Elastic-Tweak: A Framework for Short Tweak Tweakable Block Cipher

Avik Chakraborti[1], Nilanjan Datta[2], Ashwin Jha[3], Cuauhtemoc Mancillas Lopez[4], Mridul Nandi[5], Yu Sasaki[6]

[1] University of Exeter, UK
[2] Institute for Advancing Intelligence, TCG CREST, Kolkata, India
[3] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[4] Computer Science Department, CINVESTAV-IPN, Mexico
[5] Indian Statistical Institute, Kolkata, India
[6] NTT Secure Platform Laboratories, Japan

avikchkrbrti@gmail.com, nilanjan.datta@tcgcrest.org, ashwin.jha@cispa.de, cuauhtemoc.mancillas83@gmail.com, mridul.nandi@gmail.com, sasaki.yu@lab.ntt.co.jp

Abstract. Tweakable block cipher (TBC), a stronger notion than standard block ciphers, has wide-scale applications in symmetric-key schemes. At a high level, it provides flexibility in design and (possibly) better security bounds. In multi-keyed applications, a TBC with short tweak values can be used to replace multiple keys. However, the existing TBC construction frameworks, including TWEAKEY and XEX, are designed for general purpose tweak sizes. Specifically, they are not optimized for short tweaks, which might render them inefficient for certain resource constrained applications. So a dedicated paradigm to construct short-tweak TBCs (tBC) is highly desirable. In this paper, as a first contribution, we present a dedicated framework, called the Elastic-Tweak framework (ET in short), to convert any reasonably secure SPN block cipher into a secure tBC. We apply the ET framework on GIFT and AES to construct efficient tBCs, named TweGIFT and TweAES. These short-tweak TBCs have already been employed in recent NIST lightweight competition candidates, LOTUS-LOCUS and ESTATE. As our second contribution, we show some concrete applications of ET-based tBCs, which are better than their block cipher counterparts in terms of key size, state size, number of block cipher calls, and short message processing. Some notable applications include, Twe-FCBC (reduces the key size of FCBC and gives better security than CMAC), Twe-LightMAC_Plus (better rate than LightMAC_Plus), Twe-CLOC, and Twe-SILC (reduces the number of block cipher calls and simplifies the design of CLOC and SILC).

Keywords: TBC, GIFT, AES, TWEAKEY, XEX, ESTATE, LOTUS-LOCUS

1 Introduction

Since their advent in late 1970's, block ciphers [1,2] have become the ubiquitous building blocks in various symmetric-key cryptographic algorithms, including

encryption schemes [3], message authentication codes (MACs) [4], and authenticated encryption [5]. Due to their wide-scale applicability, block ciphers are also the most well-analyzed symmetric-key primitives. As a result, the cryptographic community bestows a high degree of confidence in block cipher based designs. Block cipher structures are more or less well formalized and there are formal ways to prove the security of a block cipher against the classical linear [6] and differential [7] attacks. The literature is filled with a plethora of block cipher candidates, AES [2] being the most notable among them. AES is currently the NIST standard block cipher [2], and it is the recommended choice for several standardized encryption, MAC and AE schemes such as CTR [3], CMAC [4], AES-GCM [8] etc. A recent block cipher proposal, named GIFT [9] has generated a lot of interest due to its ultra-lightweight nature.

1.1 Some Issues in Block Cipher Based Designs

Key Size of Designs: Several designs use more than one independent block cipher keys, which could be an issue for storage constrained applications. Some notable examples of such designs are sum of permutations [10, 11], EDM [12], EWCDM [12], CLRW2 [13], GCM-SIV-2 [14], Benes construction [15]. While some of these designs have been reduced to single key variants, reducing a multi-keyed design to single-key design is, in general, a challenging problem.

Auxiliary Secret State: FCBC, a three-key MAC by Black and Rogaway [16], is a CBC-MAC type construction. CMAC [4], the NIST recommended MAC design, reduces number of keys from three to one by using an auxiliary secret state (which is nothing but the encryption of zero block). Though CMAC is NIST recommended MAC design, it costs an extra block cipher call (compared to FCBC) and holds an additional state. This may be an issue in hardware applications, where area and energy consumption are very crucial parameters. Further FCBC [17, 18] allows more number of queries per key, as compared to CMAC [19].

Simplicity of Designs: Design simplification, is a closely related topic to the single-keyed vs. multi-keyed debate. A simple design could be beneficial for real life applications, and better understanding of designs themselves. Often, the single-keyed variant of a block cipher based design is much more complex than the multi-keyed version, both in implementation and security analysis. This is due to the several auxiliary functions used chiefly for domain separation. For instance CLOC and SILC [20] use several functions depending upon the associated data and message length. In contrast, the multi-keyed variants of CLOC and SILC would be much simpler.

Short Message Processing: An essential requirement in lightweight applications is efficient short input data processing, while minimizing the memory consumption and precomputation. In use cases with tight requirements on delay and latency, the typical packet sizes are small (way less than 1 Kilobytes) as large packets occupy a link for longer duration, causing more delays to subsequent packets and increasing latency. For example, Zigbee, Bluetooth low energy

and TinySec [21] limit the maximum packet lengths to 127 bytes, 47 bytes and 128 bytes, respectively. Similarly, CAN FD [22], a well-known transmission protocol in automotive networks, allows message length up to 64 bytes. The packet sizes in EPC tag [23], which is an alternate to the bar code using RFID, is typically 12 bytes.

Cryptographic designs with low latency for shorter messages could be highly beneficial for such applications. As it turns out, for many designs short message performance is not that good due to some constant overhead. For instance CMAC uses one block cipher call to generate a secret state, and SUNDAE [24] uses the first call of block cipher to distinguish different possibilities of associated data and message lengths. So, to process a single block message, SUNDAE requires two block cipher calls. CLOC and SILC [20] have similar drawbacks. They cost 2 and 4 calls to process a single block message. LightMAC_Plus [25], feeds a counter-based encoded input to the block cipher, which reduces the rate.[7]

1.2 Motivation of short-tweak TBC

Tweakable Block Ciphers: The Hasty Pudding cipher [26], an unsuccessful candidate for AES competition, was one of the first tweakable block ciphers.[8] Later, Liskov et al. in formalized this in their foundational work on tweakable block ciphers [27]. Tweakable block ciphers (TBCs) are more versatile and find a broad range of applications, most notably in authenticated encryption schemes, such as OCB [28], COPA [29], and Deoxys [30]; and message authentication codes, such as ZMAC [31], NaT [32], and ZMAC+ [33]. TBCs can be designed from scratch [26, 34, 35], or they can be built using existing primitives like block ciphers, and public permutations. LRW1, LRW2 [27], CLRW2 [13], XEX [36] and XHX [37] are some examples of the former category, whereas Tweakable Even-Mansour [32] is an example of the latter.

Tweakable block cipher can actually solve most of the aforementioned issues in block ciphers quite easily. A secure TBC with distinct tweaks is actually equivalent to independently keyed instantiations of a secure block cipher. This naturally gives a TBC based single-keyed design for any block cipher based multi-keyed design. For example, one can use this equivalence to define a single-keyed version of FCBC which is as secure as FCBC. This resolves the issues with CMAC. In some cases, TBCs can also avoid the extra block cipher calls. It also helps to simplify designs like CLOC and SILC.

In all these cases, we observe that a short tweak space (in most of the cases 2-bit or 4-bit tweaks) is sufficient. In other words, a short-tweak tweakable block cipher (in short we call tBC) would suffice for resolving these issues. An tBC is better than large tweak TBCs in two respects: (i) state size for holding tweak is small, and most importantly (ii) tBC would potentially be more efficient than large tweak TBCs.

[7] No. of message blocks processed per block cipher call.
[8] It used the term "spice" for tweaks.

THE TWEAKEY FRAMEWORK: At Asiacrypt '14, Jean et al. presented a generic framework for TBC construction, called TWEAKEY [38], that considers the tweak and key inputs in a unified manner. Basically, the framework formalized the concept of tweak-dependent keys. The TWEAKEY framework gave a much needed impetus to the design of TBCs, with several designs like Kiasu [39], Deoxys [30], SKINNY and Mantis [40] etc. As TWEAKEY is conceptualized with general purpose tweak sizes in mind, it is bit difficult to optimize TWEAKEY for tBC. For instance, take the example of SKINNY-128. To process only 4-bit tweak, the additional register is limited but their computation modes must move from TK1 to TK2, which increases the number of rounds by 8. This in turn affects the throughput of the cipher. Although, some TWEAKEY-based designs, especially Kiasu-BC [39] do not need additional rounds, yet this is true in most of the existing TWEAKEY-based designs. We also note here that Kiasu-BC, which is based on AES, is weaker than AES by one round, as observed in several previous cryptanalytic works [41–43].

