J. Cryptol. (2011) 24: 588–613
DOI: 10.1007/s00145-010-9073-y

Journal of
CRYPTOLOGY

Tweakable Block Ciphers

Moses Liskov
Computer Science Department, The College of William and Mary, Williamsburg, VA 23187, USA
mliskov@cs.wm.edu

Ronald L. Rivest
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
rivest@mit.edu

David Wagner
University of California Berkeley, Soda Hall, Berkeley, CA 94720, USA
daw@cs.berkeley.edu

Communicated by Mihir Bellare

Received 28 July 2005
Online publication 2 September 2010

Abstract. A common trend in applications of block ciphers over the past decades has been to employ block ciphers as one piece of a "mode of operation"—possibly, a way to make a secure symmetric-key cryptosystem, but more generally, any cryptographic application. Most of the time, these modes of operation use a wide variety of techniques to achieve a subgoal necessary for their main goal: instantiation of "essentially different" instances of the block cipher.

We formalize a cryptographic primitive, the "*tweakable block cipher*." Such a cipher has not only the usual inputs—message and cryptographic key—but also a third input, the "tweak." The tweak serves much the same purpose that an initialization vector does for CBC mode or that a nonce does for OCB mode. Our abstraction brings this feature down to the primitive block-cipher level, instead of incorporating it only at the higher modes-of-operation levels. We suggest that (1) tweakable block ciphers are easy to design, (2) the extra cost of making a block cipher "tweakable" is small, and (3) it is easier to design and prove the security of applications of block ciphers that need this variability using tweakable block ciphers.

Key words. Block ciphers, Tweakable block ciphers, Initialization vector, Modes of operation, Pseudorandomness.

1. Introduction

A conventional block cipher takes two inputs—a *key* $K \in \{0,1\}^k$ and a *message* (or *plaintext*) $M \in \{0,1\}^n$—and produces a single output—a *ciphertext* $C \in \{0,1\}^n$. The

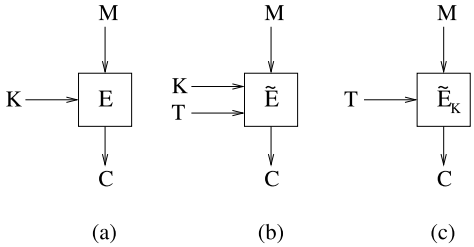

Fig. 1. (**a**) *Standard block cipher* encrypts a message M under control of a key K to yield a ciphertext C. (**b**) *Tweakable block cipher* encrypts a message M under control of not only a key K but also a "tweak" T to yield a ciphertext C. The "tweak" can be changed quickly and can even be public. (**c**) Another way of representing a tweakable block cipher; here the key K shown inside the box.

signature for a block cipher is thus (see Fig. 1(a)):

$$E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n. \tag{1}$$

The corresponding operators for variable-length symmetric-key encryption have a different signature. They take as input a *key* $K \in \{0,1\}^k$, an *initialization vector* (or *nonce*) $V \in \{0,1\}^v$, and a *message* $M \in \{0,1\}^*$ of arbitrary length, and produce as output a *ciphertext* $C \in \{0,1\}^*$. The signature for a typical encryption mode is thus

$$\mathcal{E} : \{0,1\}^k \times \{0,1\}^v \times \{0,1\}^* \to \{0,1\}^*.$$

These operators are usually called "modes of operation" for a block cipher, but this terminology is confusing: these "modes of operation" are actually the encryption schemes (and other primitives) we really care about, and block ciphers are an underlying primitive.

Block ciphers (pseudorandom permutations) are inherently deterministic: every encryption of a given message with a given key will be the same. Many modes of operation and other applications using block ciphers have a requirement for "essentially different" instances of the block cipher in order to prevent attacks that operate by, say, permuting blocks of the input. Attempts to resolve the conflict between keeping the same key for efficiency and yet achieving variability often results in a design that uses a fixed key, but which attempts to achieve variability by manipulating the input before encryption, the output after encryption, or both. Such designs seem inelegant—they are attempting to solve a problem with a primitive (a basic block cipher) that is not well suited for the problem at hand. It would be better to rethink what primitives are really ideal for such a problem.

We propose to revise the notion of a block cipher so that it contains a mechanism for variability as well. The revised primitive, which we call a *tweakable block cipher*, has the signature

$$\widetilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n. \tag{2}$$

For this operator, we call the new (second) input a "tweak" rather than a "nonce" or "initialization vector," but the intent is similar. A tweakable block cipher thus takes

three inputs—a *key* $K \in \{0,1\}^k$, a *tweak* $T \in \{0,1\}^t$, and a *message* (or *plaintext*) $M \in \{0,1\}^n$—and produces as output a *ciphertext* $C \in \{0,1\}^n$ (see Fig. 1(b)).

In designing a tweakable block cipher, we have certain goals. We want any tweakable block ciphers we design to be as efficient as possible. Furthermore, we expect tweaks to be changed frequently, so a tweakable block cipher should have the property that changing the tweak should be efficient. For tweakable block ciphers built from block ciphers, this means that changing the tweak should not make it necessary to rekey the block cipher. And, for any tweakable block cipher, changing the tweak should be less costly than changing the key.[1] Some cryptographic modes of operation such as the Davies–Meyer hash function (see Menezes et al. [28, Sect. 9.4]) have fallen into disfavor because they have a feedback path into the key input of the block cipher. See, for example, the discussion by Rogaway et al. [33] explaining the design rationale for the OCB mode of operation, which uses the same cryptographic key throughout.

A tweakable block cipher should also be secure, meaning that even if an adversary has control of the tweak input, we want the tweakable block cipher to remain secure. We will define what this means more precisely later on. Intuitively, each fixed setting of the tweak should give rise to a different, apparently independent, standard block cipher encryption operator. We wish to carefully distinguish between the function of the *key*, which is to provide uncertainty to the adversary, and the role of the *tweak*, which is to provide variability. The tweak is not intended to provide additional uncertainty to an adversary, and even keeping the tweak secret need not provide any greater cryptographic strength.

The point of this paper is to suggest that by cleanly separating the roles of cryptographic key (which provides uncertainty to the adversary) from that of the tweak (which provides independent variability) we may have just the right tool for many cryptographic purposes.

1.1. *Prior and Related Work*

We are not the first to propose adding an additional input to a block cipher for variability. Rich Schroeppel proposed the Hasty Pudding Cipher (HPC) [35] in the competition for the Advanced Encryption Standard; this cipher utilized a non-key parameter, called the "spice", described as a "secondary key which need not be concealed," and notes that "the spice can be changed very cheaply for each block encrypted." Schroeppel discussed a number of motivations and potential advantages for such an additional input, such as the ability to perform parallel encryption of multiple blocks. To the best of our knowledge, HPC was the first block cipher to utilize such an auxiliary parameter. We note that the security requirements for a tweakable block cipher as proposed in this paper are stronger than the corresponding ones for HPC, since a tweakable block cipher should be secure against a "chosen tweak attack," while Schroeppel cautions that security against a "chosen spice attack" is unknown.

Paul Crowley later proposed the Mercy cipher [13] for disk sector encryption; this cipher includes a 128-bit randomizer or "spice" (he notes Schroeppel's prior work and terminology). Crowley gives as a security goal for Mercy that "any procedure for distinguishing Mercy encryption from a sequence of 2^{128} independent random permutations

[1] More precisely, it is a goal that after computing $\widetilde{E}_K(T, M)$, it should be more efficient to compute $\widetilde{E}_K(T', M')$ than to compute $\widetilde{E}_{K'}(T, M')$.

(for the 2^{128} possible spices) should show no more bias towards correctness than a key guessing attack with the same work factor." His security requirement is thus very close to what is proposed in this paper for a basic tweakable block cipher; our definition of strong security for a tweakable block cipher extends this initial notion by giving the attacker access to a decryption oracle as well as an encryption oracle. Mercy was later broken [17].

On the application side, there are many examples of prior work that have apparent modules for variability in the constructions. The following are some key examples, but not an exhaustive list.

One motivating example for the introduction of tweakable block ciphers is the DESX construction introduced by Rivest (unpublished). The reason for introducing DESX was to cheaply provide additional key information for DES. The security of DESX has been analyzed by Kilian and Rogaway [24]; they show that DESX with n-bit inputs (and tweaks) and k-bit keys has an effective key-length of $k + n - 1 - \lg m$ where the adversary is limited to m oracle calls. In the DESX construction secret pre- and post-whitening values were added as additional key information. In a similar vein, Biham and Biryukov [8] suggest strengthening DES against exhaustive search by (among other things) applying a DESX-like construction to each of DES's S-boxes.

Even and Mansour [16] have also investigated a similar construction where the inner encryption operator F is fixed and public, and encryption is performed by $E_{K_1K_2}(M) = K_2 \oplus F(K_1 \oplus M)$. They show (see also Daemen [14]) that the effective key length here is $n - \lg l - \lg m$ where the adversary is allowed to make l calls to the encryption/decryption oracles and m calls to an oracle for F or F^{-1}.

Similarly, if one looks at the internals of the recently proposed "offset codebook mode" (OCB mode) of Rogaway et al. [33] and the work of Jutla [23], one sees DESX-like modules that may also be viewed as instances of a tweakable block ciphers. That is, the pre- and post-whitening operations are essentially there to provide distinct families of encryption operators, i.e., they are "tweaked." OCB mode was also a motivating example for us because of the complexity and difficulty of its proof of security [33].

Beyond the domain of block cipher design and analysis, it is worth noting the similarity of the tweakable block cipher idea to the idea of salts in the hashing of Unix passwords [30]; each distinct salt makes for an "essentially different" hash for each password.

This work appeared in preliminary form in [26] and [25]. Since the initial publication, the notion of tweakable block ciphers has been applied to disk sector encryption [20–22]. Rogaway [32] describes XE and XEX modes, tweakable block ciphers that are highly efficient if sequential tweaks are used; some subsequent work extends this idea [12,29]. Bellare and Kohno [5] discuss the issue of creating tweakable block ciphers from block ciphers secure against related-key attacks. Black, Cochran, and Shrimpton [9] have presented work analyzing the security of the TCH hash function presented in our preliminary paper, and showing attacks for certain instantiations of the tweakable block cipher. Goldenberg et al. [18] discuss how to add tweaks to Luby–Rackoff block ciphers directly.

