Tweakable Blockciphers with Beyond Birthday-Bound Security

Will Landecker, Thomas Shrimpton, and R. Seth Terashima

Dept. of Computer Science, Portland State University
{landeckw,teshrim,robert22}@cs.pdx.edu

Abstract. Liskov, Rivest and Wagner formalized the tweakable blockcipher (TBC) primitive at CRYPTO'02. The typical recipe for instantiating a TBC is to start with a blockcipher, and then build up a construction that admits a tweak. Almost all such constructions enjoy provable security only to the birthday bound, and the one that does achieve security beyond the birthday bound (due to Minematsu) severely restricts the tweak size and requires per-invocation blockcipher rekeying.
This paper gives the first TBC construction that simultaneously allows for arbitrarily "wide" tweaks, does not rekey, and delivers provable security beyond the birthday bound. Our construction is built from a blockcipher and an ϵ-AXU$_2$ hash function.
As an application of the TBC primitive, LRW suggest the TBC-MAC construction (similar to CBC-MAC but chaining through the tweak), but leave open the question of its security. We close this question, both for TBC-MAC as a PRF and a MAC. Along the way, we find a nonce-based variant of TBC-MAC that has a *tight* reduction to the security of the underlying TBC, and also displays graceful security degradation when nonces are misused. This result is interesting on its own, but it also serves as an application of our new TBC construction, ultimately giving a variable input-length PRF with beyond birthday-bound security.

1 Introduction

A blockcipher $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is typically viewed as a family of permutations E_K over $\{0,1\}^n$, where the index into the family is the key $K \in \{0,1\}^k$. A *tweakable blockcipher* (TBC) extends this viewpoint by adding a second "dimension" to the function family, called a *tweak*. In particular, a TBC $\widetilde{E}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a family of permutations indexed by a pair $(K,T) \in \{0,1\}^k \times \mathcal{T}$. There is, however, a semantic asymmetry between the key and the tweak: the key is secret and gives rise to security, while the tweak may be public and gives rise to variability.

Liskov, Rivest and Wagner [21] formalized the TBC primitive. Their thesis was that primitives with inherent variability are a more natural starting point for building modes of operation, whereas classical constructions would use a blockcipher (deterministic once the key is fixed) and induce variability by using a per-message IV or nonce. Subsequent papers have delivered tweakable enciphering schemes (e.g. [14–16, 32, 8] and others), message authentication

codes(e.g [28]), and authenticated encryption (e.g. [27, 28, 20]) modes of operation. The Skein [30] hash function has a TBC at its core. TBC-based constructions have found widespread practical application for full-disk encryption.

BUILDING TBCS. There are few dedicated TBC designs: the Hasty Pudding [29] and Mercy [10] ciphers natively admit tweaks. The more common approach is to start from a blockcipher and build up a TBC, incorporating support for a tweak without (one hopes) sacrificing whatever security the original blockcipher offered. The original LRW paper itself gave two constructions, which we call LRW1 and LRW2. The former construction is $\mathsf{LRW1}[E]_K(T, X) = E_K(T \oplus E_K(X))$ and it is secure tweakable-PRP[1] if the underlying n-bit blockcipher E is a secure PRP, although there is a birthday-type loss in the reduction. (That is, the security bound becomes vacuous around $2^{n/2}$ queries.) In addition to birthday-bound security, the tweakspace is limited to $\mathcal{T} \subseteq \{0,1\}^n$. The second LRW construction $\mathsf{LRW2}[H, E]_{h,K}(T, X) = h(T) \oplus E_K(X \oplus h(T))$ avoids this length restriction by hashing the tweak. LRW prove that this is a tweakable strong-PRP when E is a secure strong-PRP and h is a random element of an ϵ-almost 2-xor-universal (ϵ-AXU$_2$) hash function family H. But here, too, one finds only birthday-bound security. Variations on the LRW constructions, for example Rogaway's XE and XEX constructions [28], similarly offer provable security only to the birthday bound.

Tweakable blockciphers with beyond birthday-bound (BBB) security may be of particular interest for applications such as large-scale data-at-rest protection, where key management and negotiation issues seem likely to drive up the amount of data protected by a single key. Also, when legacy restrictions require the use of Triple-DES (where $n = 64$), delivering BBB security has obvious benefits. We also note that OCB mode [28] would deliver BBB authenticated-encryption security if constructed over a BBB tweakable blockcipher; other TBC-based constructions with (tight) security reductions to the security of the underlying TBC would similarly benefit.

Nonetheless, constructions of TBCs with BBB security are rare. One due to Minematsu [24] achieves BBB security, but only admits short tweaks (e.g. $\mathcal{T} = \{0,1\}^{n-m}$ for $m \geq n/2$). It requires two blockcipher calls per TBC invocation, and suffers an additional performance penalty by rescheduling one blockcipher key whenever the tweak changes. This last point also violates a TBC design goal, that changing a tweak should more efficient than changing a key.

[1] This notion is formally defined in Section 2. Informally, a TBC $\widetilde{E}$ is a secure tweakable-PRP if, for a random and secret key K, the family of mappings $\widetilde{E}_K(\cdot,\cdot)$ is computationally indistinguishable from a family of random permutations. The tweakable strong-PRP notion allows for inverse queries, too.

A New Construction with BBB Security: CLRW2. Our main technical result is the first TBC construction that has strong tweakable-PRP security beyond the birthday bound, admits essentially arbitrary tweaks, and does not require per-invocation rekeying of any of the underlying objects.

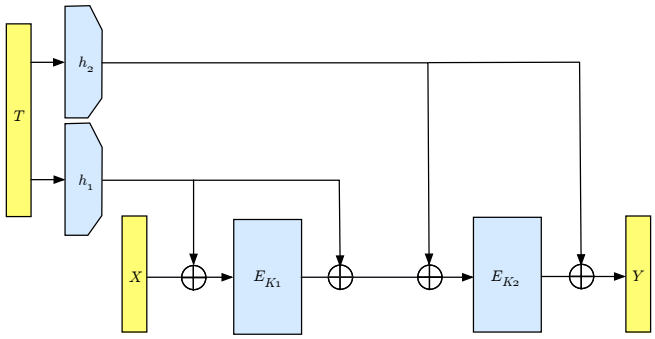

Fig. 1. The CLRW2 Construction.