So, there is a need for a generic design framework for tBC, which (i) can be applied on top of a block cipher, (ii) adds minimal overheads, and (iii) is as secure as the underlying block cipher.

XE AND XEX: Rogaway [36], proposed two efficient ways of converting a block cipher into a tweakable block cipher, denoted by XE and XEX. These methods are widely used in various modes such as PMAC [44], OCB [45], COPA [29], ELmD [46] etc. However, XE and XEX have several limitations with respect to a short tweak space, notably (i) security is limited to birthday bound, and (ii) precomputation and storage overhead to generate the secret state. In addition, it also requires to update the secret state for each invocation, which might add some overhead.

1.3 Our Contributions

Our main contributions can be divided into two parts:

1. ELASTIC-TWEAK FRAMEWORK: In this work, we address the above issues and propose a generic framework, called the Elastic-Tweak framework (ET in short), to transform a block cipher into a short tweak TBC. We consider "short tweaks" of size less than equal to 16 bits and greater than equal to 4 bits. This small size ensures that the tweak storage overhead is negligible. In this framework, given the block cipher, we first expand the short tweak using linear code, and then inject the expanded tweak at intervals of some fixed number of rounds, say r. Designs under this framework can be flexibly built over a secure block cipher, and are as secure as the underlying block cipher.

The ET framework distributes the effect of the tweak into the block cipher state that can generate several active bytes. In particular we choose a linear code with high branch number to expand the input tweak. This design is particularly suitable for short tweaks to ensure the security against differential cryptanalysis because the small weight of the short input always results in a large weight of the output.

Another advantage of the framework is the easiness of the security evaluation. First, for zero tweak value, the plaintext-ciphertext transformation is exactly the same as the original cipher (i.e. it has backward compatibility feature). Therefore, to evaluate the security of the new construction, we only need to consider the attacks that exploit at least one non-zero tweak. Second, the large weight of the expanded tweak ensures relatively high security only with a small number of rounds around the tweak injection. This allows a designer to focus on the security of the r-round transformation followed by the tweak injection and further followed by the r-round transformation, which is called "$2r$-round core."

We instantiate this framework with several designs over two well known block ciphers AES [2] and GIFT [9] with different tweak sizes varying from 4 to 16. Several of these candidates have already been extensively analyzed in [47, 48] in terms of security and performance due to their use in NIST lightweight competition candidates, LOTUS-LOCUS [49] and ESTATE [50]. However, we refer the full version [51] for the thorough security analysis (Sect 4, [51]) and performance evaluation (Sect 3.4 and Appendix C, [51]).

2. Applications of tBC: Here we demonstrate the applicability of tBC in various constructions:

1. **Reducing the Key Size in Multi-Keyed Modes:** The primary application of tBC is to reduce the key space of several block cipher based modes that use multiple independently sampled keys. We depict the applicability of tBC on FCBC MAC, Double Block Hash-then-Sum (DbHtS) paradigm, Sum of permutations, EDM, EWCDM, CLRW2, GCM-SIV-2 and the Benes construction.
2. **Efficient Processing of Short Messages:** tBC can be used to reduce the number of block cipher calls, which in turn reduces the energy consumption for short messages. We take the instance of Twe-LightMAC_Plus to demonstrate this application of tBC. Twe-LightMAC_Plus achieves a higher rate as compared to it's original counterpart LightMAC_Plus. In addition, the number of keys is reduced from 3 to 1. However, this is also applicable to Twe-CLOC and Twe-SILC (tBC based counterparts of CLOC and SILC [20] respectively).
3. **Replacement for XE and XEX.** tBC can be viewed as an efficient replacement of XE and XEX especially when we target short messages (say of size up to 1 MB). In such cases, instead of using a secret state (that we need to precompute, store and update), one can simply use tBC with the block-counters as the tweak. The applicability of this paradigm can be depicted on several MAC modes such as PMAC; encryption mode such as COPE and AEAD modes such as ELmD, COLM.

In addition to the above applications, we show that tBCs can also simplify the internal structures of various block cipher based authenticated encryption modes. For example, CLOC, SILC use several auxiliary functions mainly for domain separation. We propose tBC-based variants for these, named Twe-CLOC and Twe-SILC, which simplify the original designs (by cleaning up the auxiliary functions) and reduces the number of block cipher calls. These in turn help in

reducing the area of hardware implementation, and significantly increasing the throughput for short messages.

2 Preliminaries

NOTATIONS: For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$, and $\{0,1\}^n$ denotes the set of all n-bit binary strings. We use $\{0,1\}^+$ to denote the set of all non-empty binary strings. $\perp$ denotes the empty string and $\{0,1\}^* = \{0,1\}^+ \cup \{\perp\}$. For any string $X \in \{0,1\}^n$, $|X|$ denotes the number of bits in X, and for $i \in [|X|]$, x_i denotes the i-th significant bit ($x_{|X|}$ being the most significant bit). For $X \in \{0,1\}^+$ and $n \in \mathbb{N}$, $(X)_{[\ell]} := (X_1, \ldots, X_\ell) \xleftarrow{n} X$, denotes the n-bit block parsing of X into $(X)_{[\ell]}$, where $|X_i| = n$ for $[\ell - 1]$, and $X_\ell \in [n]$. For $k \leq n \in \mathbb{N}$, and $X \in \{0,1\}^n$, $\lfloor X \rfloor_k := X_1 \ldots X_k$. The expression $a \ ? \ b \ : \ c$ evaluates to b if a is true and c otherwise.

For $n, m \in \mathbb{N}$, $\mathsf{Perm}(n)$ denotes the set of all permutations over $\{0,1\}^n$, and $\mathsf{Func}(m,n)$ denotes the set of all functions from $\{0,1\}^m$ to $\{0,1\}^n$. For $n, \kappa \in \mathbb{N}$, $\mathsf{TPerm}(\kappa, n)$ denotes the set of all families of permutations $P_k := P(k, \cdot) \in \mathsf{Perm}(n)$ indexed by $k \in \{0,1\}^\kappa$. By extending notation, we use $\mathsf{TPerm}(\kappa, \tau, n)$ to denote the set of all families of permutations $P_{k,t} \in \mathsf{Perm}(n)$, indexed by $(k, \tau) \in \{0,1\}^\kappa \times \{0,1\}^\tau$.

(TWEAKABLE) BLOCK CIPHER: A block cipher with key size κ and block size n is a family of permutations $\mathsf{E} \in \mathsf{TPerm}(\kappa, n)$. For a fixed key $k \in \{0,1\}^\kappa$, we write $\mathsf{E}_k(\cdot) = \mathsf{E}(k, \cdot)$, and its inverse is written as $\mathsf{E}_k^{-1}(\cdot)$. A tweakable block cipher with key size κ, tweak size τ, and block size n is a family of permutations $\mathsf{E} \in \mathsf{TPerm}(\kappa, \tau, n)$. For a fixed key $k \in \{0,1\}^\kappa$ and tweak $t \in \{0,1\}^\tau$, we write $\mathsf{E}_k^t(\cdot) = \mathsf{E}(k, t, \cdot)$, and its inverse is written as $\mathsf{E}_k^{-t}(\cdot)$. Throughout this paper we fix $\kappa, \tau, n \in \mathbb{N}$ as the key size, tweak size, and block size, respectively, of the given (tweakable) block cipher.

2.1 Security Definitions

(TWEAKABLE) RANDOM PERMUTATION AND RANDOM FUNCTION: For any finite set $\mathcal{X}$, $\mathsf{X} \leftarrow\!\!\$\ \mathcal{X}$ denotes uniform and random sampling of X from $\mathcal{X}$.

We call $\Pi \leftarrow\!\!\$\ \mathsf{Perm}(n)$ a (uniform) random permutation, and $\widetilde{\Pi} \leftarrow\!\!\$\ \mathsf{TPerm}(\tau, n)$ a tweakable (uniform) random permutation on tweak space $\{0,1\}^\tau$ and block space $\{0,1\}^n$. Note that, $\widetilde{\Pi}^i$ is independent of $\widetilde{\Pi}^j$ for all $i \neq j \in \{0,1\}^\tau$. We call $\Gamma \leftarrow\!\!\$\ \mathsf{Func}(m,n)$ a (uniform) random function from $\{0,1\}^m$ to $\{0,1\}^n$.

We say that a distinguisher is "sane" if it does not make duplicate queries, or queries whose answer is derivable from previous query responses. Let $\mathbb{A}(q,t)$ denote the class of all sane distinguishers, limited to at most q queries and t computations.