1.2. *Roadmap*

In Sect. 2 we then discuss and formalize the notion of security for tweakable block ciphers. In Sect. 3 we suggest several ways of constructing tweakable block ciphers

from existing block ciphers and prove that the existence of tweakable block ciphers is equivalent to the existence of block ciphers. Then in Sect. 4 we suggest several new modes of operation utilizing tweakable block ciphers and give simple proofs for some of them. Section 5 concludes with some discussion and open problems.

2. Definitions

2.1. *Notation*

When A is a randomized algorithm, we write $y \leftarrow A(x)$ to mean that y is a random variable with value determined by A on input x. When S is a finite set, we write $x \leftarrow S$ to mean that x is a random variable chosen according to the uniform distribution on S.

We write $x_1 \leftarrow X_1; x_2 \leftarrow X_2; \ldots; x_k \leftarrow X_k$ to mean that the values of variables $x_1, \ldots, x_k$ are chosen in order, by first determining x_1, then x_2, and so on, until x_k is determined. We write $x_1 \leftarrow X_1; \ldots; x_k \leftarrow X_k : P(x_1, \ldots, x_k)$ to mean the event that $P(x_1, \ldots, x_k)$ evaluates to true, given the random choices of $x_1, \ldots, x_k$.

We write $A^{\mathcal{O}_1,\ldots,\mathcal{O}_k}$ to mean the algorithm A with access to oracles $\mathcal{O}_1, \ldots, \mathcal{O}_k$. We write $u \circ v$ to mean the concatenation of strings u and v.

2.2. *Block Ciphers*

A block cipher is a pair of functions (E, D). E takes as input a k-bit key K and an n-bit message M and outputs an n-bit ciphertext C. D takes a k-bit key K and an n-bit ciphertext C as inputs, and outputs an n-bit plaintext M. E and D must both be efficient (polynomial time) to compute, deterministic, and such that for all K and all M, $D(K, E(K, M)) = M$. For convenience, we write E_K to mean the function that takes M to $E(K, M)$.

The security of a block cipher E (e.g., parameterized as in (1)) can be quantified as $\mathsf{ADV}_{\mathsf{PRP}}(E, q, s)$, which represents the maximum advantage that an adversary can obtain when trying to distinguish E_K (with a randomly chosen key K) from a random permutation Π with the same domain, when allowed q queries to an unknown oracle (which is either E_K or Π) and when allowed computation time s. This advantage is defined as the difference between the probability the adversary outputs 1 when given oracle access to E_K and the probability the same adversary outputs 1 when given oracle access to Π.

Definition 1. Define

$$\mathsf{ADV}_{\mathsf{PRP}}(E, A) = \left| \Pr\left[b \leftarrow A^{\Pi} : b = 1\right] - \Pr\left[K \leftarrow \{0,1\}^k; b \leftarrow A^{E_K} : b = 1\right] \right|,$$

and define

$$\mathsf{ADV}_{\mathsf{PRP}}(E, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathsf{ADV}_{\mathsf{PRP}}(E, A),$$

where $\mathcal{A}_{q,s}$ is the set of all algorithms making at most q oracle queries and running in time at most s.

A stronger definition for a block cipher can be defined as the maximum advantage that an adversary can obtain when trying to distinguish the pair of oracles $E(K,\cdot), D(K,\cdot)$ from the pair Π, Π^{-1}, when allowed q queries and computation time s. This advantage is defined as the difference between the probability the adversary outputs 1 when given oracle access to E, D and the probability the same adversary outputs 1 when given oracle access to Π, Π^{-1}.

Definition 2. Define

$$\mathsf{ADV}_{\mathsf{SPRP}}(E, A) = \big|\Pr\big[b \leftarrow A^{\Pi,\Pi^{-1}} : b = 1\big] - \Pr\big[K \leftarrow \{0,1\}^k; b \leftarrow A^{E_K, D_K} : b = 1\big]\big|,$$

and define

$$\mathsf{ADV}_{\mathsf{SPRP}}(E, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathsf{ADV}_{\mathsf{SPRP}}(E, A),$$

where $\mathcal{A}_{q,s}$ is as in Definition 1.

Such a block cipher is referred to as a "strong block cipher" or a "strong pseudorandom permutation."

2.3. *Tweakable Block Ciphers*

A tweakable block cipher is a pair of functions $(\widetilde{E}, \widetilde{D})$. $\widetilde{E}$ takes as input a k-bit key K, a t-bit tweak T, and an n-bit message M and outputs an n-bit ciphertext C. $\widetilde{D}$ takes a k-bit key K, a t-bit tweak T, and an n-bit ciphertext C as inputs and outputs an n-bit plaintext M. $\widetilde{E}$ and $\widetilde{D}$ must both be efficient (polynomial time) to compute, deterministic, and such that for all K, T, and M, $\widetilde{D}(K, T, \widetilde{E}(K, T, M)) = M$. For convenience, we write $\widetilde{E}_K$ to mean the function that takes (T, M) to $\widetilde{E}(K, T, M)$.

We may measure the security of a tweakable block cipher (or a tweakable pseudorandom permutation) $\widetilde{E}$ as the maximum advantage $\mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s)$ an adversary can obtain when trying to distinguish $\widetilde{E}_K$ from a "tweakable random permutation" $\widetilde{\Pi}$, a family of independent random permutations parameterized by T. That is, for each T, we have that $\widetilde{\Pi}(T,\cdot)$ is an independent randomly chosen permutation of the message space. A tweakable block cipher $\widetilde{E}$ may be considered secure when $\mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s)$ is sufficiently small.

Definition 3. Define

$$\mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, A) = \big|\Pr\big[b \leftarrow A^{\widetilde{\Pi}} : b = 1\big] - \Pr\big[K \leftarrow \{0,1\}^k; b \leftarrow A^{\widetilde{E}_K} : b = 1\big]\big|,$$

and define

$$\mathsf{ADV}_{\mathsf{TPRP}}(E, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathsf{ADV}_{\mathsf{TPRP}}(E, A),$$

where $\mathcal{A}_{q,s}$ is as in Definition 1.

Note that the adversary is allowed to choose both the message and tweak for each oracle call.

Similarly, we define $\mathbf{ADV}_{\mathbf{STPRP}}(\widetilde{E}, q, s)$ as the maximum advantage an adversary can obtain when trying to distinguish the pair of oracles $\widetilde{E}_K, \widetilde{D}_K$ from $\widetilde{\Pi}, \widetilde{\Pi}^{-1}$, when given q queries and s time. We say that a tweakable block cipher is a strong tweakable block cipher when $\mathbf{ADV}_{\mathbf{STPRP}}(\widetilde{E}, q, s)$ is sufficiently small.

Definition 4. Define

$$\mathbf{ADV}_{\mathbf{STPRP}}(\widetilde{E}, A) = \big|\Pr\big[b \leftarrow A^{\widetilde{\Pi}, \widetilde{\Pi}^{-1}} : b = 1\big] - \Pr\big[K \leftarrow \{0,1\}^k; b \leftarrow A^{\widetilde{E}_K, \widetilde{D}_K} : b = 1\big]\big|,$$

and define

$$\mathbf{ADV}_{\mathbf{STPRP}}(E, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathbf{ADV}_{\mathbf{STPRP}}(E, A),$$

where $\mathcal{A}_{q,s}$ is as in Definition 1.

3. Constructions

In this section, we show how to construct a secure tweakable block cipher and a strong tweakable block cipher, from simple underlying primitives we assume to be secure.

3.1. *Unsuccessful Constructions*

First, we demonstrate some simple attempts at constructing tweakable block ciphers from block ciphers and show that these methods are not successful.

As a first attempt, we might try applying the DESX construction

$$\widetilde{E}_K(T, M) = E_K(M \oplus T) \oplus T.$$

Here, an adversary making the two queries (T, M), $(T \oplus 1, M \oplus 1)$ can distinguish. If C and C' are the respective outputs, $C \oplus C' = 1$ will always be true for $\widetilde{E}$, but this is very unlikely for $\widetilde{\Pi}$.

Another natural idea is to make the tweak affect the key of the underlying block cipher. This is doubly wrong, as this clearly any change to the tweak will require rekeying, and furthermore, these approaches are not necessarily secure.

Consider

$$\widetilde{E}_K(T, M) = E_{K \circ T}(M),$$

$$\widetilde{E'}_K(T, M) = E_{K \oplus T}(M).$$

If secure block ciphers exist, it is trivial to construct a secure block cipher such that $E_K(M) = E_{K \oplus 1}(M)$ for all K, M.[2] If such a block cipher were used in either of the

[2] For example, $E_{K \circ b}(M) = E'_K(M)$, where E' is a secure block cipher.

above constructions, the resulting tweakable block cipher would be easily attackable, by querying (T, M) and $(T \oplus 1, M)$. Even if this type of overt flaw is not known to exist in E, related-key attacks [7] could apply.

3.2. *A Tweakable Block Cipher*

Here, we give a first secure tweakable block cipher construction. We call this construction CMT mode, which stands for "CBC-MAC Tweaked."

Let E be a secure block cipher. CMT mode is defined by

$$\widetilde{E}_K(T, M) = E_K\big(T \oplus E_K(M)\big)$$

and

$$\widetilde{D}_K(T, C) = D_K\big(T \oplus D_K(C)\big).$$

CMT mode is reasonably efficient. The following theorem proves its security.

Theorem 1. *$\widetilde{E}$ is a secure tweakable block cipher. Specifically,*

$$\mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) \le \mathsf{ADV}_{\mathsf{PRP}}(E, 2q, s+q) + \frac{17q^2 - q}{2^{n+1}}.$$

Proof. This proof is adapted from a proof given by Bellare and Kohno [4].

Note that $\widetilde{E}(T, M)$ is just the CBC MAC of the message $M \circ T$. Let R be a random function from $t + n$ bits to n bits. Define

$$\mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, A) = \big|\Pr\big[b \leftarrow A^R : b = 1\big] - \Pr\big[K \leftarrow \{0,1\}^k; b \leftarrow A^{\widetilde{E}_K} : b = 1\big]\big|,$$

and let $\mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, A)$.