We call this the *Chained* LRW2 (CLRW2) construction, since it can be written as $\mathsf{LRW2}[H,E]_{h_2,K_2}(T,\mathsf{LRW2}[H,E]_{h_1,K_1}(T,X))$. The bulk of the paper is dedicated to showing that when E is a secure strong-PRP and H is an ϵ-AXU$_2$ hash function family with $\epsilon = 2^{-n}$, the CLRW2 TBC is a strong tweakable-PRP with security against adaptive attackers making $\mathcal{O}(2^{2n/3})$ queries. Figure 2 gives a graphical comparison of our security bound and the birthday bound.

We also consider some variations of CLRW2, for example omitting internal xors, or keying the two blockciphers with the same key.

Note that there are many efficient constructions of ϵ-AXU$_2$ families with $\epsilon \approx 2^{-n}$ and, except perhaps for very long tweaks, the running time of CLRW2 is likely to be dominated by the two blockcipher calls.

Analyzing the TBC-MAC construction and variants. In addition to formalizing the TBC primitive, LRW suggested TBC-based constructions for (authenticated) encryption, hashing and message authentication. The last of these has yet to receive formal analysis, so we consider it. The basic TBC-MAC construction operates as follows. Fix $k, n > 0$ and let $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. Fix $T_0 \in \{0,1\}^n$. Then for any key $K \in \{0,1\}^k$ and a plaintext $M = M_1, \ldots, M_b$ consisting of n-bit blocks, define $\mathsf{TBCMAC}[\widetilde{E}]_K(M) = T_b$ where $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M_i)$ for all $i \in [1..b]$. This is the TBC-MAC (over $\widetilde{E}$) of the input M. It is intuitive to think of TBC-MAC as analogous to CBC-MAC. Indeed, if $\widetilde{E}_K(T,X) = E_K(T \oplus X)$ then we have the CBC-MAC construction. But perhaps by abstracting away the details of $\widetilde{E}$ one can achieve better security than that offered by CBC-MAC? This seems a reasonable expectation, since an attacker can directly influence the input to the blockcipher E in CBC-MAC via the exclusive-or operation, but no such influence

is guaranteed when the chaining value (the tweak) is separated from the plaintext input block. Moreover, it is easy to build TBCs with tweak inputs that are much larger than n bits (LRW already gave one way), and exploiting this may allow for simple twists on the basic TBC-MAC that give better security.

We first consider TBC-MAC as a variable-input-length pseudorandom function (VIL-PRF). We show that it is secure if the underlying TBC is secure tweakable-PRP. Like CBC-MAC, however, TBC-MAC has only birthday-bound security. A small benefit is that this result is not restricted to prefix-free encoded inputs as it is for CBC-MAC. Actually, one can view TBC-MAC as an instance of the Merkle-Damgård iteration [23, 11] over a compression function with a dedicated key input. In this setting Bellare and Ristenpart [3] have already shown that various versions of Merkle-Damgård (plain, suffix-free encoded inputs, prefix-free encoded inputs) are PRF-preserving.

A more interesting result is found if the underlying TBC allows "wide" tweaks, i.e. tweaks that are wider than the blocksize. In this case, a simple nonce-based version of TBC-MAC (TBCMAC2) achieves much better PRF security bounds. In fact, if nonces are properly respected, the mode of operation imparts *no* loss over the security of the underlying TBC. Thus, TBCMAC2 instantiated with a beyond-birthday secure TBC yields a variable-input-length PRF with beyond-birthday security. What's more, the security bound degrades quadratically in the maximum number of times any nonce is repeated, providing more graceful behavior than most nonce-based constructions, which fail catastrophically when a nonce-repeat occurs. Such nonce misuse-resistance can be quite useful in practice.

Lastly, we show that TBC-MAC is unforgeable assuming only that the underlying TBC is likewise unforgeable. This holds only for prefix-free encoded inputs. In fact, this follows from the work of Maurer and Sjödin [22], who give general results for the Merkle-Damgård iteration. When the prefix-free encoding restriction is lifted, we exhibiting a TBC $\widetilde{E}$ that is unforgeable, yet TBC-MAC over $\widetilde{E}$ is easily forged.

Unforgeability Preservation of TBC constructions. In the full version of this work, we make another contribution to the theory of TBC constructions. We begin to explore the provable security of TBCs built from blockciphers that are assumed only to be unpredictable, rather than pseudorandom. In particular, we show that LRW1 is *not* unforgeability preserving. That is, we build a blockcipher E that is unforgeable but for which is it easy to forge LRW1$[E]$. (In fact, we use LRW1 against itself in this result!) Likewise for LRW2, we show that there is an ϵ-AXU$_2$ hash function family and an unforgeable blockcipher E such that LRW2$[H, E]$ is easily forged. (Again, we use LRW1 again to construct the E we need.) At this time, we do not know if CLRW2 remains unforgeable given only unforgeable underlying blockciphers.

Additional Related Work. We have already mentioned the paper of Liskov et al. [21] as the starting point for our work. Goldenberg et al. [17] show how to build a TBC by directly tweaking the Luby-Rackoff construction. Using n-bit

random functions, the resulting $2n$-bit TBC has strong tweakable PRP security to roughly 2^n queries, and can accommodate a tweak of length ℓn using $\ell + 6$ rounds.

Coron et al. [9] show that a three-round Feistel construction over an n-bit TBC with a wide tweak yields a $2n$-bit TBC that has beyond birthday-bound security if the underlying TBC does. Our CLRW2 construction meets this requirement.