TWEAKABLE STRONG PSEUDORANDOM PERMUTATION (TSPRP): The TSPRP advantage of any distinguisher $\mathcal{A}$ against $\widetilde{\mathsf{E}}$ instantiated with key $\mathsf{K} \leftarrow\!\!\$\ \{0,1\}^\kappa$,

is defined as

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\mathcal{A}) := \left|\Pr[\mathcal{A}^{\widetilde{\mathsf{E}}^{\pm}_{\mathsf{K}}} = 1] - \Pr[\mathcal{A}^{\widetilde{\Pi}^{\pm}} = 1]\right|.$$

The TSPRP security of $\widetilde{\mathsf{E}}$, is defined as

$$\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(q,t) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{\mathsf{E}}}(\mathcal{A}). \tag{1}$$

TPRP or tweakable pseudorandom permutation and its advantage $\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(q,t)$ is defined similarly when adversary has no access of the inverse oracle.

Pseudorandom Function (PRF): The PRF advantage of distinguisher $\mathcal{A}$ against a keyed family of functions $\mathsf{F} := \{\mathsf{F}_K : \{0,1\}^m \to \{0,1\}^n\}_{K\in\{0,1\}^\kappa}$ is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}) := \left|\Pr_{\mathsf{K}\leftarrow\$\,\{0,1\}^\kappa}[\mathcal{A}^{\mathsf{F}_\mathsf{K}} = 1] - \Pr[\mathcal{A}^{\Gamma} = 1]\right|.$$

The PRF security of F against $\mathbb{A}(q,t)$ is defined as

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q,t) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(\mathcal{A}). \tag{2}$$

The keyed family of functions F is called weak PRF family, if the PRF security holds when the adversary only gets to see the output of the oracle on uniform random inputs. This is clearly a weaker notion than PRF. We denote the weak prf advantage as $\mathbf{Adv}^{\mathsf{wprf}}_{\mathsf{F}}(q,t)$.

IV-Based Encryption: An IV-Based Encryption ivE scheme is a tuple $\Psi := (\mathcal{K},\mathcal{N},\mathcal{M},\mathsf{Enc},\mathsf{Dec})$. Encryption algorithm Enc takes a key $K \in \mathcal{K}$ and a message $M \in \mathcal{M}$ and returns $(\mathsf{iv}, C) = \mathsf{Enc}(K, M)$, where $\mathsf{iv} \in \mathcal{N}$ is the initialization vector and $C \in \mathcal{M}$ is the ciphertext. Decryption algorithm Dec takes K, iv, C and returns $M = \mathsf{Dec}(K, \mathsf{iv}, C)$. Correctness condition says that for all $K \in \mathcal{K}$ and $M \in \mathcal{M}$ $\mathsf{Dec}(K, \mathsf{Enc}(K, M)) = M$. The Priv\$ advantage [14, 52–54] of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{priv\$}}_{\mathsf{ivE}}(\mathcal{A}) := \left|\Pr_{\mathsf{K}}\left[\mathcal{A}^{\mathsf{Enc}_\mathsf{K}} = 1\right] - \Pr_{\Gamma}\left[\mathcal{A}^{\Gamma} = 1\right]\right|$$

where $\mathsf{K} \leftarrow\$\, \mathcal{K}$ and Γ is a random function from $\mathcal{M} \to \mathcal{N}\times\mathcal{M}$. The Priv\$ security of ivE, is defined as

$$\mathbf{Adv}^{\mathsf{priv\$}}_{\mathsf{ivE}}(q,t) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{priv\$}}_{\mathsf{ivE}}(\mathcal{A}). \tag{3}$$

(Nonce-Based) Authenticated Encryption with Associated Data: A (nonce-based) authenticated encryption with associated data or NAEAD scheme $\mathfrak{A}$ consists of a key space $\mathcal{K}$, a (possibly empty) nonce space $\mathcal{N}$, a message space $\mathcal{M}$, an associated data space $\mathcal{A}$, and a tag space $\mathcal{T}$, along with two functions $\mathsf{Enc} : \mathcal{K}\times\mathcal{N}\times\mathcal{A}\times\mathcal{M} \to \mathcal{M}\times\mathcal{T}$, and $\mathsf{Dec} : \mathcal{K}\times\mathcal{N}\times\mathcal{A}\times\mathcal{M}\times\mathcal{T} \to \mathcal{M}\cup\{\perp\}$, with the correctness condition that for any $K \in \mathcal{K}, N \in \mathcal{N}, A \in \mathcal{A}, M \in \mathcal{M}$, we

must have $\mathsf{Dec}(K, N, A, \mathsf{Enc}(M)) = M$. When the nonce space is empty, we call the AE scheme a deterministic AE or DAE scheme.

Following the security definition in [14, 52–54], we define the NAEAD (DAE for deterministic AE) advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{ae}}_{\mathfrak{A}}(\mathcal{A}) := \left| \Pr_{\mathsf{K}} \left[\mathcal{A}^{\mathsf{Enc}_\mathsf{K}, \mathsf{Dec}_\mathsf{K}} = 1\right] - \Pr_{\Gamma} \left[\mathcal{A}^{\Gamma, \perp} = 1\right] \right|,$$

where $\mathsf{K} \xleftarrow{\$} \mathcal{K}$ and Γ is a random function from $\mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$, and $\perp$ is the reject oracle that takes (N, A, C, T) as input and returns the reject symbol $\perp$. The NAEAD/DAE security of $\mathfrak{A}$, is defined as

$$\mathbf{Adv}^{\mathsf{ae}}_{\mathfrak{A}}(q, t) := \max_{\mathcal{A}} \mathbf{Adv}^{\mathsf{ae}}_{\mathfrak{A}}(\mathcal{A}). \tag{4}$$

3 The Elastic-Tweak Framework

In this section, we introduce the Elastic-Tweak framework (illustrated in Figure 3.1) on SPN based block ciphers that allows one to efficiently design tweakable block ciphers with short tweaks. As the name suggests, Elastic-Tweak refers to elastic expansion of short tweaks and we typically consider tweaks of size less than or equal to 16 bits. Using this framework, one can convert a block cipher to a short tweak tweakble block cipher denoted by tBC. We briefly recall the SPN structure on which this framework would be applied. An SPN block cipher iterates for rnd many rounds, where each round consists of three operations:

(a) SubCells (divides the state into cells and substitutes each cell by an s-bit S-box which is always non-linear),
(b) LinLayer (uses a linear mixing layer over the full state to create diffusion), and
(c) AddRoundKey (add a round keys to the state).

The basic idea of the framework is to expand a small tweak (of size t) using a suitable linear code of high distance and then the expanded tweak (of size t_e) is injected (i.e. xored) to the internal block cipher state affecting a certain number of S-boxes (say, tic). We apply the same process after every gap number of rounds. An important feature of tBC is that it is implemented using very low tweak state and without any tweak schedule (only tweak expansion). In the following, we describe the linear code to expand the tweak and how to inject the tweak into the underlying block cipher state. If BC denotes the underlying SPN block cipher, we denote the tweakable block cipher as Twe BC $[t, t_e, \mathsf{tic}, \mathsf{gap}]$ where $t, t_e, \mathsf{tic}, \mathsf{gap}$ are suitable parameters as described above.

3.1 Exp: Expanding the Tweak

In this section, we describe our method to expand the tweak T of t bits to an expanded tweak T_e of t_e bits. We need the parameters to satisfy the following conditions:

(a) t_e is divisible by $2t$ and tic. Let $w := t_e/\mathsf{tic}$, the underlying word size.

(b) w divides t and $w \leq s$.

The tweak expansion, called Exp, follows an "Expand then (optional) Copy" style as follows:

(i) Let $\tau := t/w$, and we view $T = (T_1, \ldots, T_\tau)$ as a $1 \times \tau$ vector of elements from $\mathbb{F}_{2^w}$. We expand T by applying a $[2\tau, \tau, \tau]$-linear code[9] over $\mathbb{F}_{2^w}$ with the generating matrix $G_{\tau \times 2\tau} = [I_\tau : I_\tau \oplus J_\tau]$, where I_τ is the identity matrix of dimension τ and J is the all 1 square matrix of dimension τ over $\mathbb{F}_{2^w}$. Let $T' = T \cdot G$ be the resultant code. Note that, T' can be computed as $S \oplus T_1 \| \cdots \| S \oplus T_\tau$ where $S = T_1 \oplus \cdots \oplus T_\tau$.

(ii) Finally, we compute the expanded tweak by concatenating $t_e/2t$ many copies of T' i.e.

$$T_e = T' \| \cdots \| T'.$$

Note that, T_e can be viewed as an application of $[\mathsf{tic}, \tau, \mathsf{tic}/2]$-linear code on T. The main rationale behind the choice of this expansion function is that it generates high distance codes (which is highly desired from the cryptanalysis point of view) with a low cost (only $(2\tau - 1)$ addition over $\mathbb{F}_{2^w}$ is required).