Bellare et al. [3] show that if A is an adversary that aims to distinguish a CBC MAC oracle from a random oracle, an adversary A' can be constructed that aims to distinguish the block cipher from a random permutation, such that

$$\mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, A) \le \mathsf{ADV}_{\mathsf{PRP}}(E, A') + \frac{4q^2}{2^{n-1}},$$

where q is the number of queries made by A, where A' makes $2q$ queries, and where A' takes time $s + q$ where s is the running time of A. The time overhead is due to the need to make twice as many oracle queries. This proves that $\mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, q, s) \le \mathsf{ADV}_{\mathsf{PRP}}(E, 2q, s+q) + \frac{16q^2}{2^{n+1}}$.

Note that R differs from $\widetilde{\Pi}$ only in that for any fixed T, R is a random function, while Π is a random permutation. The output of a single random function and a single random permutation, each with n-bit outputs, are statistically close: if q queries are made, the statistical difference is at most $\frac{q^2-q}{2^{n+1}}$. Thus, the outputs of R and Π have statistical difference at most $\frac{1}{2^{n+1}} \sum_{i=1}^{l} q_i^2 - q_i$, where l is the number of distinct tweaks

queried, and where q_i is the number of queries involving the ith tweak. This difference is maximized when $l = 1$ and $q_1 = q$. Therefore,

$$\begin{aligned}\mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) &\le \mathsf{ADV}_{\mathsf{PRF}}(\widetilde{E}, q, t) + \frac{q^2 - q}{2^{n+1}} \\ &\le \mathsf{ADV}_{\mathsf{PRP}}(E, 2q, s) + \frac{17q^2 - q}{2^{n+1}},\end{aligned}$$

as stated. □

Theorem 2. *The existence of secure block ciphers is equivalent to the existence of secure tweakable block ciphers.*

Proof of Theorem 2. One direction follows from Theorem 1.

For the other, let $\widetilde{E}_K(T, M)$ be a tweakable block cipher, and let $E_K(M) = \widetilde{E}_K(0^t, M)$. Any distinguishing attack against E is clearly a distinguishing attack against $\widetilde{E}$ in which every tweak is chosen to be 0^t. Therefore, $\mathsf{ADV}_{\mathsf{PRP}}(E, q, s) = \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s)$ for all q and s. □

We leave it as an open problem to devise a construction with a tighter bound than Theorem 1.

Efficiency The construction of Theorem 1 has an overall running time that is twice the running time of the underlying block cipher.

For this construction, it can be significantly faster to change the tweak than to change the key. If $\widetilde{E}_K(T, M)$ has been computed, the cost to compute $\widetilde{E}_K(T', M')$ is the cost of two block cipher computations. The cost of computing $\widetilde{E}_{K'}(T, M')$, however, is the cost of two block cipher computations plus the cost of key scheduling for the block cipher.

3.3. *A Strong Tweakable Block Cipher*

The construction of Sect. 3.2 is interesting as a simple example of a tweakable block cipher. However, it is not a strong tweakable block cipher, and its efficiency leaves room for improvement. Next we give a strong tweakable block cipher that is also more efficient. We call this construction LRW mode, after a later precedent in the literature [29].

A family $\mathcal{H}$ of functions with signature $\{0,1\}^t \to \{0,1\}^n$ is said to be an *ϵ-almost 2-xor-universal* hash function family ("ϵ-AXU_2 hash function family", for short) if $\Pr_h[h(x) \oplus h(y) = z] \le \epsilon$ holds for all x, y, z with $x \ne y$, where the probability is taken over h chosen uniformly at random from $\mathcal{H}$.

Let $\mathcal{H}$ be such a hash function family. The LRW mode tweakable block cipher uses a key (K, h) where $K \leftarrow \{0,1\}^k$ and $h \leftarrow \mathcal{H}$, and is given by

$$\widetilde{E}_{K,h}(T, M) = E_K(M \oplus h(T)) \oplus h(T),$$

$$\widetilde{D}_{K,h}(T, C) = D_K(M \oplus h(T)) \oplus h(T).$$

Theorem 3. *If $\mathcal{H}$ is an ϵ-AXU_2 hash function family with $\epsilon \geq 2^{-n}$, and E is a strong pseudorandom permutation, then $\widetilde{E}$ is a strong tweakable block cipher. Specifically, for all $q \leq 2^{n-1}$,*

$$\mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, q, s) \leq \mathsf{ADV}_{\mathsf{SPRP}}(E, q, s+q) + 3\epsilon q^2.$$

First, we establish some notation. Let $\widetilde{E}'_h$ be Π used in LRW mode. We use $\Pr_0[\cdot]$ to represent the probability measure in the case where A interacts with $\widetilde{E}_{K,h}$, where the probability is taken over the choice of $K \in \{0,1\}^k$ and $h \in \mathcal{H}$ uniformly and independently at random. Also, we let $\Pr_1[\cdot]$ denote the measure where A interacts with $\widetilde{E}'_h$, where the probability is taken over the uniform random choice of $h \in \mathcal{H}$. Let $\Pr_2[\cdot]$ denote the measure when A interacts with $\widetilde{\Pi}$. In all three cases, we write $\mathcal{O}$ for A's oracle.

We let the random variable T_i denote the tweak input on A's ith oracle call, and we let M_i and C_i denote the plaintext and ciphertext corresponding to this call, so that $\mathcal{O}(T_i, M_i) = C_i$. In other words, if A's ith oracle query is an encryption query (to $\mathcal{O}$), then (T_i, M_i) denotes the input and C_i the return value, whereas if A's ith oracle query is a decryption query (to $\mathcal{O}^{-1}$), then the input is (T_i, C_i), and the result of the query is M_i. Moreover, we define the random variables N_i, B_i by $N_i = M_i \oplus h(T_i)$ and $B_i = C_i \oplus h(T_i)$. Note that in probability measure $\Pr_0[\cdot]$, we have $E_K(N_i) = B_i$, and in probability measure $\Pr_1[\cdot]$, we have $\Pi(N_i) = B_i$. We define the random variable τ_n by $\tau_n = \langle (T_1, M_1, C_1), \ldots, (T_n, M_n, C_n) \rangle$, and we use $\tau = \tau_q$ to represent the full transcript of interaction with the oracle.

We fix an adversary A, and we assume without loss of generality that A does not make any repeated or redundant queries to its oracle. As a consequence of this assumption, the pairs (T_i, M_i) are all distinct, or in other words, for all $i \neq j$, we have $(T_i, M_i) \neq (T_j, M_j)$. Similarly, the pairs (T_i, C_i) are also distinct, as are the (T_i, N_i)'s and the (T_i, B_i)'s. Also, the output of A can be viewed as a function of the transcript τ, so we sometimes write the output of A as $A(\tau)$.

Our proof is separated into two parts. In the information-theoretic part, we let Π denote a permutation chosen uniformly at random and show that $\widetilde{E}'_h$ is a secure tweakable block cipher. Then, in the computational part, we let E be arbitrary, and we show that if E_K and Π are computationally indistinguishable, then $\widetilde{E}$ will also be a secure tweakable block cipher.

The information-theoretic part of the proof uses the following strategy. We define a bad event Bad. We show that when conditioning on the complement event (that is, when nothing bad happens), the probability measures $\Pr_1[\cdot|\overline{\mathsf{Bad}}]$ and $\Pr_2[\cdot|\overline{\mathsf{Bad}}]$ are in fact identical, that is, the adversary can only distinguish with probability 1/2. Then, we show that $\Pr_1[\mathsf{Bad}]$ and $\Pr_2[\mathsf{Bad}]$ are both small; thus, if we replace E with a random permutation, the adversary could only distinguish with a negligible advantage.

In our arguments, we define Bad_n to be the event that, for some $1 \leq i < j \leq n$, either $N_i = N_j$ or $B_i = B_j$. Also, we let $\mathsf{Bad} = \mathsf{Bad}_q$.

Lemma 1. *Let Ev_τ be the event that τ is the transcript generated. Then* $\Pr_1[\mathsf{Ev}_\tau|\overline{\mathsf{Bad}}] = \Pr_2[\mathsf{Ev}_\tau|\overline{\mathsf{Bad}}]$.

Proof. We show this by induction on the length of the transcript, q. Consider the qth oracle query: it is either an encryption or a decryption query. Suppose first that the qth oracle query is an encryption query, with inputs (T_q, M_q). By the inductive hypothesis, we can assume that the distribution of τ_{q-1} is the same for both the $\Pr_1[\cdot|\overline{\mathsf{Bad}}_{q-1}]$ and $\Pr_2[\cdot|\overline{\mathsf{Bad}}_{q-1}]$ probability measures, hence the same is true of the distribution of (τ_{q-1}, T_q, M_q).

Now fix any h such that $N_q \notin \{N_1, \ldots, N_{q-1}\}$, so that the only remaining random choice is over Π or $\widetilde{\Pi}$. When $\mathcal{O} = \widetilde{\Pi}$, $C_q = \widetilde{\Pi}(T_q, M_q)$ is uniformly distributed on the set $S = \{0,1\}^n \setminus \{C_i : T_i = T_q \text{ and } 1 \le i < q\}$ if we condition on $\overline{\mathsf{Bad}}_{q-1}$ (but before conditioning on $\overline{\mathsf{Bad}}_q$). When $\mathcal{O} = \widetilde{E}'$, we find something slightly different: $B_q = \Pi(N_q)$ is uniformly distributed on the set $\{0,1\}^n \setminus \{B_1, \ldots, B_{q-1}\}$ (conditioned on $\overline{\mathsf{Bad}}_{q-1}$, but before conditioning on $\overline{\mathsf{Bad}}_q$), hence $C_q = B_q \oplus h(T_q)$ is uniformly distributed on the set $S' = \{0,1\}^n \setminus \{C_i \oplus h(T_i) \oplus h(T_q) : i = 1, \ldots, q-1\}$. In both cases, the probabilities are independent of the choice of h. Also, note that $S' \subseteq S$, since when $T_i = T_q$, we have $C_i \oplus h(T_i) \oplus h(T_q) = C_i$. Adding the condition $\overline{\mathsf{Bad}}_q$ amounts to adding the condition that $B_q \in \{0,1\}^n \setminus \{B_1, \ldots, B_{q-1}\}$, i.e., that $C_q \in S'$. Thus, after conditioning on $\overline{\mathsf{Bad}}_q$, we see that C_q is uniformly distributed on S' and independent of the rest of the transcript, and hence the distribution of τ is the same for both the $\Pr_1[\cdot|\overline{\mathsf{Bad}}_q]$ and $\Pr_2[\cdot|\overline{\mathsf{Bad}}_q]$ probability measures. (Here we have used the following simple fact: if the random variable X is uniform on a set S and if S' is some subset of S, then after conditioning on the event $X \in S'$ we find that the resulting random variable is uniform on S'.)