The PMAC1 construction by Rogaway [28] builds a (parallelizable) VIL-PRF from a TBC, achieving birthday-bound security. Recently, Yasuda [34] introduced the `PMAC_plus` construction, which has $\mathcal{O}(2^{2n/3})$ security like TBCMAC2 but is more efficient and parallelizable. `PMAC_plus` could be viewed as a construction over a tweakable blockcipher (which might be called the "XXE" construction, following Rogaway's naming convention), but neither the construction nor the proof is cast this way. Separately, Yasuda [33] proves that Algorithm 6 from ISO 9797-1 and SUM-ECBC both have security against $\mathcal{O}(2^{2n/3})$ queries.

The WMAC construction of Black and Cochran [6] is a stateful hash-then-MAC construction that, like our TBCMAC2 construction, allows for graceful (quadratic) security degradation when nonces are repeated. There are various methods for using randomness to build VIL-PRFs with beyond birthday-bound security; for example MACRX [2], RMAC [19], randomized WMAC and enhanced hash-then-MAC [25]

We note that real-world protocols such as TLS [31] employ nonce-based PRFs by using per-message sequence numbers. Nonce-based PRFs also have applications in secure memory; see Garay et al. [18] and references therein.

Bellare and Ristenpart [3] study unforgeability preservation of iterated Merkle-Damgård constructions in the dedicated-key compression-function setting. They show that, in general, these iterations do not preserve unforgeability; however, their counterexample does not apply to TBC-MAC because the compression function they construct is not a TBC.

Zhang et al. [35] study so-called rate-1 MACs constructed from variations of the PGV [26, 7] blockcipher-based compression functions. They show that certain of these compression functions, for example $f(T, X) = E_{K \oplus T}(X)$, iterate (through T) to unforgeable MACs under the assumption that the underlying blockcipher is related-key unpredictable for specific related-key functions. In the case of our example, the related-key functions are $\{K \mapsto K \oplus T \mid T \in \{0,1\}^{|K|}\}$. But in this example and others, assuming that the blockcipher is related-key unforgeable is equivalent to assuming that the compression function is an unforgeable TBC, chaining through the tweak leads to TBC-MAC. Hence our results generalize some of those given by Zhang et al. [35]. We note that TBCs like $E_{K \oplus T}(X)$ are inefficient choices for iteration through the tweak, since they require rescheduling the blockcipher key each round.

We mention in passing that the basic three-key enciphered CBC construction due to Dodis et al. [12] can, in large, part be viewed as an instance of TBC-MAC over the LRW1 TBC. (The IV is no longer a fixed value, but depends on the first input block.)

2 Preliminaries

NOTATION. When $\mathcal{X}$ is a set, we write $x \xleftarrow{\$} \mathcal{X}$ to mean that an element (named x) is uniformly sampled from $\mathcal{X}$. We overload the notation for probabilistic algorithms, writing $x \xleftarrow{\$} M$ to mean that algorithm M runs and outputs a value named x. When X and Y are strings, we write $X \parallel Y$ for their concatenation. When $X \in \{0,1\}^*$ we write $|X|$ for its length. For a tuple of strings $(X_1, X_2, \ldots, X_r)$ we define $|(X_1, X_2, \ldots, X_r)| = |X_1 \parallel X_2 \parallel \cdots \parallel X_r|$. The set $\{0,1\}^n$ is the set of all n-bit strings, $(\{0,1\}^n)^r$ is the set of all nr-bit strings understood as r blocks of n-bits each, and $(\{0,1\}^n)^+$ is the set of all strings that are a positive number of n-bit blocks in length. When $X \in (\{0,1\}^n)^+$, we write $X_1, \ldots, X_b \xleftarrow{n} X$ to mean that X is parsed into b blocks of n-bits each. For a string X of even length n, we define X_L and X_R to be $X[1..\frac{n}{2}]$ and $X[(\frac{n}{2}+1)..n]$, respectively. An *adversary* A is a probabilistic algorithm that takes zero or more oracles. We often use the notation $A \Rightarrow x$ to denote the event (defined over some specified probability space) that some algorithm A outputs value x.

We make use of the code-based game-playing framework of Bellare and Rogaway [5]. When G is a game and A an adversary, we write $\Pr\left[\, G^A \Rightarrow y \,\right]$ for the probability that the **Finalize** procedure of game G outputs y when executed with adversary A. The probability is over the coins of G and A. When the **Finalize** procedure is trivial, returning whatever A does, we omit the procedure from the game and write $\Pr\left[\, A^G \Rightarrow y \,\right]$ for the probability that A outputs y when executed with game G. In games, all boolean flags are initialized to false and all arrays are initially undefined at every point.

FUNCTION FAMILIES AND (TWEAKABLE) BLOCKCIPHERS. Let $\mathcal{K}, \mathcal{D}$ and $\mathcal{R}$ be sets, where at least $\mathcal{K}$ is non-empty. A mapping $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ can be thought of as a function family $F = \{F_K\}$ where for each $K \in \mathcal{K}$ we assign $F_K(\cdot) = F(K, \cdot)$. We will use both representations of the family, as a two-argument mapping and as a set indexed by the first argument, choosing whichever is most convenient. We write $\mathrm{Func}(\mathcal{D}, \mathcal{R})$ for the set of all mappings from $\mathcal{D}$ to $\mathcal{R}$. We write $\mathrm{Perm}(n)$ to denote the set of all permutations (bijections) over $\{0,1\}^n$. We can view each of these as function families with some understood ordering.

