like to again emphasize that despite the fact that our security analysis targets a completely different security notion called nivE notion than the targeted TSPRP [DN18] notion of THCTR, our attack and case analysis holds for settings with $q = 1$ as shown above in the proof of Prop. 1.