This covers the case where the qth query is a chosen-plaintext query. The other case, where the qth query is a chosen-ciphertext query, is treated similarly. This concludes the proof of Lemma 1. □

Lemma 2. *If $\mathcal{H}$ is ϵ-AXU_2, then* $\Pr_2[\mathsf{Bad}_q] \le \epsilon q(q-1)$.

Proof. Note that, when $\mathcal{O} = \widetilde{\Pi}$, h is independent of the transcript τ. Hence, we can defer the choice of h until after A completes all q of its queries and the values of T_i, M_i, C_i are fixed. Then, we find

$$\Pr_2[\mathsf{Bad}_q] = \Pr_h[\exists i < j : N_i = N_j \vee B_i = B_j] \tag{3}$$

$$\le \sum_{1 \le i < j \le q} \Pr_h[N_i = N_j] + \Pr_h[B_i = B_j] \tag{4}$$

$$= \sum_{1 \le i < j \le q} \Pr_h\big[h(T_i) \oplus h(T_j) = M_i \oplus M_j\big] + \Pr_h\big[h(T_i) \oplus h(T_j) = C_i \oplus C_j\big] \tag{5}$$

$$\le \sum_{1 \le i < j \le q} 2\epsilon = \epsilon q(q-1). \tag{6}$$

Equation (3) is from the definition of Bad_q. Equation (4) follows by a union bound. Equation (5) follows from the definition of N_i and B_i. Finally, (6) follows by the defin-

ition of ϵ-AXU$_2$ hash functions. Note that if $T_i = T_j$ and $i \neq j$, then $M_i \oplus M_j \neq 0$, and thus $\Pr[h(T_i) \oplus h(T_j) = M_i \oplus M_j] = 0 \leq \epsilon$; similarly for $C_i \oplus C_j \neq 0$. □

Lemma 3. *If $\mathcal{H}$ is ϵ-AXU$_2$ for $\epsilon \geq 1/2^n$ and if $E_K = \Pi$ and $q \leq 2^{n-1}$, then* $\Pr_1[\mathsf{Bad}_q] \leq 1.5\epsilon q(q-1)$.

Proof. We will prove that $\Pr_1[\mathsf{Bad}_q] \leq 1.5\epsilon q(q-1)$ by induction on q. Let Ev denote that event that, for some i, we have $N_i = N_q$, and let Ev′ denote the event that, for some i, we have $B_i = B_q$. Note that $\Pr_1[\mathsf{Bad}_q] = \Pr_1[\mathsf{Bad}_{q-1}] + \Pr_1[\mathsf{Bad}_q | \overline{\mathsf{Bad}}_{q-1}] \Pr_1[\overline{\mathsf{Bad}}_{q-1}]$. By the inductive hypothesis, $\Pr_1[\mathsf{Bad}_{q-1}] \leq 1.5\epsilon(q-1)(q-2)$. Also, $\Pr_1[\overline{\mathsf{Bad}}_{q-1}] \leq 1$. Hence all that remains is to bound the term $\Pr_1[\mathsf{Bad}_q | \overline{\mathsf{Bad}}_{q-1}]$.

Applying a union bound shows that $\Pr_1[\mathsf{Bad}_q | \overline{\mathsf{Bad}}_{q-1}] \leq \Pr_1[\mathsf{Ev} | \overline{\mathsf{Bad}}_{q-1}] + \Pr_1[\mathsf{Ev}' | \overline{\mathsf{Ev}} \wedge \overline{\mathsf{Bad}}_{q-1}]$. We next bound each of these two terms in turn. By Lemma 1, and since $\mathcal{H}$ is ϵ-AXU$_2$, we see $\Pr_1[\mathsf{Ev} | \overline{\mathsf{Bad}}_{q-1}] = \Pr_2[\mathsf{Ev} | \overline{\mathsf{Bad}}_{q-1}] \leq \epsilon(q-1)$. Moreover, $\Pr_1[\mathsf{Ev}' | \overline{\mathsf{Ev}} \wedge \overline{\mathsf{Bad}}_{q-1}] = \Pr_1[\Pi(N_q) \in \{B_1, \ldots, B_{q-1}\} | \overline{\mathsf{Ev}} \wedge \overline{\mathsf{Bad}}_{q-1}] \leq (q-1)/(2^n - q + 1) \leq 2(q-1)/2^n \leq 2\epsilon(q-1)$, since $\Pi(N_q)$ is uniformly distributed on a set of size at least $2^n - q + 1$ and since $\epsilon \geq 1/2^n$. Finally, $1.5\epsilon(q-1)(q-2) + \epsilon(q-1) + 2\epsilon(q-1) \leq 1.5\epsilon q(q-1)$. The statement of the lemma now follows. □

We are now ready to prove the security theorem.

Proof of Theorem 3. First, we show that $\mathsf{ADV}^{\mathsf{STPRP}}(\widetilde{E}', q, s) \leq 3\epsilon q(q-1)$. For notational convenience, define $p_1 = \Pr_1[A(\tau) = 1 | \overline{\mathsf{Bad}}]$, $p_1' = \Pr_1[A(\tau) = 1 | \mathsf{Bad}]$, $p_2 = \Pr_2[A(\tau) = 1 | \overline{\mathsf{Bad}}]$, and $p_2' = \Pr_2[A(\tau) = 1 | \mathsf{Bad}]$.

We perform the following calculation:

$$
\begin{aligned}
\mathsf{ADV}^{\mathsf{STPRP}}(\widetilde{E}', A) &= \left| \Pr_1\left[A(\tau) = 1\right] - \Pr_2[A(\tau) = 1] \right| \\
&= \left| p_1 \Pr_1[\overline{\mathsf{Bad}}] + p_1' \Pr_1[\mathsf{Bad}] - p_2 \Pr_1[\overline{\mathsf{Bad}}] - p_2' \Pr_2[\mathsf{Bad}] \right| \\
&\leq \left| p_1 \Pr_1[\overline{\mathsf{Bad}}] - p_2 \Pr_2[\overline{\mathsf{Bad}}] \right| + \left| p_1' \Pr_1[\mathsf{Bad}] - p_2' \Pr_2[\mathsf{Bad}] \right| \quad (7) \\
&\leq \left| p_1 \Pr_1[\overline{\mathsf{Bad}}] - p_2 \Pr_2[\overline{\mathsf{Bad}}] \right| + 1.5\epsilon q(q-1) \quad (8) \\
&\leq \Pr[A(\tau) = 1 | \overline{\mathsf{Bad}}] \left| \Pr_1[\overline{\mathsf{Bad}}] - \Pr_2[\overline{\mathsf{Bad}}] \right| + 1.5\epsilon q(q-1) \quad (9) \\
&\leq \left| \Pr_2[\mathsf{Bad}] - \Pr_1[\mathsf{Bad}] \right| + 1.5\epsilon q(q-1) \quad (10) \\
&\leq 3\epsilon q(q-1). \quad (11)
\end{aligned}
$$

Line (7) follows by the triangle inequality. Lemmas 2 and 3 show that $\Pr[\mathsf{Bad}] \leq 1.5\epsilon q(q-1)$, which justifies (8). Lemma 1 establishes that $p_1 = p_2 = \Pr[A(\tau) = 1 | \overline{\mathsf{Bad}}]$, which justifies (9). We then note that $|\Pr_1[\overline{\mathsf{Bad}}] - \Pr_2[\overline{\mathsf{Bad}}]| = |\Pr_2[\mathsf{Bad}] - \Pr_1[\mathsf{Bad}]| \leq 1.5\epsilon q(q-1)$, to arrive at (10) and then (11). Since the adversary A was arbitrary, it follows that $\mathsf{ADV}^{\mathsf{STPRP}}(\widetilde{E}', q, s) \leq 3\epsilon q(q-1)$.

If A is an adversary attacking $\widetilde{E}$, let A' be the adversary attacking E as follows. A' runs A in its attack but uses its own oracle in LRW mode to supply answers to A's

oracle queries. A' outputs what A outputs. Note that if A makes q queries and runs in s time, then A' makes q queries and runs in $s+q$ time (the additional q accounts for the hash computations). Note that $\mathsf{ADV}_{\mathsf{SPRP}}(E, A') = |\Pr_0[A(\tau)=1] - \Pr_1[A(\tau)=1]|$, and therefore $\max_{A\in\mathcal{A}_{q,s}} |\Pr_0[A(\tau)=1] - \Pr_1[A(\tau)=1]| \le \mathsf{ADV}_{\mathsf{SPRP}}(E, q, s+q)$.