A *blockcipher* is a function family $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for all $K \in \mathcal{K}$ the mapping $E_K(\cdot) \in \mathrm{Perm}(n)$. We write $\mathrm{BC}(\mathcal{K}, n)$ to mean the set of all such blockciphers, shortening to $\mathrm{BC}(k, n)$ when $\mathcal{K} = \{0,1\}^k$. A *tweakable blockcipher* (TBC) is a function family $\widetilde{E}\colon \mathcal{K} \times (\mathcal{T} \times \{0,1\}^n) \to \{0,1\}^n$ such that for every $K \in \mathcal{K}$ and $T \in \mathcal{T} \subseteq \{0,1\}^*$ the mapping $\widetilde{E}_K(T, \cdot)$ is a permutation over $\{0,1\}^n$. The set $\mathcal{T}$ is called the *tweakspace* of the TBC, and the element $T \in \mathcal{T}$ is the *tweak*.

SECURITY NOTIONS. Let $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function family, and let A be an adversary taking one oracle. Then we define

$$\mathbf{Adv}^{\mathrm{prf}}_F(A) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K} \,:\, A^{F_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, \rho \xleftarrow{\$} \mathrm{Func}(\mathcal{D}, \mathcal{R}) \,:\, A^{\rho(\cdot)} \Rightarrow 1 \,\right]$$

to be the PRF advantage of A attacking F. Here, and throughout, the probability is over the random choices of the described experient and those of the adversary. We define

$$\mathbf{Adv}^{\text{uf-cma}}_{F}(A) = \Pr\left[\, K \xleftarrow{\$} \mathcal{K}\,;\, (M,\tau) \xleftarrow{\$} A^{F_K(\cdot)} \,:\, F_K(M) = \tau \wedge \text{new-msg} \,\right]$$

to be the UF-CMA advantage (or "forging" advantage) of A. Here the event new-msg holds iff the string M was never asked by A to its oracle.

Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\widetilde{E}: \{0,1\}^k \times (\mathcal{T} \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher. Let $K \xleftarrow{\$} \{0,1\}^k$, $\pi \xleftarrow{\$} \text{Perm}(n)$, and $\mathit{\Pi} \xleftarrow{\$} \text{BC}(\mathcal{T}, n)$. Then we define

$$\begin{aligned}
\mathbf{Adv}^{\text{prp}}_{E}(A) &= \Pr\left[\, A^{E_K(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\pi(\cdot)} \Rightarrow 1 \,\right] \\
\mathbf{Adv}^{\text{sprp}}_{E}(A) &= \Pr\left[\, A^{E_K(\cdot), E_K^{-1}(\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1 \,\right] \\
\mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(A) &= \Pr\left[\, A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathit{\Pi}(\cdot,\cdot)} \Rightarrow 1 \,\right] \\
\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(A) &= \Pr\left[\, A^{\widetilde{E}_K(\cdot,\cdot), \widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\mathit{\Pi}(\cdot,\cdot), \mathit{\Pi}^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right]
\end{aligned}$$

to be (respectively) the PRP, strong PRP, tweakable-PRP, and strong tweakable-PRP advantages of A, an adversary taking the indicated number of oracles. These probabilities are over the random coins of A and the random choices of K, π, and $\mathit{\Pi}$, as appropriate.

A function family $F\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ is ϵ-almost-XOR-universal (ϵ-AXU$_2$) if for all distinct $X, X' \in \mathcal{D}$ and $Y \in \mathcal{R}$, $\Pr\left[\, K \xleftarrow{\$} \mathcal{K} \,:\, F_K(X) \oplus F_K(X') = Y \,\right] \;\leq\; \epsilon$.

Resourses and conventions. We consider the following adversarial resources: the running time t, the number of oracle queries asked q, and the total length of these queries μ. For the PRP and strong PRP notions, we suppress μ since it is implicitly computable from q and the blocksize. In the UF-CMA advantage, μ includes the length of the output forgery attempt (M, τ). It will often be the case that queries (and forgery attempts) are strings in $(\{0,1\}^n)^+$ for some blocksize $n > 0$, and here it will be convenient to speak of the total number of blocks $\sigma = \mu/n$. The running time of an adversary is relative to some (implicit) fixed underlying model of computation. Running times will always be given with respect to some security experiment, and we define the running time to include the time to execute the entire experiment. We assume that adversaries do not make pointless queries: they do not repeat queries, nor do they ask queries that are outside of the domain of oracles they may access.

3 Tweakable SPRP-security of CLRW2

The centerpiece of this work is a TBC construction that provides BBB security, admits a large tweakspace, and does not require rekeying of any underlying

object. Given a blockcipher $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and a hash function family $H\colon \mathcal{K}_H \times \mathcal{D} \to \{0,1\}^n$, the CLRW2 construction $\widetilde{E}[H,E]\colon (\mathcal{K}_H)^2 \times (\{0,1\}^k)^2 \times \mathcal{D} \times \{0,1\}^n \to \{0,1\}^n$ is given by

$$\widetilde{E}[H,E]_{h_1,h_2,K_1,K_2}(T,X) = \\ E_{K_2}(E_{K_1}(X \oplus H_{h_1}(T)) \oplus H_{h_1}(T) \oplus H_{h_2}(T)) \oplus H_{h_2}(T).$$

The following theorem is our main technical result.

Theorem 1. *Fix $k, n > 0$ and let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Fix a non-empty set $\mathcal{K}_H$, and let $\mathcal{D} \subseteq \{0,1\}^*$. Let $H\colon \mathcal{K}_H \times \mathcal{D} \to \{0,1\}^n$ be an ϵ-AXU_2 function family. Let $\widetilde{E} = \widetilde{E}[H,E]$ be the CLRW2 construction, defined above. Let A be an adversary asking a total of q queries to its oracles, these of total length μ, and running in time t. Let $\hat{\epsilon} = \max\{\epsilon, 1/(2^n - 2q)\}$. Then there exists an adversary B using the same resources, such that.*

$$\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(A) \leq 2\mathbf{Adv}^{\text{sprp}}_{E}(B) + \frac{6q^3\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}^2}$$