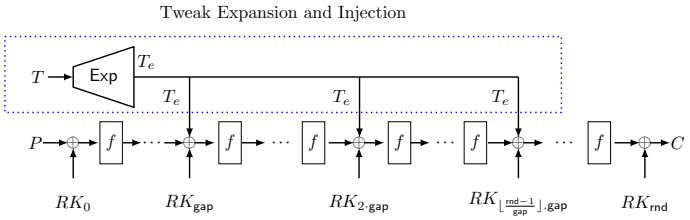

Fig. 3.1: Elastic-Tweak Construction.

3.2 Injecting Expanded Tweak into Round Functions

Note that the expanded tweak can be viewed as $T_{e,1} \| \cdots \| T_{e,\mathsf{tic}}$ where each $T_{e,i}$ is of size w-bits and $w \leq s$. Now we xor these tweak in addition to the round keys in tic number of S-boxes. The exact choices of S-box would be design specific so that the diffusion due to tweak difference is high.

The tweak injection is optional for each round, the tweak injection starts from round start and it is injected at an interval of gap rounds and stops at round end. To be precise, we inject tweak at the round number start, $\mathsf{start} + \mathsf{gap}$, $\mathsf{start} + 2.\mathsf{gap}$, $\ldots$, end. To have a uniformity in the tweak injection rounds, we typically choose $\mathsf{start} = \mathsf{gap}$ and inject the tweaks at an interval of gap rounds. This implicitly sets $\mathsf{end} = \mathsf{gap}.\lfloor \frac{\mathsf{rnd}-1}{\mathsf{gap}} \rfloor$.

[9] An $[n, k, d]$-linear code over a field $\mathbb{F}$ is defined by a $k \times n$ matrix G called the *generator* matrix over $\mathbb{F}$ such that for all nonzero vectors $v \in \mathbb{F}^k$, $v \cdot G$ has at least d many nonzero elements.

Function $\mathsf{Exp}[t_e, w](T)$	**Algorithm** tBC $[t_e, \mathsf{tic}, \mathsf{gap}](X, K, T)$
1. $\tau \leftarrow \frac{\lvert T \rvert}{w}$	1. $w \leftarrow t_e/tic$
2. $T_e \leftarrow \phi$	2. $T_e \leftarrow \mathsf{Exp}[t_e, w](T)$
3. $(T_1, T_2, \ldots, T_\tau) \xleftarrow{w} T$	3. **for** $i = 1$ **to** rnd
4. $T' \leftarrow T \| (T \oplus T \cdot J_\tau)$	4. $\quad X \leftarrow \mathsf{SubCells}(X)$
5. **for** $i = 1$ **to** $t_e/2t$	5. $\quad X \leftarrow \mathsf{LinLayer}(X)$
6. $\quad T_e \leftarrow T_e \| T'$	6. $\quad (K, X) \leftarrow \mathsf{AddRoundKey}(K, X, i)$
7. **return** T_e	7. $\quad$ **if** i % gap $= 0$ and $i <$ rnd
	8. $\quad\quad \mathsf{AddTweak}[\mathsf{tic}](X, T_e)$
	9. **return** X

Fig. 3.2: Function Exp and tBC. Here, $\mathsf{AddTweak}[\mathsf{tic}](X, T_e)$ represents the xoring tweak in to the state of the block cipher.

Requirements from Twe BC. We must ensure Twe BC should have same security level as the underlying block cipher.

From the performance point of view, our target is to obtain the above mentioned security

"minimizing t_e (signifies the area) and $t_e.\lfloor \frac{rnd-1}{gap} \rfloor$ (signifies the energy)."

Features of Twe BC.

1. Our tBC is applied to any SPN based block ciphers.
2. Due to linear expansion of tweak, tBC with zero tweak turns out to be same as the underlying block cipher (note that we keep same number of rounds as the block cipher). This feature would be useful to reduce overhead due to nonzero tweak. Later we see some applications (e.g., application on FCBC) where the nonzero tweaks is only applied to process the last block.

3.3 Tweakable GIFT and AES

In this section, we provide various instantiation of tBC built upon the two popular block ciphers GIFT and AES. We are primarily interested on tweak size $4, 8, 16$, and hence considered $t \in \{4, 8, 16\}$.

Instantiation of tBC with 4 bit Tweak. All the recommendations with 4-bit tweaks have extremely low overhead over the original block cipher and they can be ideal for reducing multiple keys scheme to an equivalent single key scheme instance with a minuscule loss in efficiency. Detailed description can be found in Sect. 4.

(i) GIFT-64[4, 16, 16, 4]. In this case the tweak is expanded from 4 bits to 16 bits and the expanded tweak is injected at bit positions $4i + 3$, for $i = 0, \ldots, 15$.

(ii) GIFT-128[4, 32, 32, 5]. Here we expand the 4 bit tweak to 32 bits and the expanded tweak is injected at bit positions $4i + 3$, for $i = 0, \ldots, 31$.

(iii) AES[4, 8, 8, 2]. Here we expand the 4 bit tweak to 8 bits and the expanded tweak is injected at the least-significant bits of each of the 8 S-Boxes in the top two rows.

Instantiation of tBC with 8 and 16 bit Tweak. tBC with tweak size of 8/16-bits are ideal for replacing the length counter bits (or masking) used in many constructions. Detailed description can be found in Sect. 4.

(i) AES[8, 16, 8, 2]. For 8 bit tweak, we only use AES. The tweak is first extended to 16 bits and the tweak is injected at the two least-significant bits of each of the 8 S-Boxes in the top two rows.

(ii) GIFT-128[16, 32, 32, 4]. Here we expand the 16 bit tweak to 32 bits and the expanded tweak is injected at bit positions $4i + 3$, for $i = 0, \ldots, 31$.

(iii) AES[16, 32, 8, 2]. Here we expand the 16 bit tweak to 32 bits and expanded tweak is injected at the four least-significant bits of each of the 8 S-Boxes in the top two rows.

Cryptanalysis of the proposed candidates: A detailed security analysis of all the proposed candidates is given in Sect 4 in the full version [51]. We remark that several of these candidates have already been analyzed in [47–50].

Performance: Sect 3.4 and Appendix C in the full version [51] summarize the hardware and software performance of all the proposed candidates.

4 Applications of Short-Tweak Tweakable Block Ciphers

In this section, we present some use cases where an efficient tBC would be beneficial.

4.1 Reducing the Key Size in Multi-Keyed Modes of Operation

Several block cipher based modes of operation employ a block cipher with multiple independently sampled keys. In general, this is done either to boost the security, or to simplify the analysis of the overall construction. The number of keys can be naturally reduced to a single key by replacing the multi-keyed block cipher with a single keyed tBC where distinct tweaks are used to simulate independent block cipher instantiations. Proposition 1 below gives the theoretical justification for this remedy. The proof is obvious from the definitions of (tweakable) random permutation.

Proposition 1. *For some fixed $t \in \mathbb{N}$, and $k \in [2^t]$. Let $(\Pi_1, \ldots, \Pi_k) \leftarrow\!\!\$\, (\mathsf{Perm}[n])^k$ and $\widetilde{\Pi} \leftarrow\!\!\$\, \mathsf{TPerm}[t, n]$. Let $\mathcal{O}_{\Pi;k}$ and $\mathcal{O}_{\widetilde{\Pi};k}$ be two oracles giving bidirectional access to $(\Pi_1, \ldots, \Pi_k)$, and $(\widetilde{\Pi}^1, \ldots, \widetilde{\Pi}^k)$, respectively. Then, for all distinguisher $\mathcal{A}$, we have*

$$\Delta_{\mathcal{A}}(\mathcal{O}_{\Pi;k}; \mathcal{O}_{\widetilde{\Pi};k}) := \left|\Pr[\mathcal{A}^{\mathcal{O}_{\Pi;k}} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\widetilde{\Pi};k}} = 1]\right| = 0.$$

Now, we demonstrate the utility of this idea through some examples.