The result then follows:

$$\begin{aligned}
\mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, q, s) &= \max_{A\in\mathcal{A}_{q,s}} \left|\Pr_0\big[A(\tau)=1\big] - \Pr_2\big[A(\tau)=1\big]\right| \\
&\le \max_{A\in\mathcal{A}_{q,s}} \left|\Pr_0\big[A(\tau)=1\big] - \Pr_1\big[A(\tau)=1\big]\right| \\
&\quad + \left|\Pr_1\big[A(\tau)=1\big] - \Pr_2\big[A(\tau)=1\big]\right| \\
&\le \max_{A\in\mathcal{A}_{q,s}} \left|\Pr_0\big[A(\tau)=1\big] - \Pr_1\big[A(\tau)=1\big]\right| \\
&\quad + \max_{A\in\mathcal{A}_{q,s}} \left|\Pr_1\big[A(\tau)=1\big] - \Pr_2\big[A(\tau)=1\big]\right| \\
&= \mathsf{ADV}_{\mathsf{SPRP}}(E, q, s+q) + \mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}', q, s) \\
&\le \mathsf{ADV}_{\mathsf{SPRP}}(E, q, s+q) + 3\epsilon q(q-1).
\end{aligned}$$

This completes the proof. □

As there are plenty of known constructions of ϵ-AXU_2 hash families with $\epsilon \approx 1/2^n$, the security theorem shows that we can obtain a construction with good security against adaptive chosen-ciphertext attacks for up to the birthday bound, i.e., for $q \ll 2^{n/2}$.

Efficiency It is harder to estimate the efficiency of the LRW construction, because it is not clear how to compare the computational cost of the block cipher to the cost of the hash functions. However, we expect that LRW mode will be faster than CMT mode in practice. For instance, for $t = n = 128$, a generalized division hash runs in 300 cycles [36], UMAC/UHASH runs in 200 cycles [10], hash127 runs in 150 cycles [6], and a DFC-style decorrelation module should run in about 200 cycles [19].[3] If we compare the best of these to AES, which runs in 237–333 cycles [2], we expect that a version of AES tweaked in this way will run about 45–63% slower than the plain AES. These estimates are conservative: the above hash functions are designed for large input strings, whereas no compression is necessary for this application. Indeed, subsequent work of Rogaway [32] shows that even better performance can be achieved with a more carefully designed hash function.

In this construction, changing only the key requires that we rekey the cipher and compute the block cipher. Changing only the tweak requires that we recompute the hash function and the block cipher, but no rekeying is necessary. We expect that changing the tweak will generally be faster than changing the key. For AES, key expansion runs in

[3] All estimates are based on a Pentium II class machine and are rough estimates based on speed benchmarks that are not directly comparable.

277–374 cycles [15]. Based on our estimates of the cost of hashing, changing the tweak should be faster by about 23–32%.[4]

Though LRW mode is likely to be faster than CMT mode, LRW mode does require a longer key. However, if a pseudorandom generator can be agreed upon in advance, we can simply use the key of length k as an input to the pseudorandom generator which will enable us to get a longer pseudorandom key that can be used to generate both K and h. This negatively impacts performance, but only when the key of the tweakable block cipher is changed.

3.4. *Related Work*

There are similarities between our constructions and various results in the literature.

As is discussed in the proof, the CMT construction is the CBC MAC of the two-block message $M \circ T$.

The LRW construction bears a similarity to the Luby–Rackoff construction [27] if we think of E_K and h as different pseudorandom functions. Even more relevant is the recent work of Naor and Reingold [31], who showed that we can relax the Luby–Rackoff construction by using merely pairwise-independent functions in the first and last rounds and by using a single pseudorandom function (instead of two) in the intervening rounds (in the four-round Luby–Rackoff construction of strong PRPs). We can think of LRW mode as doing this on a message block made up of M and T together, except using a single middle round, but with a pseudorandom permutation instead of a pseudorandom function, and revealing only the second half of the output.

It should be noted that in both of these cases what we do is actually slightly different, but they bear enough of a strong resemblance to these prior results that it is worth mentioning the relationship.

In fact, it may be possible to view LRW mode as a strengthening of Naor and Reingold's result. In a sense we modify the Feistel-type construction by outputting only one side of it, which we assume is accompanied by T, and yet show that this still gives a pseudorandom permutation.

Both our constructions use an underlying block cipher in a black-box way. Thus, in a sense, we are providing a mechanism to take a random permutation oracle and create a pseudorandom permutation family oracle. In a way this is like getting many permutation oracles for the price of one: we simply think of the one permutation oracle as a random permutation and apply a tweakable block cipher general construction, and we get a whole family of pseudorandom permutations. This has been studied previously; for instance, Black and Rogaway [11] prove that having two independent random permutation oracles is indistinguishable from having $\Pi(\cdot)$ and $\Pi(K \oplus \cdot)$ where K is a secret value. Our work can be seen as an improvement that provides a whole family of independent random permutations from a single one.

4. Tweakable Modes of Operation

We claim that it is easier to design and prove the security of applications using the tweakable block cipher abstraction. In evidence of this, we present three tweakable

[4] Specifically, 23% for 192-bit keys (150 + 286 vs. 277 + 286 cycles), 29% for 128-bit keys (150 + 237 vs. 305 + 237 cycles), and 32% for 256-bit keys (150 + 333 vs. 374 + 333 cycles).

modes of operation, with proofs of security for each. In all three cases, we do not suggest that the tweakable mode is superior to existing modes of operation. We stress that these modes should be taken primarily as a demonstration of the conceptual advantages gained by working with tweakable block ciphers.

However, our third tweakable mode is of independent interest as it is a generalization of OCB mode [33] and because of its very tight reduction, relative to the tweakable block cipher it uses.

4.1. *Tweak Chaining (TC)*

Tweak block chaining (TC) is a symmetric encryption mode modeled after cipher block chaining (CBC). An initial tweak T_0 plays the role of the initialization vector (IV) for CBC. Each successive message block M_i is encrypted under the encryption key K and a tweak T_{i-1}, where $T_i = C_i$ for $i > 0$. The final ciphertext is $(T_0, C_1 \circ \cdots \circ C_l)$. See Fig. 2.

To decrypt, we compute $M_1 = \widetilde{D}_K(T_0, C_1)$ and $M_i = \widetilde{D}_K(C_{i-1}, C_i)$ for $i > 1$.

To handle messages whose length is greater than n but not a multiple of n, a variant of ciphertext-stealing [34] can be used; see Fig. 3. One can also adapt the TC construction to make a TC-MAC in the same manner that one can use the CBC construction to make a CBC-MAC, though these constructions still need a security analysis.

Chosen-plaintext security for a symmetric encryption scheme is defined in terms of the maximum advantage an adversary can achieve in distinguishing one of two mes-

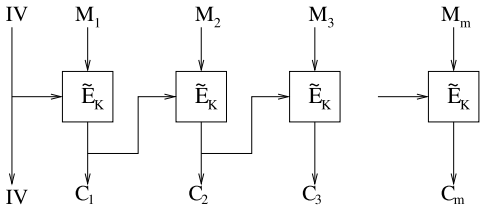

Fig. 2. Tweak chaining: a symmetric encryption mode for a tweakable block cipher. Each ciphertext becomes the tweak for the next encryption. Note that to decipher a block, one needs only the key, C_i, and C_{i-1}, so decryption can be done in parallel.

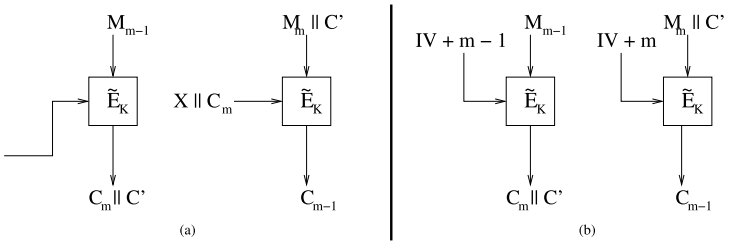

Fig. 3. (**a**) Ciphertext stealing for tweak chaining handles messages whose length is at least n bits long but not a multiple of n. Let r denote the length of the last (short) block M_m of the message. Then $|C_m| = |M_m| = r$ and $|C'| = n - r$. Here X denotes the rightmost $n - r$ bits of C_{m-2} (or of T_0 if $m = 2$). (**b**) A similar technique is used to handle similar messages in tweak incrementation encryption mode.

sages, under a chosen plaintext attack. Formally, define

$$\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, A) = \Pr\big[K \leftarrow \{0,1\}^k; (m_0, m_1) \leftarrow A^{E_K}; b \leftarrow \{0,1\}; \\ b' \leftarrow A^{\mathcal{E}_K}\big(\mathcal{E}_K(m_b)\big) : b' = b \text{ and } |m_0| = |m_1|\big] - \frac{1}{2}$$

and define

$$\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) = \max_{A \in \mathcal{A}_{q,s}} \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, A),$$

where $\mathcal{A}_{q,s}$ is the set of all algorithms A that make queries with at most q blocks of input and run in at most s time.

Theorem 4. *If $\mathcal{E}$ is the symmetric encryption system made from $\widetilde{E}$ in TC mode, then*

$$\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \le \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) + \frac{q^2 - q}{2^{n+1}}.$$

Proof. Conceptually, the proof is very simple. If $\mathcal{E}$ were implemented with a random permutation family instead of $\widetilde{E}$, then the adversary would be unable to get any advantage unless some tweak is used more than once, which happens with probability $\frac{q^2-q}{2^{n+1}}$. Thus, the maximum advantage is bounded by this probability plus the maximum advantage in distinguishing $\widetilde{E}$ from $\widetilde{\Pi}$.

Let $\mathcal{E}'$ be the symmetric encryption system made from $\widetilde{\Pi}$ in TC mode. Then

$$\begin{aligned}
\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) &= \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) + \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \\
&\quad - \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) \\
&\le \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) + \big|\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \\
&\quad - \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s)\big| \\
&\le \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) + \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s).
\end{aligned}$$

This last step is due to the natural "simulating" reduction. Namely, if A is an adversary attacking the IND-CPA property of its target, we can construct an A' that attempts to distinguish $\widetilde{E}$ from $\widetilde{\Pi}$. A' runs A in its attack and answers queries by using its own oracle run in TC mode; A' outputs whatever A outputs. A' makes the same number of queries and runs in the same time as A.