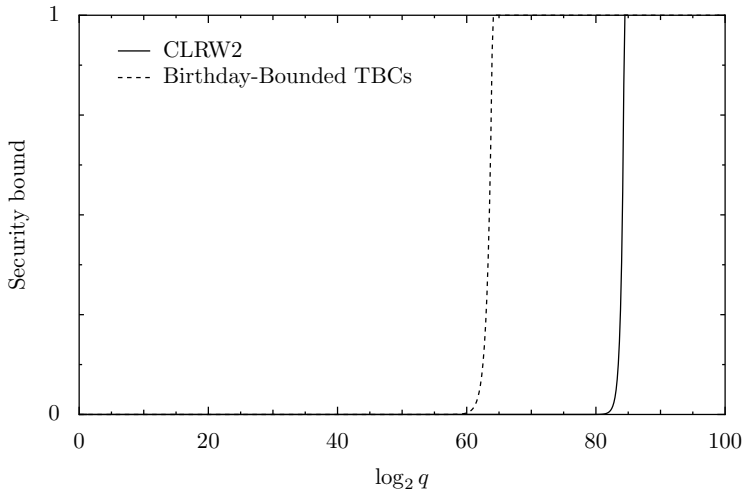

Fig. 2. The maximum advantage of an adversary making q queries agaisnt CLRW2 (solid line) and constructions limited by the birthday bound, $q^2/2^n$ (dashed line). Here, $n = 128$, $\epsilon = 2^{-n}$, and we have assumed the $\mathbf{Adv}^{\text{sprp}}_{E}(B)$ term is negligible.

This bound deserves some interpretation. Consider $\epsilon = 2^{-n}$ (since there are efficient constructions meeting this), and assume $q \leq 2^{n-2}$. Then $\hat{\epsilon} \leq 1/2^{n-1} \approx 2^{-n}$ for interesting values of n. The second term in the bound is at most p when

$q \leq (p/(p+6))^{1/3}\hat{\epsilon}^{-2/3}$, so for any small constant p we have $q = \mathcal{O}(2^{2n/3})$. Thus when $\mathbf{Adv}^{\mathrm{sprp}}_{E}(B)$ is sufficiently small, CLRW2 is secure as a tweakable-SPRP up to about $2^{2n/3}$ queries.[2] Figure 2 gives a graphical comparison of our bound and the standard birthday bound.

Proof overview. The proof of Theorem 1 is quite long and involved, so we'll start by giving a high-level overview of it. Proofs demonstrating birthday-bound security for TBC constructions typically "give up" if the adversary can cause a collision at a blockcipher input. In constructions like LRW1 and LRW2, the TBC output is "no longer random", even when the blockcipher has been replaced by a random permutation. We overcome this problem by using two rounds of LRW2, and showing that it takes two independent collisions *on the same query* to force non-random CLRW2 outputs.

The chief difficulty is ensuring that the second LRW2 round can withstand a collision so long as there was not also one on the first round. To this end, we argue that given a collision-free first round, the resulting distribution of CLRW2 output values — including those which *require* a second-round collision to obtain — is extremely close to that of an ideal TBC.

The bulk of the proof is a sequence of games bounding the success probability of an adversary in the information-theoretic setting, where the blockciphers have been replaced by random permutations. The first three games address first-round collisions, and show that the distribution of CLRW2 outputs is consistent with that of an ideal cipher unless there is simultaneous a second-round collision. Our next three games address the case in which there is no first-round collision. By swapping the order in which dependent random variables are assigned values, we can choose the output early on in the game, and gain insight into the distribution by which it is governed. This distribution is shown to be very close to the ideal one. The final two games are used to derive an upper bound for the probability that the adversary can set a "bad flag", which would force the game to exhibit non-ideal behavior. In the end, we are able to assume that the adversary is non-adaptive by giving it explicit control over oracle return values. At that point, the ϵ-AXU$_2$ property can be applied.

Proof. For notational simplicity, we write h_1 for H_{h_1}, and h_2 for H_{h_2}; this should cause no confusion. The majority of the proof will consider the construction $\widetilde{E}$ with E_{K_1} and E_{K_2} replaced with random permutations π_1 and π_2, which we write as $\widetilde{E}_{h_1,h_2,\pi_i,\pi_2}$. At the end we can make a standard move to lift to the fully complexity theoretic setting.

Let A be an adversary making q queries. If the i^{th} query is to the left (encryption) oracle, we denote the query with (T_i, X_i) and the response with Y_i; if the query is to the right (decryption) oracle, the roles of X_i and Y_i are reversed. We make the standard simplifying assumption that A makes no redundant queries by, for example, giving its encryption oracle the result of a decryption oracle

[2] We note that $\mathbf{Adv}^{\mathrm{sprp}}_{E}(B)$ will be at least $t/2^k \approx q/2^k$ by exhaustive key search so, $q = 2^{2n/3}$ requires $k > 2n/3$, which is met by AES ($k = n = 128$) and DES ($k = 56, n = 64$).

query with the same tweak. We denote by $\mathcal{Y}_i$ the set of permissible (tweak-respecting) return values for an encryption oracle query, and similarly, $\mathcal{X}_i$ is the set of permissible return values for a decryption oracle query. That is,

$$\mathcal{Y}_i = \{0,1\}^n \setminus \{Y_j \,:\, j < i, T_j = T_i\}$$
$$\mathcal{X}_i = \{0,1\}^n \setminus \{X_j \,:\, j < i, T_j = T_i\}\,.$$

An encryption oracle simulating an ideal cipher $\Pi \xleftarrow{\$} \mathrm{BC}(\mathcal{T}, n)$ would sample its return value Y_i from $\mathcal{Y}_i$.

The bulk of this proof concerns showing that a sequence of games are identical, or are identical until a specified event occurs (a boolean variable is set to true). Due to space limitations, we will simply describe game transitions on a high level and refer readers interested in a more rigorous argument to the full version of this paper. The permutations π_1 and π_2 are constructed lazily, while h_1 and h_2 are already defined. Initially, boolean variables have the value false.