FCBC MAC: FCBC mode is a 3-key message authentication code, by Black and Rogaway [16], which is defined as follows:

$$\Sigma := \mathsf{E}_{K_0}\Big(M_{m-1} \oplus \mathsf{E}_{K_0}\big(M_{m-2} \oplus \mathsf{E}_{K_0}\big(\cdots \oplus (M_2 \oplus \mathsf{E}_{K_0}(M_1))\big)\big)\Big),$$
$$\mathsf{FCBC}[\mathsf{E}](M) := \mathsf{E}_{K_t}\big(\Sigma \oplus \mathsf{ozp}(M_m)\big)\,, \text{ where } t \leftarrow (|M_m| = n)?\ 1 : 2.$$

FCBC has not received much appreciation in its existing 3-key form, even though it offers better security, $O(q^2/2^n + q\ell^2/2^n + q^2\ell^4/2^{2n})$ in [17, 18, Theorem 3 and Remark 5], than CMAC [4, 55], $O(q^2\ell/2^n + q^2\ell^4/2^{2n})$ in [19, Theorem 4.6]. Quantitatively, the number of queries per key increases from $2^{3n/8}$ to $2^{n/2}$ for message lengths up to $2^{n/4}$ blocks. This is mainly due to presence of three keys which not only costs keys size of the algorithm but it requires to run three key scheduling algorithms. Keeping these in mind, we define Twe-FCBC, as follows:

$$\Sigma := \widetilde{\mathsf{E}}^0_K\Big(M_{m-1} \oplus \widetilde{\mathsf{E}}^0_K\big(M_{m-2} \oplus \widetilde{\mathsf{E}}^0_K\big(\cdots \oplus (M_2 \oplus \widetilde{\mathsf{E}}^0_K(M_1))\big)\big)\Big),$$
$$\mathsf{Twe\text{-}FCBC}[\widetilde{\mathsf{E}}](M) := \widetilde{\mathsf{E}}^t_K\big(\Sigma \oplus \mathsf{ozp}(M_m)\big)\,, \text{ where } t \leftarrow (|M_m| = n)?\ 1 : 2.$$

It is clear that Twe-FCBC is a variant of FCBC, that follows the principle established in Proposition 1, and replaces the 3 block ciphers E_{K_0}, E_{K_1}, E_{K_2} with $\widetilde{\mathsf{E}}^0_K$, $\widetilde{\mathsf{E}}^1_K$ and $\widetilde{\mathsf{E}}^2_K$, respectively. Using Proposition 1 and [18, Theorem 3 and Remark 5], we get the PRF security for Twe-FCBC in a straightforward manner in Proposition 2.

Proposition 2. *Assuming all queries are of length $\ell \leq 2^{n/4}$, and $\sigma \leq q\ell$, we have*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{Twe\text{-}FCBC}[\widetilde{\mathsf{E}}]}(t, q, \sigma) \leq \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(t', \sigma) + O\left(\frac{q^2}{2^n}\right).$$

Clearly, Twe-FCBC has two major advantages over CMAC- (i) no need to hold an additional state for final message block masking, (ii) security bound is free of length factor for all reasonably sized messages (close to 6 Gigabyte for a 128-bit block cipher). In addition, Twe-FCBC can also avoid the additional block cipher call used to generate the masking. Due to backward compatibility, except the last block we have used the original block cipher. So the performance overhead due to nonzero tweak only applies to the last block cipher call. This features ensures to get similar performance (or even better) for long message.

Double Block Hash-then-Sum: The very basic version of Double-block Hash-then-Sum or DbHtS [56], is defined as below

$$\mathsf{DbHtS}(M) := \mathsf{E}_{K_1}(\Sigma) \oplus \mathsf{E}_{K_2}(\Theta),$$

where H is a $2n$-bit output hash function, $(\Sigma, \Theta) := \mathsf{H}_L(M)$, and L, K_1, K_2 are all sampled independently. DbHtS is a generic design paradigm that captures several popular BBB secure MACs such as PMAC_Plus, LightMAC_Plus, SUM_ECBC and 3kf9. Using a tBC, the two block cipher keys can now simply be replaced by a single tweakable block cipher key and two distinct tweaks. Formally, we define Twe-DbHtS as follows

$$\mathsf{Twe\text{-}DbHtS}(M) := \widetilde{\mathsf{E}}^1_K(\Sigma) \oplus \widetilde{\mathsf{E}}^2_K(\Theta).$$

Moreover, one can also generate the dedicated hash key using the tweakable block cipher key itself. Suppose the hash function is block cipher based, then the tBC key can be used along with a different tweak to replace the dedicated hash key. In all other cases, the hash key can be derived as $L := (\widetilde{\mathsf{E}}^0_K(0) \| \widetilde{\mathsf{E}}^0_K(1) \| \cdots \| \widetilde{\mathsf{E}}^0_K(h-1))$, where $|L| = hn$. Since $\widetilde{\mathsf{E}}^0_K(i)$'s are sampled in without replacement manner, this adds an additional factor of $\frac{h^2}{2^n}$ due to the PRP-PRF switching, which can be ignored for small h. One can easily verify that due to Proposition 1, the result on DbHtS [56, Theorem 2.(iii)] also applies to Twe-DbHtS. Formally, the security of Twe-DbHtS is given by Proposition 3.

Proposition 3.

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{Twe\text{-}DbHtS}[\mathsf{H},\widetilde{\mathsf{E}}]}(q,\ell,t) \leq 2\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(2q,t') + \mathbf{Adv}^{\mathsf{prf}}_{C_3^*[H,\pi_0,\pi_1,\pi_2]}(q,\ell,t).$$

In this way, we have one-key versions of different well known designs PMAC_Plus, LightMAC_Plus, SUM_ECBC, 3kf9 etc. We note that one key version of PMAC_Plus based on solely block cipher has been proposed [57]. However, one key version of the other designs either are not known or it can be shown to be secure up to the birthday bound.[10]

Sum of Permutations: The sum of permutations is a popular approach of constructing an n-bit length preserving PRF. Given 2 independent instantiations, E_{K_0} and E_{K_1}, of a secure block cipher over $\{0,1\}^n$, the sum of permutations, denoted XOR2, is defined by the mapping $x \mapsto \mathsf{E}_{K_0}(x) \oplus \mathsf{E}_{K_1}(x)$. The XOR2 construction has been proved to be n-bit secure [11]. There is a single key variant of XOR2, but it sacrifices one bit (i.e. defined from $\{0,1\}^{n-1}$ to $\{0,1\}^n$) for domain separation. Instead, we can use a tBC to simply replace the two block cipher keys with one tBC key and two distinct tweaks. We define $\mathsf{Twe\text{-}XOR2}(x) := \widetilde{\mathsf{E}}^0_K(x) \oplus \widetilde{\mathsf{E}}^1_K(x)$. Again combining Proposition 1 with [11, Theorem 4], we obtain

[10] 1kf9 is proposed in the ePrint version [58], which later found to be attacked in birthday complexity [59].

Proposition 4. *For $q \leq 2^{n-4}$,*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{Twe\text{-}XOR2}}(t,q) \leq \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(t',q) + (q/2^n)^{1.5}.$$

Tweaking Various Other Constructions: In the following list, we apply similar technique as above to several other constructions with multiple keys. The security of all the tBC-based variants is similar to the multi-key original constructions, so we skip their explicit security statements.

1. **Encrypted Davis Meyer (EDM) [12]:** EDM uses two keys and obtains BBB PRF security. We define the tBC-based variant as follows:

$$\mathsf{Twe\text{-}EDM}(x) := \widetilde{\mathsf{E}}^1_K(\widetilde{\mathsf{E}}^0_K(x) \oplus x).$$

2. **Encrypted Wegman Carter Davis Meyer (EWCDM) [12]:** EWCDM is a nonce-based BBB secure MAC that requires two block cipher keys and a hash key. The tBC-based variant of EWCDM is defined as:

$$\mathsf{Twe\text{-}EWCDM}(N,M) := \widetilde{\mathsf{E}}^2_K\big(\widetilde{\mathsf{E}}^1_K(N) \oplus N \oplus H_{\widetilde{\mathsf{E}}^0_K(0)}(M)\big).$$

3. **Chained LRW2 (CLRW2) [13]:** The CLRW2 construction is a TBC that achieves BBB TSPRP security using two independent block cipher keys and two independent hash keys. We define a tBC-based variant of CLRW2 as follows:

$$\mathsf{Twe\text{-}CLRW2}(M,T) := \widetilde{\mathsf{E}}^2_K\big(\widetilde{\mathsf{E}}^1_K(M \oplus h_{L_1}(T)) \oplus h_{L_1}(T) \oplus h_{L_2}(T)\big) \oplus h_{L_2}(T),$$

   where L_1 and L_2 can be easily derived using $\widetilde{\mathsf{E}}$ with dedicated independent tweaks. It is easy to see that one can easily extend the idea to obtain single keyed CLRWr [60] using r distinct tweaks.
4. **GCM-SIV-2 [14].** GCM-SIV-2 is an MRAE scheme with $2n/3$-bit security. However, it requires 6 independent block cipher keys along with 2 independent hash keys. We can easily make it single keyed using a tBC:

$$\begin{aligned}
V_1 &:= H_{\widetilde{\mathsf{E}}^0_K(0)}(N,A,M)\,,\ V_2 := H_{\widetilde{\mathsf{E}}^0_K(1)}(N,A,M)\\
T_1 &:= \widetilde{\mathsf{E}}^1_K(V_1) \oplus \widetilde{\mathsf{E}}^2_K(V_2)\,,\ T_2 := \widetilde{\mathsf{E}}^3_K(V_1) \oplus \widetilde{\mathsf{E}}^4_K(V_2),\\
C_i &:= M_i \oplus \widetilde{\mathsf{E}}^5_K(T_1 \oplus i) \oplus \widetilde{\mathsf{E}}^6_K(T_2 \oplus i).
\end{aligned}$$

   Extending the same approach, one can get a single keyed version of GCM-SIV-$r$as well.
5. **The Benes Construction [15]:** The Benes construction is a method to construct $2n$-bit length preserving PRF construction with n-bit security that uses 8 independent n bit to n bit PRFs. Formally,

$$\begin{aligned}
L' &:= f_1(L) \oplus f_2(R)\\
R' &:= f_3(L) \oplus f_4(R)\\
\mathsf{Benes}(L,R) &:= (f_5(L') \oplus f_6(R'), f_7(L') \oplus f_8(R')).
\end{aligned}$$

Now these f_i functions can be constructed using sum of two permutations, however that would essentially require 16 block cipher keys. With a tBC, we can reduce the number of keys to one by instantiating $f_i := \widetilde{\mathsf{E}}_K^{2i} \oplus \widetilde{\mathsf{E}}_K^{2i+1}$ for each $i \in [8]$.