Thus, it remains only to determine $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s)$. Let A be an adversary that attacks $\mathcal{E}'$ and makes queries with q total blocks. Define Bad to be the event that $\mathcal{E}'$ ever invokes $\widetilde{\Pi}$ with some tweak more than once. We have $\Pr[\mathsf{Bad}] \le \frac{q^2-q}{2^{n+1}}$. Furthermore, note that A gets no advantage at all if $\overline{\mathsf{Bad}}$ holds: since every tweak used is distinct, the ciphertext given to A is random and independent of b. Therefore, $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) \le \frac{q^2-q}{2^{n+1}}$, which proves the theorem. □

If making a tweakable block cipher requires even one extra XOR, the performance of CBC mode is the same or better than the performance of TC mode. Again, we stress

that TC mode is given to illustrate the conceptual simplicity of proofs for tweakable modes of operation.

4.2. *Tweak Incrementation Encryption (TIE)*

Another symmetric encryption scheme is "tweak incrementation encryption" mode (or TIE for short) which is meant to be similar to CTR mode for block ciphers [1,35,37]. In TIE mode, the message is divided into message blocks $M_0, \ldots, M_m$. An initial tweak IV is generated, and each ciphertext block is generated as follows: $C_i = \widetilde{E}(IV + i, M_i)$, where $IV + i$ refers to simply adding i to IV modulo 2^t. See Fig. 4. This mode was suggested by Schroeppel in his presentation of the Hasty Pudding Cipher.

The proof that TIE mode is secure is extremely similar to the proof that TBC mode is secure, but even more simple.

In "state-preserving" TIE mode, IV is chosen to be 0 for the first message, and subsequent IV values are chosen to be the next value that has not yet been used as the tweak. This requires the user to maintain a state that keeps track of which IV to use next.

Theorem 5. *If $\mathcal{E}$ is the symmetric encryption system made from $\widetilde{E}$ in (state-preserving) TIE mode, then if $q < 2^t$,*

$$\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \leq \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s).$$

Proof. If $\mathcal{E}'$ is the symmetric encryption system made from $\widetilde{\Pi}$ in TIE mode, then $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \leq \mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) + \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s)$, as we established in our proof of Theorem 4.[5] Furthermore, $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}', q, s) = 0$, because in the state-preserving version of TIE mode, no tweak is ever repeated if fewer than 2^t total queries are made. Therefore, every block of output is an independent random string of length n, and the encryption of m_b is random and independent of b. □

In "non-state-preserving" TIE mode, IV is chosen randomly for each message.

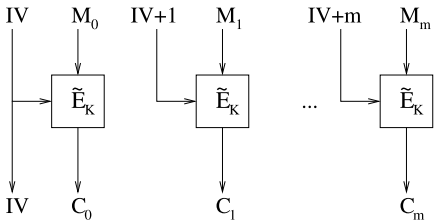

Fig. 4. Tweak incrementation encryption: a mode of operation for tweakable block ciphers to produce secure symmetric encryption. The initial tweak IV is chosen randomly (or sequentially) and incremented by 1 for each successive block.

[5] We again use the natural simulating reduction; this time, A' uses its oracle in state-preserving TIE mode. Again, A' runs in s time and makes q queries.

Theorem 6. *If $\mathcal{E}$ is the symmetric encryption system made from $\widetilde{E}$ in (non-state-preserving) TIE mode, then if $q < 2^t$,*

$$\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \le \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) + \frac{q^2 - q}{2^{n+1}}.$$

Proof. The proof of Theorem 5 shows that $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \le \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) + \Pr[\mathsf{Bad}]$, where Bad is the event that the queries A makes cause $\mathcal{E}$ to use a tweak more than once. (For the state-preserving version of TIE mode, $\Pr[\mathsf{Bad}] = 0$.)

Here, $\Pr[\mathsf{Bad}] \le \frac{q^2-q}{2^{n+1}}$. Pr[Bad] is maximized when A queries $\mathcal{E}$ on one-block messages. In that case, the ith block has at most an $i - 1/2^n$ chance of having a repeated tweak. If message blocks are longer, this probability is decreased for subsequent blocks, to $i'/2^n$, where i' is the number of full messages (rather than message blocks) queried to $\mathcal{E}$ before the current query.

Therefore, $\mathsf{ADV}_{\mathsf{IND\text{-}CPA}}(\mathcal{E}, q, s) \le \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q, s) + \frac{q^2-q}{2^{n+1}}$. □

The advantages of TIE mode are similar to the advantages of CTR mode: for instance, blocks can be deciphered in parallel and online. Also, some work has been done by Rogaway in optimizing tweakable block ciphers when the tweak is changed in predictable ways [32]; such techniques may be applicable here. However, CTR mode allows the sender to precompute the keystream needed for encryption; TIE cannot support that optimization.

Again, TIE mode is less efficient than CTR mode and is given as an example of a conceptually simple proof for a tweakable mode of operation.

4.3. *Tweakable Authenticated Encryption (TAE)*

In this section we suggest an authenticated mode of encryption (TAE) based on the use of a tweakable block cipher. This mode can be viewed as a paraphrase or restatement of the architecture of the OCB (offset codebook) mode proposed by Rogaway et al. [33], modified to utilize tweakable block ciphers rather than DESX-like modules. The result is shown in Fig. 5.

The OCB paper goes to considerable effort to analyze the probability that various encryption blocks all have distinct inputs. We feel that a tweakable mode such as TAE

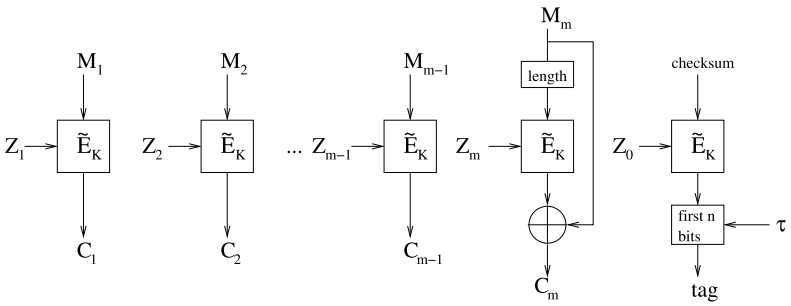

Fig. 5. TAE mode: Authenticated encryption based on a tweakable block cipher.

should be much simpler to analyze, since the use of tweaks obviates this concern. We will give a fairly easy proof that a tweakable block cipher used in TAE mode gives all the security properties claimed for OCB mode.

Following Rogaway et al., an authenticated encryption system involves two algorithms, $\mathcal{AE}$ and $\mathcal{AD}$. $\mathcal{AE}$ (authenticated encryption) takes as input a key K of length k, a message M of length $\geq n$, a nonce N, and a parameter $1 \leq \tau \leq n$, and outputs a ciphertext of length $|M|$ and a tag of length τ. $\mathcal{AD}$ (verification and decryption) takes as input a key K of length k, a ciphertext C, a nonce N, and a tag *tag*, and outputs either a message M of length $|C|$, or $\perp$, the latter signifying that the authentication check failed. As before, we refer to $\mathcal{AE}_K$ and $\mathcal{AD}_K$ for the sake of simplicity, when the key is fixed.

Adversaries are assumed to be "nonce-respecting," meaning that while the adversary has the ability to choose the nonce, the adversary does not make more than one query with the same nonce. Informally, the security requirements are [33]:

- *Pseudorandomness.* To any adversary, the output of OCB mode is pseudorandom. In other words, no adversary can distinguish between an OCB mode oracle and a random oracle that outputs messages of the correct length.
- *Unforgeability.* Any adversary, with the ability to obtain authenticated encryptions of chosen values from an oracle, can forge a new valid encryption with probability at worst negligibly greater than $2^{-\tau}$.

Formally, we define

$$\mathsf{ADV}_{\mathsf{PRF}}(\mathcal{AE}, A) = \left| \Pr\left[K \leftarrow \{0,1\}^k; b \leftarrow A^{\mathcal{AE}_K} : b = 1\right] - \Pr\left[b \leftarrow A^R : b = 1\right]\right|$$

and

$$\mathsf{ADV}_{\mathsf{PRF}}(\mathcal{AE}, q, q', s) = \max_{A \in \mathcal{A}_{q,q',s}} \mathsf{ADV}_{\mathsf{PRF}}(E, A),$$

where R is a function that on input (M, N, τ) outputs a random string of length $|M| + \tau$, and where $\mathcal{A}_{q,q',s}$ is the set of all nonce-respecting algorithms that run in s time and make at most q queries with a total of at most q' blocks. We also define

$$\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, A, \tau) = \Pr\left[K \leftarrow \{0,1\}^k; (C, N, tag) \leftarrow A^{\mathcal{AE}_K} : \right.$$
$$\left. \mathcal{AD}_K(C, N, tag) \neq \perp \wedge |tag| \geq \tau \wedge (C, N) \notin \mathsf{Out}\right] - 2^{-\tau}$$

and

$$\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) = \max_{A \in \mathcal{A}_{q,q',s}, 1 \leq \tau \leq n} \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, A, \tau),$$

where Out is the list of pairs (C, N) for which some (C, N, tag) was an output of the E_K oracle.

TAE Mode Now we describe the methods of TAE mode.

Algorithm $\mathcal{AE}(K, M, N, \tau)$:
  for $i = 1$ **to** $m - 1$ **do**
    $Z_i = N \circ i \circ 0$

$C_i = \widetilde{E}_K(Z_i, M_i)$
$Z_m = N \circ m \circ 0$
if $|M_m| = n$ **then**
$C_m = \widetilde{E}_K(Z_m, M_m)$
else
$C = \widetilde{E}_K(Z_m, |M_m|)$
$C_m = (C|_{|M_m|}) \oplus M_m$
$Z_0 = N \circ |M| \circ 1$
$C_0 = \widetilde{E}_K(Z_0, M_1 \oplus \cdots \oplus M_m)$
$tag = C_0|_\tau$
return $(C_1 \circ \cdots \circ C_m, tag)$

Algorithm $\mathcal{AD}(K, C, N, tag)$:
for $i = 1$ **to** $m - 1$ **do**
$Z_i = N \circ i \circ 0$
$M_i = \widetilde{E}_K(Z_i, C_i)$
$Z_m = N \circ m \circ 0$
if $|C_m| = n$ **then**
$M_m = \widetilde{E}_K(Z_m, C_m)$
else
$C = \widetilde{E}_K(Z_m, |C_m|)$
$M_m = (C|_{|C_m|}) \oplus C_m$
$Z_0 = N \circ |C| \circ 1$
$C_0 = \widetilde{E}_K(Z_0, M_1 \oplus \cdots \oplus M_m)$
if $tag == C_0|_{|tag|}$ **then**
return $M_1 \circ \cdots \circ M_m$
else
return $\perp$

See Fig. 5 for further illustration. τ must be in the range $1 \le \tau \le n$. Here, M is assumed to be broken into m blocks $M_1, \ldots, M_m$, of which the first $m - 1$ are all of size n, and the last is of size at most n but at least 1. C is likewise broken into blocks $C_1, \ldots, C_m$. We use the notation $A|_k$ to denote the k least significant bits of A. When computing $M_1 \oplus \cdots \oplus M_m$, we presume that M_m, if less than n bits, will be padded with 0s.