Note that $\widetilde{E}$ is the dual of $\widetilde{E}^{-1}$, in the sense that $\widetilde{E}^{-1}_{h_1,h_2,\pi_1,\pi_2}(Y,T) = \widetilde{E}_{h_2,h_1,\pi_2^{-1},\pi_1^{-1}}(Y,T)$. When arguing that transitions between games are correct, we will exploit this duality by limiting our discussion to changes in the encryption oracle, and hence to queries made to that oracle; the arguments used to justify the corresponding changes in the decryption oracle are practically identical. Therefore fix some value $i \in [1..q]$, and assume the i^{th} query is to the encryption oracle.

Game *G1* simulates $\widetilde{E}$ by defining π_1 and π_2 through lazy sampling, so

$$\Pr\left[\, A^{\widetilde{E},\widetilde{E}^{-1}} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G1} \Rightarrow 1 \,\right].$$

In Game *G2*, we change what happens when there is a collision at the first blockcipher: we sample $Y_i \xleftarrow{\$} \mathcal{Y}_i$, but raise a bad flag if we also encounter a collision at the input of second blockcipher (bad_1) or if $Y_i \oplus h_2(T_i)$ is already in its range ($\mathsf{bad2}$). Should either of these events occur, we fall back to the lazy-sampling method of Game *G1* to choose a new value for Y_i. Game *G3* is identical to Game *G2*, except Y_i is not reassigned after a bad flag is set. Hence

$$\Pr\left[\, A^{G1} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G2} \Rightarrow 1 \,\right] \le \Pr\left[\, A^{G3} \Rightarrow 1 \,\right] + \Pr\left[\, A^{G3} \,:\, \mathsf{bad}_1 \vee \mathsf{bad2} \,\right].$$

Next we modify the section of code in Game *G3* that is executed when no collision occurs at π_1; i.e., when $X_i \oplus h_1(T_i) \neq X_j \oplus h_1(T_j)$ for all $j < i$. The behavior of the encryption oracle in this game during the i^{th} query can be completely described by the pair (P_i, Q_i), where P_i is the output of π_1 and Q_i is the output of π_2. The oracle's output, Y_i, is uniquely determined by Q_i, since $Y_i = Q_i \oplus h_2(T_i)$. Hence, we treat the pair (P_i, Q_i) as a single random variable; any method of assigning it a value that respects the joint distribution on P_i and Q_i preserves the black-box behavior of Game *G3*'s oracles.

Fix a query (X_i, T_i). Suppose no collision occurs at π_1 (i.e., $X_i \oplus h_1(T_i) \neq X_j \oplus h_1(T_j)$ for all $j < i$). Call $s \in \{0,1\}^n$ *possible* if $\Pr[\, Q_i = s \,] > 0$. A possible s is *fresh* when $s \neq Q_j$ for all $j < i$ (i.e., s is not yet in the range of π_2), and

stale otherwise. The correspondence $Y_i = Q_i \oplus h_2(T_i)$ between Y_i and Q_i allows us to describe the distribution governing Y_i in terms of Q_i. To do so, we first define:

$$S_1 = \{y \in \{0,1\}^n \ : \ y \oplus h_2(T_i) \text{ is fresh}\}$$
$$S_2 = \{y \in \{0,1\}^n \ : \ y \oplus h_2(T_i) \text{ is stale}\}$$
$$S_3 = \{y \in \mathcal{Y}_i \ : \ y \oplus h_2(T_i) \text{ is not possible}\}$$
$$S_4 = \left\{y \in \overline{\mathcal{Y}_i} \ : \ y \oplus h_2(T_i) \text{ is not possible}\right\}$$

(Hence $\{0,1\}^n = S_1 \cup S_2 \cup S_3 \cup S_4$). One can show that for fixed $k = 1, 2, 3, 4$, any two values in S_k are equally likely to be returned by the Game *G3*'s encryption oracle on the i^{th} query. With a little effort, one can compute these probabilities in terms of $N = |\{p \ : \ \Pr[\, P_i = p \,] > 0\}|$ — the number of values not in π_1's range — and each $|S_k|$. As Figure 3 shows, the resulting distribution is very close to the one an ideal cipher would provide.

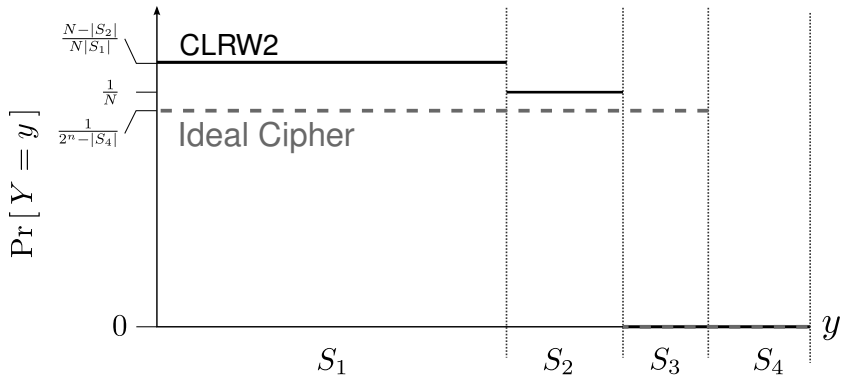

Fig. 3. When there is no collision at π_1, the distribution governing $\widetilde{E}$'s outputs is very close to the distribution an ideal cipher would provide. Horizontal scaling suggests plausible relative sizes of each $|S_k|$: likely $|S_1| \gg |S_2 \cup S_4| \gg |S_3|$. This graph is accurate for the oracles in Games 1–5.

The two distributions vary with each query (and in particular, with each tweak change), and an adaptive adversary could conceivably make the difference between them significant after a large number of queries. Intuitively, S_3 is the problematic set, and indeed, $\Pr\left[\, Y_i \xleftarrow{\$} \mathcal{Y}_i \,;\, Y_i \in S_3 \,\right]$ is the statistical distance between the two distributions. The next few games reveal this more explicitly by "bubbling up" Y_i's assignment in the pseudocode, then linking the two distributions together by setting a bad flag if $Y_i \xleftarrow{\$} \mathcal{Y}_i$ gives $Y_i \in S_3$.