4.2 Efficient Processing for Short Messages

In energy constrained environments, reducing the number of primitive invocations is crucial, as for short messages, this reduction leads to efficient energy consumption. The tBC framework can be used to reduce the number of primitive invocations for many existing constructions such as LightMAC_Plus [61].

LightMAC_Plus is a counter-based PMAC_Plus in which $\langle i \rangle_m \| M_i$ is input to the i-th keyed block cipher call, where $\langle i \rangle_m$ is the m-bit binary representation of i and M_i is the i-th message block of $n - m$ bits. The counters ensure that there is no input collision, which indirectly helps in negating the influence of ℓ. LightMAC_Plus has been shown to have $O(q^3/2^{2n})$ PRF security. However, it has two shortcomings: (i) it requires 3 keys, and (ii) it has rate $1 - m/n$ which increases the number of block cipher calls. This is highly undesirable in low memory and energy constrained scenarios.

To resolve these shortcomings specifically for short to moderate length messages (slightly less than 1 Megabyte), we propose Twe-LightMAC_Plus, which can be viewed as an amalgamation of LightMAC_Plus [61] and PMACx [33]. The key

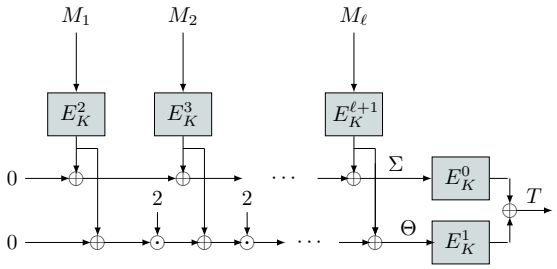

Fig. 4.1: Twe-LightMAC construction.

idea is to use the block counters as tweak in hash layer, while having distinct tweaks for the finalization. The pictorical description of the algorithm is given in Fig. 4.1. It is easy to see that Twe-LightMAC_Plus is single-keyed and it achieves rate 1. This reduces the number of block cipher calls by up to 50% for short messages, which has direct effect on reducing the energy consumption. We claim that Twe-LightMAC_Plus is as secure as LightMAC_Plus. Formally, we have the following security result. We note that similar improvements can also be applied to PMAC, PMAC_Plus.

Proposition 5. *For $q \leq 2^{n-1}$,*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{Twe\text{-}LightMAC\_Plus}[\widetilde{\mathsf{E}}]}(t, q, \ell) \leq \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(t', q\ell) + O\left(\frac{q^3}{2^{2n}}\right).$$

Proof. Twe-LightMAC_Plus is an instance of Twe-DbHtS, and hence offers similar security. The security bound of Twe-DbHtS includes a term

$$\mathbf{Adv}^{\mathsf{prf}}_{C_3^*[H,\pi_0,\pi_1,\pi_2]}(q, \ell, t)$$

from [56]. One can verify from [56, Proof of Theorem 2.(iii)], that this term is predominantly bounded by two probabilities:

1. $\Pr[\exists$ distinct i, j, k such that $\Sigma_i = \Sigma_j, \Theta_i = \Theta_k]$.
2. $\Pr[\exists$ distinct i, j such that $\Sigma_i = \Sigma_j, \Theta_i = \Theta_j]$.

Now the hash layer of Twe-LightMAC_Plus is exactly same as the PHASHx of [33]. Using similar arguments as in [33, Proof of Theorem 1] it can be shown that 1. is upper bounded by $O(q^3/2^{2n})$, and 2. is upper bounded by $O(q^2/2^{2n})$. The result follows by combining 1 and 2. □

4.3 A Note on tBC's Advantages over XE and XEX

The XE and XEX modes, by Rogaway [36], are two reasonably efficient ways of converting a block cipher into a tweakable block cipher. These methods are widely used in various modes such as PMAC [44], OCB [45], COPA [29], ELmD [46] etc. The XE scheme to generate a TBC $\widetilde{\mathsf{E}}$ from a BC E is defined as

$$\mathsf{XE}: \ \widetilde{\mathsf{E}}_K^{i_1,\cdots,i_t}(M) := \mathsf{E}_K(\Delta \oplus M)$$

where $\Delta = \alpha_1^{i_1} \cdots \alpha_t^{i_t} \cdot L$. Here L is generally an n-bit secret state, which is generated using block cipher call.[11] It is sufficient for us to compare XE and tBC, as XEX is much similar to XE. Now one may think of using XE instead of tBC to convert multi-keyed modes to single-keyed mode, as above. But in comparison to tBC, XE lacks two important features:

1. Degradation to Birthday Bound Security: XE (and XEX) is proved to be birthday bound secure TBC mode. This is not a big issue for birthday secure multi-keyed modes. In fact, the CMAC mode can be viewed as an example that uses the XE mode, much in the same way as Twe-FCBC uses tBC. However, if we use XE in multi-keyed applications such as DbHtS or XOR2, the security of these constructions would degrade to birthday bound. So, we cannot use XE or XEX, in a black box fashion, to instantiate the tweakable variants, without a significant degradation in the security of the modified mode. In contrast, tBC directly works on the block cipher level, and hence does not suffer from such degradation unless the block cipher is itself weak.

[11] Alternative constructions to define Δ can be found in [62, 63].

2. Additional Computational and Storage Overheads: The XE mode requires, precomputation of the secret state L, (ii) an additional block cipher invocation to generate L, and (iii) an additional storage to store L. This cannot be neglected in constrained computation and communication environments, as mentioned earlier. On the other hand, the tBC framework incurs far less overheads. In this respect, one can easily define simple tBC-variants of PMAC [36] (based on XE), COPE [29] (based on XEX), COLM [64] (XE like processing) etc. much along the same line as Twe-LightMAC_Plus.

5 Simplification of Authenticated Encryption Schemes

In this section, we demonstrate some AE schemes that achieve a combination of advantages discussed in section 4.

5.1 Twe-CLOC and Twe-SILC

We propose tBC variant for CLOC and SILC, called Twe-CLOC and Twe-SILC, respectively. CLOC and SILC are nonce-based authentication encryption (NAEAD) modes, which aim to optimize the implementation overhead beyond the block cipher calls, the precomputation complexity, and the memory requirement. CLOC is suitable for uses in embedded processors, and SILC aims to optimize hardware implementation cost. Our choices of CLOC and SILC are motivated by two factors (see subsection 5.2 below): design simplification and reduction in block cipher calls.

The three tBC variants are described in Fig. 5.1. We have made minimal changes in the original schemes. CLOC and SILC employ Encrypt-then-PRF paradigm and use a variant of CFB [3] mode in its encryption part and a variant of FCBC in the authentication part.

$\mathsf{CFB}(V, M, t)$

1. $M_1 \| \cdots \| M_m \leftarrow M$
2. $C_1 \leftarrow V \oplus M_1$
3. **for** $i = 2$ **to** m
4. $\quad C_i \leftarrow \lfloor \widetilde{\mathscr{E}}_K^t(C_{i-1}) \rfloor_{|M_i|} \oplus M_i$
5. **return** $(C_1 \| \cdots \| C_m)$

$\mathsf{ivFCBC}(T, D, t_0, t_1, t_2)$

1. $D_1 \| \cdots \| D_d \leftarrow D$
2. **for** $i = 1$ **to** $d - 1$
3. $\quad T \leftarrow \widetilde{\mathscr{E}}_K^{t_0}(T \oplus D_i)$
4. $t \leftarrow (|D_d| = n)?\ t_1 : t_2$
5. $T \leftarrow \widetilde{\mathscr{E}}_K^{t}(T \oplus \mathsf{pad}(D_d))$
6. **return** T

$\mathsf{Twe\text{-}SILC}_K(N, A, M)$

1. $T \leftarrow \mathsf{ivFCBC}(0^n, N \| A, 0, 0 \| 1, \mathsf{Len} \| 1)$
2. $C \leftarrow \mathsf{CFB}(T, M, 0 \| 2)$
3. $T \leftarrow \mathsf{ivFCBC}(0, C, 1, 0 \| 3, \mathsf{Len} \| 0)$
4. **return** (C, T)

$\mathsf{Twe\text{-}CLOC}_K(N, A, M)$

1. $T \leftarrow \mathsf{ivFCBC}(T, A \| N, 0, 1, 2)$
2. $C \leftarrow \mathsf{CFB}(T, M, 3)$
3. $T \leftarrow \mathsf{ivFCBC}(0, C, 4, 5, 6)$
4. **return** (C, T)

Fig. 5.1: Encryption and algorithm of Twe-SILC and Twe-CLOC. pad uses 10^* padding for Twe-CLOC and 0^* padding for Twe-SILC.