We now prove that TAE mode satisfies these properties.

Theorem 7. *If $\mathcal{AE}$ is $\widetilde{E}$ used in TAE mode and if $\widetilde{E}$ is a secure tweakable block cipher, then $\mathcal{AE}$ is pseudorandom, and*

$$\mathsf{ADV}_{\mathsf{PRF}}(\mathcal{AE}, q, q', s) \le \mathsf{ADV}_{\mathsf{TPRP}}(\widetilde{E}, q + q', s + q').$$

Proof. Let $\mathcal{AE}'$ refer to the authenticated encryption scheme we obtain by using $\widetilde{\Pi}$ in TAE mode.

To prove that TAE mode is pseudorandom, we note that no tweak is ever repeated when the adversary is nonce-respecting. Therefore, $\mathbf{Adv}_{\mathsf{PRF}}(\mathcal{AE}', q, q', s) = 0$: the output of each tweakable block cipher is independent and random, so the entirety of $\mathcal{AE}'$ is random, just as it would be for R. Therefore,

$$\mathbf{Adv}_{\mathsf{PRF}}(\mathcal{AE}, q, q', s) = \mathbf{Adv}_{\mathsf{PRF}}(\mathcal{AE}, q, q', s) - \mathbf{Adv}_{\mathsf{PRF}}(\mathcal{AE}', q, q', s).$$

If A is an adversary attacking the pseudorandomness of $\mathcal{AE}$ or $\mathcal{AE}'$, we can construct an adversary A' that attacks the underlying tweakable block cipher as follows. A' runs A in its attack on TAE mode, but A' responds to the oracle queries A makes by using its own oracle in TAE mode. A' outputs whatever A outputs. When A makes q' total queries of q total blocks, A' makes $q + q'$ queries to its oracle: one for each block and an additional one per query for the tag. A' runs in time $s + q'$, as A' need only assemble and make the tweakable block cipher queries. Thus we have $|\mathbf{Adv}_{\mathsf{PRF}}(E, q, q', s) - \mathbf{Adv}_{\mathsf{PRF}}(E', q, q', s)| \le \mathbf{Adv}_{\mathsf{TPRP}}(\widetilde{E}, q + q', s + q')$. Thus,

$$\begin{aligned}\mathbf{Adv}_{\mathsf{PRF}}(E, q, q', s) &= \mathbf{Adv}_{\mathsf{PRF}}(E, q, q', s) - \mathbf{Adv}_{\mathsf{PRF}}(E', q, q', s) \\ &\le \left|\mathbf{Adv}_{\mathsf{PRF}}(E, q, q', s) - \mathbf{Adv}_{\mathsf{PRF}}(E', q, q', s)\right| \\ &\le \mathbf{Adv}_{\mathsf{TPRP}}(\widetilde{E}, q + q', s + q'),\end{aligned}$$

which completes the proof. □

Theorem 8. *If $\mathcal{AE}$ is $\widetilde{E}$ used in TAE mode and if $\widetilde{E}$ is a strong tweakable block cipher, then $\mathcal{AE}$ is unforgeable.*[6] *Formally,*

$$\mathbf{Adv}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) \le \mathbf{Adv}_{\mathsf{STPRP}}(\widetilde{E}, s + q + q' + 1, 2s + q' + 1) + \frac{3}{2^{n+1} - 2}.$$

Proof. As before, $\mathcal{AE}'$ is defined to be $\widetilde{\Pi}$ run in TAE mode.

Lemma 4. $\mathbf{Adv}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \le \frac{3}{2^{n+1}-2}$.

Proof. We define two types of forgeries. A *type-1 forgery* is a forgery for which either (1) A outputs (C, N, tag) where N is a nonce that A never used in a query, or (2) A outputs (C, N, tag) where N is a nonce used in a prior query (M', N, τ) made by A where $|M'| \ne |C|$. A *type-2 forgery* is a forgery for which A outputs (C, N, tag) where N is a nonce used in a prior query (M', N, τ) for which $|M'| = |C|$ but for which the result of the prior query did not produce C as the ciphertext. Successful forgeries will always be of one of these two types.

For type-1 forgeries, the value Z_0 used to validate the adversary's forgery will be a tweak value never used before. Therefore, the correct tag will be a random string of length τ, so the probability that the forgery is successful is $2^{-\tau}$.

For type-2 forgeries, there are two subcases. If (C, N, tag) is the adversary's output, let M be the message C encrypts (that is, what $\mathcal{AD}$ would output if the tag were correct),

[6] Note that in [26] we incorrectly claimed that TAE mode is secure without requiring that $\widetilde{E}$ be strong. Thanks to Wang Peng for pointing out this error.

let M' be the message used in the prior query involving N, and let (C', tag') be the output of that query. We define an event Bad, which occurs when the adversary gives a type-2 output in which $M'_1 \oplus \cdots \oplus M'_m = M_1 \oplus \cdots \oplus M_m$. If Bad does not occur in a type-2 forgery, *tag* will be correct with probability at most $\frac{2^{n-\tau}}{2^n-1}$. In such a case, the checksum is distinct, and therefore C_0 is equally likely to be any value other than the value used to generate tag'. At most $2^{n-\tau}$ values can lead to *tag*. Note that

$$\begin{aligned}\frac{2^{n-\tau}}{2^n-1} - 2^{-\tau} &= 2^{-\tau}\left(\frac{2^n}{2^n-1} - 1\right)\\ &= 2^{-\tau}\frac{1}{2^n-1}\\ &\le \frac{1}{2^{n+1}-2}.\end{aligned}$$

If Bad occurs, *tag* will be correct if $tag = tag'$. However, the probability that Bad occurs in a type-2 forgery is very small. Since $C \neq C'$, let C_i be the last ciphertext block for which there is a difference. If $i < m$, the probability that the checksums collide is at most $\frac{1}{2^n-1}$: we can think of all the blocks M_j for $i \neq j$ being calculated first, and then there are $2^n - 1$ equally likely results for M_i, each of which leads to a different checksum. If $i = m$, let $C_{i'}$ be the second-last such block, and the same argument applies. If C_m is the only block that differs, the checksums cannot collide. Thus, the probability that Bad occurs for a type-2 forgery is at most $\frac{1}{2^n-1}$.

Let Ev denote the event that the adversary creates a successful forgery. Let Ev_1 denote the event that the adversary has a type-1 output, and let Ev_2 denote the event that the adversary has a type-2 output. Then for any A,

$$\begin{aligned}\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', A, \tau) &= \Pr[\mathsf{Ev}] - 2^{-\tau}\\ &= \Pr[\mathsf{Ev}|\mathsf{Ev}_1]\Pr[\mathsf{Ev}_1] + \Pr[\mathsf{Ev}|\mathsf{Ev}_2]\Pr[\mathsf{Ev}_2] - 2^{-\tau}\\ &\le \Pr[\mathsf{Ev}|\mathsf{Ev}_1]\Pr[\mathsf{Ev}_1] + \Pr[\mathsf{Ev}|\mathsf{Ev}_2]\Pr[\mathsf{Ev}_2]\\ &\quad - 2^{-\tau}\big(\Pr[\mathsf{Ev}_1] + \Pr[\mathsf{Ev}_2]\big)\\ &= (\Pr[\mathsf{Ev}|\mathsf{Ev}_2] - 2^{-\tau})\Pr[\mathsf{Ev}_2]\\ &\le \Pr[\mathsf{Ev}|\mathsf{Ev}_2] - 2^{-\tau}\\ &= \Pr[\mathsf{Ev}|\mathsf{Ev}_2, \overline{\mathsf{Bad}}]\Pr[\overline{\mathsf{Bad}}] + \Pr[\mathsf{Ev}|\mathsf{Ev}_2, \mathsf{Bad}]\Pr[\mathsf{Bad}] - 2^{-\tau}\\ &\le \Pr[\mathsf{Ev}|\mathsf{Ev}_2, \overline{\mathsf{Bad}}] + \Pr[\mathsf{Bad}] - 2^{-\tau}\\ &= \Pr[\mathsf{Ev}|\mathsf{Ev}_2, \overline{\mathsf{Bad}}] - 2^{-\tau} + \Pr[\mathsf{Bad}]\\ &\le \frac{1}{2^{n+1}-2} + \frac{1}{2^n-1}\\ &= \frac{3}{2^{n+1}-2},\end{aligned}$$

and therefore $\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \leq \frac{3}{2^{n+1}-2}$. This completes the proof of the lemma. □

If A is a forging adversary, we can construct an adversary A' that attacks the underlying tweakable block cipher as follows. A' uses its (encryption) oracle in TAE mode to answer queries for A. When A returns an output, A' uses its oracles to run the TAE decryption procedure on A's output. If the result is $\perp$, A outputs 1, otherwise A outputs 0.

The number of queries made by A' is $q + q' + r + 1$ where r is the number of blocks in the output of A. The time A' takes to run is at most $s + q' + r + 1$. Since $r \leq s$, we observe that A' runs in time at most $2s + q' + 1$ and makes at most $s + q + q' + 1$ queries. Note that $\mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, A') = |\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, A) - \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', A)|$, and thus $\mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, s + q + q' + 1, 2s + q' + 1) \leq |\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) - \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s)|$.