In Game *G4*, we flip a weighted coin to determine if Q_i is fresh or stale, and then choose (P_i, Q_i) after conditioning on the outcome.

Next, instead of using the coin flip to determine Q_i (or equivalently, Y_i), Game *G5* samples $Y_i \xleftarrow{\$} \mathcal{Y}_i$ to determine Q_i — and hence how the coin landed. Again, ultimately all that matters is the distribution on the joint random variable (P_i, Q_i). Since $Q_i = Y_i \oplus h_2(T_i)$ may be neither fresh nor stale (i.e., not possible), Game *G5* falls back to the technique of Game *G4* when $Y_i \in S_3$, after setting bad_3. This overwrites the value assigned to Y_i.

Finally, in Game *G6*, we do not do anything special after setting bad_3; we simply keep the value of Y_i we originally sampled from $\mathcal{Y}_i$.

Game *G7* simplifies some of Game *G6*'s flow. In particular, Y_i is now always sampled from $\mathcal{Y}_i$ (regardless of whether or not there is a collision at π_1), and its value is never overwritten; consequently, the line $Y_i \xleftarrow{\$} \mathcal{Y}_i$ is moved to the start of the encryption oracle pseudocode. Since a family of random permutations would return Y_i from this precise distribution, Game *G7* simulates an ideal cipher:

$$\Pr\left[\, \Pi \xleftarrow{\$} \mathrm{BC}(\mathcal{T}, n)\,;\, A^{\Pi,\Pi^{-1}} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G7} \Rightarrow 1 \,\right].$$

As consequences of the Fundamental Lemma of Game Playing,

$$\begin{aligned}\Pr\left[\, A^{G3} \Rightarrow 1 \,\right] &= \Pr\left[\, A^{G4} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G5} \Rightarrow 1 \,\right] \\ &\le \Pr\left[\, A^{G6} \Rightarrow 1 \,\right] + \Pr\left[\, A^{G6} \,:\, \mathsf{bad}_3 \,\right] \\ &= \Pr\left[\, A^{G7} \Rightarrow 1 \,\right] + \Pr\left[\, A^{G7} \,:\, \mathsf{bad}_3 \,\right],\end{aligned}$$

and similarly

$$\Pr\left[\, A^{G3};\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] \le \Pr\left[\, A^{G7};\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + \Pr\left[\, A^{G7};\, \mathsf{bad}_3 \,\right].$$

Using a standard hybrid argument, it follows that there exists an SPRP adversary B making q queries and running in time $O(t)$ such that

$$\mathbf{Adv}^{\widetilde{\mathrm{sprp}}}_{\widetilde{E}}(A) \le 2\mathbf{Adv}^{\mathrm{sprp}}_{E}(B) + \Pr\left[\, A^{G7};\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + 2\Pr\left[\, A^{G7};\, \mathsf{bad}_3 \,\right].$$

We wish to find an upper bound for the probabilities in this expression. Predictably, the difficulty here is that A is adaptive, and hence its queries are not independent of, for example, h_1. In Game *G8*, we give the adversary control over what value is assigned to Y_i (or X_i, in the case of decryption queries), but insist that it be in $\mathcal{Y}_i$ or $\mathcal{X}_i$, as appropriate. Because this new adversary $\widetilde{A}$ can compute $\mathcal{Y}_i$ and $\mathcal{X}_i$, he may simulate the oracles of Game *G7* if desired; hence, given q queries, $\widetilde{A}$ can set the bad flags in Game *G8* with probability at least as high as any A can set the corresponding flags in Game *G7*. The oracle's outputs are now deterministic, and may be (trivially) computed by the adversary in advance. Hence, we may assume without loss of generality that $\widetilde{A}$ is non-adaptive.

One can show that in order to set bad_m ($m = 1, 2, 3$), $\widetilde{A}$ must make a sequence of queries such that there exist $i, j, k \le q$ such that $j, k \ne i$, $X_i \oplus h_1(T_i) = X_j \oplus h_1(T_j)$, and either

1. $Y_i \oplus h_2(T_i) = Y_k \oplus h_2(T_k)$ or
2. $\pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i) = \pi_1(L_k) \oplus h_1(T_k) \oplus h_2(T_k)$,

where $L_i = X_i \oplus h_1(T_i)$, and similarly for L_k. Either case requires query i to "collide" with independently with two other queries in some fashion. The ϵ-AXU$_2$ property makes the first type of collision unlikely, and the fact that in this game, $\pi_1(L_i)$ will be sampled from a set of size at least $1/(2^n - 2q)$ makes it unlikely that $\pi_1(L_i)$ will be assigned the unique value that causes the second type of collision to occur (for given i, j, and k).

Let $\hat{\epsilon} = \max(\epsilon, 1/(2^n - 2q))$. By carefully computing these upper bounds for the probabilities of these collisions and taking a union bound over all permissible (i, j, k) pairs, one can show that

$$\mathbf{Adv}^{\widetilde{\text{sprp}}}_{\widetilde{E}}(A) \leq 2\mathbf{Adv}^{\text{sprp}}_{E}(B) + \frac{6q^3\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}^2},$$

completing the proof. □

Attacks on simpler variants. Having seen our construction, one wonder if simpler variants work. For example, consider CLRW2 without the first $H_{h_2}(T)$ XOR operation, leaving

$$\widetilde{E}_{h_1,h_2,K_1,K_2}(T, X) = H_{h_2}(T) \oplus E_{K_2}(H_{h_1}(T) \oplus E_{K_1}(H_{h_1}(T) \oplus X)).$$