5.2 Features of the Proposed AE Schemes

The proposed tBC-based AE schemes offer two added features over the existing block cipher based schemes.

Design Simplification: Twe-CLOC and Twe-SILC simplifies their respective original algorithms very efficiently. CLOC and SILC require several linear functions (f, g_1, g_2, h_1, h_2 for CLOC and g for SILC) for domain separations and bit fixing operations. Twe-CLOC and Twe-SILC perform all the domain separations by using distinct tweaks, which significantly simplifies the design.

Table 5.1: Comparison between the number of (tweakable) block cipher invocations for original CLOC and SILC, and their tBC counterparts. Here a, and m denote the length of associated data and plaintext, respectively.

Modes	No. of BC calls		No. of tBC calls	
	$a \neq 0$	$a = 0$	$a \neq 0$	$a = 0$
CLOC	$a + 2m + 1$	$2m + 2$	$a + 2m$	$2m$
SILC	$a + 2m + 3$	$2m + 2$	$a + 2m$	$2m$

Energy Efficient for Short Inputs: Apart from the simplification of the original designs, the proposed AE schemes offer another advantage over the non-tweaked versions. They require lesser number of block cipher calls for shorter/empty AD or message processing, which essentially makes them more efficient in terms of energy consumption. The number of block cipher invocations required to process an associated data of a blocks and message of m blocks are given in Table 5.1. As seen from the table, SILC requires 4 block cipher calls to process 1 block AD and empty message, Twe-SILC requires only 1 block cipher call.

5.3 Security of the Proposed AE Schemes

Twe-CLOC and Twe-SILC are in essence just the multi-key variants of CLOC and SILC, respectively. So, intuitively they should be at least as secure as the original modes, and the security argument for these schemes is relatively easier than the original schemes. We show in Proposition 6 that our intuitions are correct to a large extent. For the sake of simplicity, we refrain from giving exact bounds, and instead give the asymptotic expressions.

We first look at the abstract design paradigm behind Twe-CLOC and Twe-SILC, which is the so-called Encrypt-then-PRF, or EtPRF.

The EtPRF Paradigm: EtPRF [53, Construction A5] is a design paradigm to construct NAEAD schemes. It is composed of three stages (illustrated in Figure 5.2): a random IV generator, G that generates iv using the nonce N and (possibly) the AD A; an IV-based encryption phase, ivE that generates the ciphertext C using iv as the random IV; and a tag-generation phase, F that generates the tag on the input N, A, C. Formally, for key space $\mathcal{K} \times \mathcal{L}$ the encryption algorithm of EtPRF is defined by the following mapping

$$(K, L, N, A, M) \mapsto \mathsf{ivE}(K, N, A, M) \,\|\, \mathsf{F}\left(L, N, A, \mathsf{ivE}(K, N, A, M)\right),$$

for all $(L, K, N, A, M) \in \mathcal{L} \times \mathcal{K} \times \mathcal{A} \times \mathcal{M}$. Here, $C := \mathsf{ivE}(K, N, A, M) \in \mathcal{M}$, and $T := \mathsf{F}(L, N, A, C) \in \mathcal{M}$. Note that, for the sake of simplicity we subsumed the G function within the ivE phase. In [53], Namprempre et al. showed that the NAEAD security of an EtPRF scheme, $\mathfrak{A}$, given by:

$$\mathbf{Adv}^{\mathsf{ae}}_{\mathfrak{A}}(q, \ell, \sigma) \leq \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(q, \ell, \sigma) + \mathbf{Adv}^{\mathsf{prf}}_{\mathsf{G}}(q, \ell, \sigma) + \mathbf{Adv}^{\mathsf{priv\$}}_{\mathsf{ivE}}(q, \ell, \sigma), \quad (5)$$

where PRIV denotes the Priv\$ security (see section2.1).

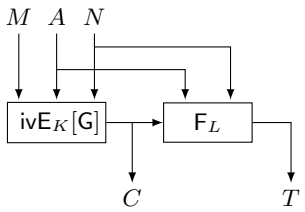

Fig. 5.2: The EtPRF paradigm based on an IV-based encryption scheme ivE for the encryption phase, and a PRF F for the tag generation phase. The $[\mathsf{G}]$ denotes that ivE internally uses G to generate the random IV.

In case of both $\mathsf{Twe\text{-}CLOC}$ and $\mathsf{Twe\text{-}SILC}$, G and F are variants of $\mathsf{Twe\text{-}FCBC}$, and hence can be shown to have $O(\sigma^2/2^n)$ PRF security [16]. ivE phase is an instance of the CFB mode with random IV, which has been shown to have $O(\sigma^2/2^n)$ security in [65]. Hence, by substituting the relevant bounds in Eq. (5), we get the following security result for $\mathsf{Twe\text{-}CLOC}$ and $\mathsf{Twe\text{-}SILC}$.

Proposition 6. *The security of Twe-CLOC and Twe-SILC is given by:*

$$\mathbf{Adv}^{\mathsf{ae}}_{\mathit{Twe\text{-}CLOC}[\widetilde{\mathsf{E}}]}(t, q, \ell, \sigma) \leq \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(t', q\ell) + O(\frac{\sigma^2}{2^n}),$$
$$\mathbf{Adv}^{\mathsf{ae}}_{\mathit{Twe\text{-}SILC}[\widetilde{\mathsf{E}}]}(t, q, \ell, \sigma) \leq \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{\mathsf{E}}}(t', q\ell) + O(\frac{\sigma^2}{2^n}).$$

where t, q, ℓ, σ denote the computational time, query bound, maximum query length, and the total number of tBC calls across all encryption and decryption queries, respectively.

Remark 1. The security of CLOC and SILC do not follow from Eq. (5), in a straightforward way, as the tag generation and encryption share the same key.

6 Further Applications and Future Directions

We think that tBC can have several other applications. For instance, consider a scenario where two multiple algorithms are running on the same platform, sharing the same secret key. We could find several examples where such an arrangement could be vulnerable. For example, consider a scenario where $\mathsf{AES\text{-}GCM}$ and $\mathsf{AES\text{-}CMAC}$ are running on the same device, sharing the same secret

key. Now, it is easy to see that, an adversary can trivially forge a tag for AES-CMAC using an encryption query on AES-GCM. tBC can efficiently take care of such problems by separating these algorithms using different tweak values, i.e. unique tweak values for each of these algorithms.

We have defined the Elastic-Tweak framework for SPN based block ciphers. Extending this further for ARX based constructions could be an interesting problem. Also, it would be interesting to see designs for short-tweak tweakable public permutations, which might have strong impact on the simplification of permutation based constructions such as Sponge, Beetle, Minalpher etc.

Acknowledgements. The authors would like to thank all the anonymous reviewers of Indocrypt 2021 for their valuable comments. Prof. Mridul Nandi is supported by the project "Study and Analysis of IoT Security" by NTRO under the Government of India at R.C.Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata. Dr. Ashwin Jha's work was carried out in the framework of the French-German-Center for Cybersecurity, a collaboration of CISPA and LORIA.