Therefore,

$$\begin{aligned}
\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) &= \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \\
&\quad + (\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) - \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \\
&\leq \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \\
&\quad + \left|\mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}, q, q', s) - \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s)\right| \\
&\leq \mathsf{ADV}_{\mathsf{FRG}}(\mathcal{AE}', q, q', s) \\
&\quad + \mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, s + q + q' + 1, 2s + q' + 1) \\
&\leq \mathsf{ADV}_{\mathsf{STPRP}}(\widetilde{E}, s + q + q' + 1, 2s + q' + 1) + \frac{3}{2^{n+1} - 2},
\end{aligned}$$

which completes the proof of Theorem 8. □

Remarks. It is interesting to note that the construction *loses nothing* in terms of its pseudorandomness advantage compared to the advantage of the tweakable block cipher. Furthermore, in terms of its forging advantage, TAE mode loses a negligible amount of advantage, regardless of the adversary's power! This is somewhat remarkable and helps to emphasize our main point that tweakable block ciphers may be the most natural and useful construct for designing higher-level modes of operation. What is more, we note that if we use LRW mode to instantiate $\widetilde{E}$, TAE mode is very similar to OCB mode. One critical difference is that OCB mode (essentially) derives its choice of h from the key K, whereas our construction would require h to be additional key information (though, as we commented, we could derive h from K as part of that construction in order to reduce the key size). Also, OCB mode uses a Gray code to fine-tune efficiency, which we do not. However, our proof is significantly shorter and simpler than the original proof for OCB mode, which further strengthens our point that tweakable block ciphers are the right primitive for this kind of task. Indeed, Rogaway revisited OCB mode in his Asiacrypt 2004 paper [32]. In that result, among other improvements, the proof was much simpler, thanks in part to Rogaway's use of the tweakable block cipher abstraction.

5. Conclusions

The notion of tweakable block ciphers allows one to "repartition" many cryptographic design problems into two parts: designing good tweakable block ciphers and designing good modes of operation based on tweakable block ciphers. We feel that this repartitioning is likely to be more useful and fruitful than the usual structure, since certain issues (e.g., having to do with collisions, say) can be handled once and for all at the lower level and can then be ignored at the higher levels, instead of having to be dealt with repeatedly at the higher levels.

We feel that the notions of a *tweakable block cipher* and *modes of operation based upon on tweakable block ciphers* are interesting and worthy of further study.

One advantage of this framework is the new division of issues between design and analysis of the underlying primitive and the design and analysis of the higher-level modes of operation. We feel that the new primitive may result in a more fruitful partition.

Acknowledgements

We would like to thank Mihir Bellare, Burt Kaliski, Tadayoshi Kohno, Wang Peng, Zulfikar Ramzan, Matthew Robshaw, Rich Schroeppel, and the anonymous reviewers for helpful discussions and comments. David Wagner was supported by the National Science Foundation (NSF CCF-0424422).

References

[1] American National Standards Institute (ANSI). American National Standard for Information Systems–Data Encryption Algorithm–Modes of Operation (1983)

[2] K. Aoki, H. Lipmaa, Fast implementations of AES candidates, in *Third AES Candidate Conference*, April 2000

[3] M. Bellare, J. Killian, P. Rogaway, The security of cipher block chaining message authentication code. *JCSS* **61**(3), 362–399 (2000)

[4] M. Bellare, T. Kohno, A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. Full version, available at http://www-cse.ucsd.edu/users/mihir/papers/rka.html

[5] M. Bellare, T. Kohno, A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications, in *Advances in Cryptology—EUROCRYPT 2003*, ed. by E. Biham. Lecture Notes in Computer Science (Springer, Berlin, 2003), pp. 491–506

[6] D.J. Bernstein, Floating-point arithmetic and message authentication, March 2000. Unpublished manuscript. Available at http://cr.yp.to/papers.html#hash127

[7] E. Biham, New types of cryptanalytic attacks using related keys. *J. Cryptol.* **7**(4), 229–246 (1994)

[8] E. Biham, A. Biryukov, How to strengthen DES using existing hardware, in *Proceedings ASIACRYPT '94*. Lecture Notes in Computer Science, vol. 917 (Springer, Berlin, 1994), pp. 398–412

[9] J. Black, M. Cochran, T. Shrimpton, On the impossibility of highly-efficient blockcipher-based hash functions, in *Advances in Cryptology—EUROCRYPT 2005*, ed. by R. Cramer. Lecture Notes in Computer Science (Springer, Berlin, 2005), pp. 526–541

[10] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, P. Rogaway, UMAC: Fast and secure message authentication, in *Proceedings CRYPTO '99*. Lecture Notes in Computer Science, vol. 1666 (Springer, Berlin, 1999), pp. 216–233
[11] J. Black, P. Rogaway, CBC MACs for arbitrary-length messages: The three-key constructions. *J. Cryptol.* **18**(2), 111–131 (2005)
[12] D. Chakraborty, P. Sarkar, A general construction of tweakable block ciphers and different modes of operations, in *Inscrypt 2006—Information Security and Cryptography, Second SKLOIS Conference*. Lecture Notes in Computer Science, vol. 4318 (Springer, Berlin, 2006), pp. 88–102
[13] P. Crowley, Mercy: A fast large block cipher for disk sector encryption, in *Fast Software Encryption: 7th International Workshop*. Lecture Notes in Computer Science, vol. 1978 (Springer, Berlin, 2000), pp. 49–63. Also available at: www.ciphergoth.org/crypto/mercy
[14] J. Daemen, Limitations of the Even–Mansour construction, in *Proceedings ASIACRYPT '91*. Lecture Notes in Computer Science, LNCS, vol. 739 (Springer, Berlin, 1991), pp. 495–499
[15] J. Daemen, V. Rijmen, AES proposal: Rijndael. Available at http://www.nist.gov/aes. August (1998)
[16] S. Even, Y. Mansour, A construction of a cipher from a single pseudorandom permutation. *J. Cryptol.* **10**(3), 151–161 (1997)
[17] S. Fluhrer, Cryptanalysis of the Mercy block cipher, in *Fast Software Encryption, 8th International Workshop*, ed. by M. Matsui. Lecture Notes in Computer Science, vol. 2355 (Springer, Berlin, 2002), pp. 28–36
[18] D. Goldenberg, S. Hohenberger, M. Liskov, H. Seyalioglu, E.C. Schwartz, On tweaking Luby–Rackoff blockciphers, in *Advances in Cryptology—ASIACRYPT 2007*. Lecture Notes in Computer Science, vol. 4833 (Springer, Berlin, 2007), pp. 342–356
[19] L. Granboulan, P. Nguyen, F. Noilhan, S. Vaudenay, DFCv2, in *Selected Areas in Cryptography*. Lecture Notes in Computer Science, vol. 2012 (Springer, Berlin, 2001), pp. 57–71
[20] S. Halevi, EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data, in *INDOCRYPT*, ed. by A. Canteaut, K. Viswanathan. Lecture Notes in Computer Science, vol. 3348 (Springer, Berlin, 2004), pp. 315–327
[21] S. Halevi, P. Rogaway, A tweakable enciphering mode, in *Advances in Cryptology: CRYPTO 2003*, ed. by D. Boneh. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 482–429
[22] S. Halevi, P. Rogaway, A parallelizable enciphering mode, in *Topics in Cryptology, CT-RSA 2004*. LNCS, vol. 2964 (Springer, Berlin, 2004), pp. 292–304
[23] C. Jutla, Encryption modes with almost free message integrity, in *Advances in Cryptology—EUROCRYPT 2001*, ed. by B. Pfitzmann. Lecture Notes in Computer Science, vol. 2045 (Springer, Berlin, 2001)
[24] J. Kilian, P. Rogaway, How to protect DES against exhaustive search (an analysis of DESX), in *Proceedings CRYPTO '96*. Lecture Notes in Computer Science, vol. 1109 (Springer, Berlin, 1996), pp. 252–267. See http://www.cs.ucdavis.edu/rogaway/papers/desx.ps for an updated version
[25] M. Liskov, New tools in cryptography: Mutually independent commitment, tweakable block ciphers, and plaintext awareness via key registration. Ph.D. Thesis, MIT Laboratory for Computer Science (2004)
[26] M. Liskov, R. Rivest, D. Wagner, Tweakable block ciphers, in *Advances in Cryptology—CRYPTO 2002*, ed. by M. Yung. Lecture Notes in Computer Science (Springer, Berlin, 2002), pp. 31–46
[27] M. Luby, C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions, in *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, Berkeley, California, 28–30 May 1986
[28] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, Boca Raton, 1997)
[29] K. Minematsu, Improved security analysis of XEX and LRW modes, in *Selected Areas in Cryptography—SAC 2006*. Lecture Notes in Computer Science, vol. 4356 (Springer, Berlin, 2006), pp. 96–113
[30] R. Morris, K. Thompson, Password security: A case history. *Commun. ACM* **22**(11), 594–597 (1979)
[31] M. Naor, O. Reingold, On the construction of pseudo-random permutations: Luby-Rackoff revisited. *J. Cryptol.* **12**, 29–66 (1999). Extended abstract in *Proc. 29th Annual ACM STOC* (1997), pp. 189–199
[32] P. Rogaway, Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC, in *Advances in Cryptology—ASIACRYPT 2004*, Jeju Island, Korea, 5–9 December 2004, ed. by P.J. Lee. Lecture Notes in Computer Science, vol. 3329 (Springer, Berlin, 2004)

[33] P. Rogaway, M. Bellare, J. Black, T. Krovetz, A block-cipher mode of operation for efficient authenticated encryption, in *Eighth ACM Conference on Computer and Communications Security (CCS-8)* (ACM, New York, 2001), pp. 196–205. See http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-doc.htm

[34] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C* (Wiley, New York, 1996)

[35] R. Schroeppel, The Hasty Pudding Cipher. NIST AES proposal, available at http://www.cs.arizona.edu/~rcs/hpc/ (1998)

[36] Victor Shoup, On fast and provably secure message authentication based on universal hashing, in *Proceedings CRYPTO '96*. Lecture Notes in Computer Science, vol. 1109 (Springer, Berlin, 1996), pp. 313–328

[37] US Department of Commerce National Bureau of Standards. DES modes of operation (1980). Federal Information Processing Standards Publication 81