References

1. NIST: Data Encryption Standard (AES). FIPS Publication (Withdrawn) **46-3** (1999)
2. 197, N.F.: Advanced Encryption Standard (AES). Federal Information Processing Standards Publication **197** (2001)
3. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A (2001) National Institute of Standards and Technology.
4. Dworkin, M.: Recommendation for Block Cipher Modes of Operation – Methods and Techniques. NIST Special Publication 800-38A, National Institute of Standards and Technology, U. S. Department of Commerce (2001)
5. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C (2004) National Institute of Standards and Technology.
6. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Advances in Cryptology - EUROCRYPT '93, Proceedings. (1993) 386–397
7. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Advances in Cryptology - CRYPTO '90, Proceedings. (1990) 2–21
8. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007) National Institute of Standards and Technology.
9. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In: CHES 2017. Proceedings. (2017) 321–345
10. Patarin, J.: Security in $O(2^n)$ for the Xor of Two Random Permutations - Proof with the standard H technique -. IACR Cryptology ePrint Archive **2013** (2013) 368

11. Dai, W., Hoang, V.T., Tessaro, S.: Information-Theoretic Indistinguishability via the Chi-Squared Method. In: Advances in Cryptology - CRYPTO 2017. Proceedings, Part III. (2017) 497–523
12. Cogliati, B., Seurin, Y.: EWCDM: An Efficient, Beyond-Birthday Secure, Nonce-Misuse Resistant MAC. In: CRYPTO 2016, Proceedings, Part I. (2016) 121–149
13. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable Blockciphers with Beyond Birthday-Bound Security. In: Advances in Cryptology - CRYPTO 2012. Proceedings. (2012) 14–30
14. Iwata, T., Minematsu, K.: Stronger Security Variants of GCM-SIV. IACR Cryptology ePrint Archive **2016** (2016) 853
15. Patarin, J.: A Proof of Security in $O(2^n)$ for the Benes Scheme. In: Progress in Cryptology - AFRICACRYPT 2008. (2008) 209–220
16. Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. J. Cryptology **18**(2) (2005) 111–131
17. Jha, A., Nandi, M.: Revisiting Structure Graphs: Applications to CBC-MAC and EMAC. J. Mathematical Cryptology **10**(3-4) (2016) 157–180
18. Jha, A., Nandi, M.: Revisiting Structure Graph and Its Applications to CBC-MAC and EMAC. IACR Cryptology ePrint Archive **2016** (2016) 161
19. Nandi, M.: Improved Security Analysis for OMAC as a Pseudorandom Function. J. Mathematical Cryptology **3**(2) (2009) 133–148
20. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: CLOC and SILC. Submission to CAESAR (2016) `https://competitions.cr.yp.to/round3/clocsilcv3.pdf`.
21. Karlof, C., Sastry, N., Wagner, D.: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In: Proceedings of Embedded Networked Sensor Systems. SenSys '04, ACM (2004) 162–175
22. 11898, I.: CAN FD Standards and Recommendations `https://www.can-cia.org/news/cia-in-action/view/can-fd-standards-and-recommendations/2016/9/30/`.
23. EPCglobal: Electronic Product Code (EPC) Tag Data Standard (TDS). Technical Report `http://www.epcglobalinc.org/standards/tds/`.
24. Banik, S., Bogdanov, A., Luykx, A., Tischhauser, E.: SUNDAE: Small Universal Deterministic Authenticated Encryption for the Internet of Things. IACR Transactions on Symmetric Cryptology **2018**(3) (Sep. 2018) 1–35
25. Luykx, A., Preneel, B., Tischhauser, E., Yasuda, K.: A MAC Mode for Lightweight Block Ciphers. In: FSE 2016. (2016) 43–59
26. Schroeppel, R.: The Hasty Pudding Cipher. Submitted candidate for AES (1998)
27. Liskov, M., Rivest, R.L., Wagner, D.A.: Tweakable Block Ciphers. In: CRYPTO 2002. (2002) 31–46
28. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: FSE. (2011) 306–327
29. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v.2. Submission to CAESAR (2015) `https://competitions.cr.yp.to/round2/aescopav2.pdf`.
30. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1.41. Submission to CAESAR (2016) `https://competitions.cr.yp.to/round3/deoxysv141.pdf`.
31. Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication. In: Advances in Cryptology - CRYPTO '17. Proceedings, Part III. (2017) 34–65
32. Cogliati, B., Lampe, R., Seurin, Y.: Tweaking Even-Mansour Ciphers. In: CRYPTO 2015. Proceedings, Part I. (2015) 189–208

33. List, E., Nandi, M.: ZMAC+ - An Efficient Variable-output-length Variant of ZMAC. IACR Trans. Symmetric Cryptol. **2017**(4) (2017) 306–325
34. Crowley, P.: Mercy: A Fast Large Block Cipher for Disk Sector Encryption. In: Fast Software Encryption – FSE 2000. Proceedings. (2000) 49–63
35. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. In: Submission to NIST (round 3), 7(7.5):3. (2010)
36. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Advances in Cryptology - ASIACRYPT 2004. Proceedings. (2004) 16–31
37. Jha, A., List, E., Minematsu, K., Mishra, S., Nandi, M.: XHX - A Framework for Optimally Secure Tweakable Block Ciphers from Classical Ciphers and Universal Hashing. IACR Cryptology ePrint Archive **2017** (2017) 1075
38. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In: Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part II. (2014) 274–288
39. Jean, J., Nikolić, I., Peyrin, T.: KIASU v1. Submission to CAESAR (2016) `https://competitions.cr.yp.to/round1/kiasuv1.pdf`.
40. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In: Advances in Cryptology - CRYPTO 2016. Proceedings, Part II. (2016) 123–153
41. Dobraunig, C., Eichlseder, M., Mendel, F.: Square Attack on 7-Round Kiasu-BC. In: Applied Cryptography and Network Security - ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. (2016) 500–517
42. Dobraunig, C., List, E.: Impossible-Differential and Boomerang Cryptanalysis of Round-Reduced Kiasu-BC. In: Topics in Cryptology - CT-RSA 2017 - San Francisco, CA, USA, February 14-17, 2017, Proceedings. (2017) 207–222
43. Tolba, M., Abdelkhalek, A., Youssef, A.M.: A Meet in the Middle Attack on Reduced Round Kiasu-BC. IEICE Transactions **99-A**(10) (2016) 1888–1890
44. Black, J., Rogaway, P.: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In: Advances in Cryptology - EUROCRYPT 2002. Proceedings. (2002) 384–397
45. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3) (2003) 365–403
46. Datta, N., Nandi, M.: Proposal of ELmD v2.1. Submission to CAESAR (2015) `https://competitions.cr.yp.to/round2/elmdv21.pdf`.
47. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: INT-RUP Secure Lightweight Parallel AE Modes. IACR Trans. Symmetric Cryptol. **2019**(4) (2019) 81–118
48. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: ESTATE: A Lightweight and Low Energy Authenticated Encryption Mode. IACR Trans. Symmetric Cryptol. **2020**(S1) (2020) 350–389
49. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: LOTUS-AEAD and LOCUS-AEAD. Submission to NIST LwC Standardization Process (Round 2) (2019)
50. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: ESTATE. Submission to NIST LwC Standardization Process (Round 2) (2019)

51. Chakraborti, A., Datta, N., Jha, A., López, C.M., Nandi, M., Sasaki, Y.: Elastic-Tweak: A Framework for Short Tweak Tweakable Block Cipher. IACR Cryptol. ePrint Arch. (2019) 440
52. Gueron, S., Lindell, Y.: GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In: ACM SIGSAC Conference on Computer and Communications Security 2015. Proceedings. (2015) 109–119
53. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. In: Advances in Cryptology - EUROCRYPT 2014. Proceedings. (2014) 257–274
54. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. (2006) 373–390
55. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: FSE. (2003) 129–153
56. Datta, N., Dutta, A., Nandi, M., Paul, G.: Double-block Hash-then-Sum: A Paradigm for Constructing BBB Secure PRF. IACR Trans. Symmetric Cryptol. **2018**(3) (2018) 36–92
57. Datta, N., Dutta, A., Nandi, M., Paul, G., Zhang, L.: Single Key Variant of PMAC_Plus. IACR Trans. Symmetric Cryptol. **2017**(4) (2017) 268–305
58. Datta, N., Dutta, A., Nandi, M., Paul, G., Zhang, L.: Single Key Variant of PMAC_Plus. IACR Cryptology ePrint Archive **2017** (2017) 848
59. Leurent, G., Nandi, M., Sibleyras, F.: Generic Attacks Against Beyond-Birthday-Bound MACs. In: Advances in Cryptology - CRYPTO 2018. Proceedings, Part I. (2018) 306–336
60. Lampe, R., Seurin, Y.: Tweakable Blockciphers with Asymptotically Optimal Security. In: FSE 2013. Revised Selected Papers. (2013) 133–151
61. Naito, Y.: Blockcipher-Based MACs: Beyond the Birthday Bound Without Message Length. In: Advances in Cryptology - ASIACRYPT 2017. Proceedings, Part III. (2017) 446–470
62. Chakraborty, D., Sarkar, P.: A General Construction of Tweakable Block Ciphers and Different Modes of Operations. IEEE Trans. Information Theory **54**(5) (2008) 1991–2006
63. Granger, R., Jovanovic, P., Mennink, B., Neves, S.: Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption. In: EUROCRYPT 2016. Proceedings, Part I. (2016) 263–293
64. Andreeva, E., Bogdanov, A., Datta, N., Luykx, A., Mennink, B., Nandi, M., Tischhauser, E., Yasuda, K.: COLM v1. Submission to CAESAR (2016) `https://competitions.cr.yp.to/round3/colmv1.pdf`.
65. Wooding, M.: New Proofs for Old Modes. IACR Cryptology ePrint Archive **2008** (2008) 121