This variation permits birthday-bound attack. Namely, an adversary could submit queries in pairs, (T_i, X') and (T_i, X''), where X' and X'' are fixed, and a new random tweak is used for each pair. By remembering $\widetilde{E}(T_i, X') \oplus \widetilde{E}(T_i, X'')$ values, which are independent of H_{h_2}, it could detect collisions in H_{h_1}, say by using a hash table. That is, if $H_{h_1}(T_i) = H_{h_1}(T_j)$, then $\widetilde{E}(T_i, X') \oplus \widetilde{E}(T_i, X'') = \widetilde{E}(T_j, X') \oplus \widetilde{E}(T_j, X'')$. The converse is false, but false positives could be weeded out by testing a small number of X-values. Such an adversary would gain advantage close to one. Similar variations on $\widetilde{E}$ permit analogous attacks, though we believe (but do not prove) that omitting the second $H_{h_1}(T)$ XOR operation yields a construction secure against adversaries constrained to chosen-plaintext attacks.

One might also wish to try setting $K_2 = K_1$. While we know of no attacks here, modifying our proof to accomodate this change would be non-trivial. In particular, bounding certain probabilities required us to trace back through a game's execution history to determine when π_1 became defined at particular points (L_i or L_k in the above proof); this task would be messier and more difficult to verify if $\pi_2 = \pi_1$. Still, this may merit future investigation.

4 PRF-security of TBC-MAC

The TBC-MAC function family. Fix $k, n > 0$ and let $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. We define the TBC-MAC function

family $\mathsf{TBCMAC}[\widetilde{E}]\colon \{0,1\}^k \times (\{0,1\}^n)^+ \to \{0,1\}^n$ as follows. On input $K \in \{0,1\}^k$ and $M \in (\{0,1\}^n)^+$, let $\mathsf{TBCMAC}[\widetilde{E}]_K(T,M) = T_b$ where $T_0 = 0^n$; let $M_1, \cdots, M_b \xleftarrow{n} M$, and $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M_i)$ for $i \in \{1, \ldots, b\}$. To extend the domain to $\{0,1\}^*$, one could introduce an explicit, unambiguous padding rule mapping $\{0,1\}^* \to (\{0,1\}^n)^+$, say mapping $M \mapsto M\|10^r$ where r is the smallest integer needed to reach a block boundary. But for simplicity we assume that all strings input to $\mathsf{TBCMAC}[\widetilde{E}]$ are block-aligned. We extend this assumption by writing $\mathsf{TBCMAC}^{\mathsf{pf}}$ for the TBC-MAC construction restricted to prefix-free encoded, block-aligned inputs.

Building from a "narrow" tweaksize TBC. Our first result in this section is a natural one. We prove that TBC-MAC is a secure PRF if the underlying TBC $\widetilde{E}$, with n-bit tweaks and blocksize, is secure as a tweakable-PRP. One might hope that the security bound for $\mathsf{TBCMAC}[\widetilde{E}]$ is better than for CBC-MAC over an n-bit blockcipher, since the former is intuitively a "stronger" object than the latter. This is not the case. This is because the IV is fixed; thus an adversary can ask a series of distinct one-block messages and wait for a collision. Considering the information-theoretic setting, the fixed IV effectively reduces the ideal cipher to a random permutation in the first round, and so the standard PRP-PRF distinguishing attack forces us to accept birthday-bound security. The following theorem closely follows the code-based game-playing proof of CBC-MAC due to Bellare and Rogaway [5]. We note that a tighter bound could be achieved (with more work) following the techniques of Bellare et al. [4]. The proof appears in the full version.

Theorem 2. (TBCMAC is a PRF.) *Fix $n > 0$. Let $\widetilde{E}\colon \{0,1\}^n \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher. Let A be an adversary running in time t, asking q queries, each of length at most ℓ blocks of n-bits. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{TBCMAC}[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B) + \frac{(q\ell)^2}{2^n}$$

for an adversary B that runs in time $t' = t + \mathcal{O}(\ell q)$ and asks at most $q' = q\ell$ queries. ∎

Building from a "wide" tweaksize TBC. The LRW2 and CLRW2 constructions each give TBC that can handle tweaks that are potentially much larger than the blocksize. So we now consider the security of a nonce-based version of TBC-MAC based upon such a TBC. In particular, fix $k, n, b > 0$ and let $\widetilde{E}\colon \{0,1\}^k \times (\{0,1\}^{n+b+1} \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher with tweaksize $n + b + 1$ bits and blocksize n bits. For an ℓ-block message $M_1, \ldots, M_\ell$ where $\ell > 1$, nonce $N \in \{0,1\}^b$, and a fixed $T_0 = IV$, define $\mathsf{TBCMAC2}[\widetilde{E}]_K(N,M)$ as $T_\ell = \widetilde{E}_K(T_{\ell-1} \,\|\, 1 \,\|\, N, M_\ell)$ where for $i = 1$ to $\ell - 1$, $T_i = \widetilde{E}_K(T_{i-1} \,\|\, 0 \,\|\, 0^b, M_i)$. We say that a PRF-adversary A is *nonce-respecting* (for $\mathsf{TBCMAC2}$) if it never repeats a nonce. The *multiplicity* α of a nonce N is the number of times it is used in an attack, e.g. $\alpha = 1$ for every nonce if the attack is nonce-respecting.

Theorem 3. (TBCMAC2 is a PRF.) *Fix $n > 0$ and $b \geq 0$. Let $\widetilde{E}\colon \{0,1\}^n \times (\{0,1\}^{n+b+1} \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher. Let $\mathsf{TBCMAC2}[\widetilde{E}]$ be as described above. Let A be an adversary that runs in time t, asks q queries for the form (N, M) where the length of M is at most ℓ blocks. Assume that there are r distinct values of N among these queries, and let $\alpha_1, \ldots, \alpha_r$ denote the multiplicities of these. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{TBCMAC2}[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B) + \frac{1}{2^{n+1}}\left(\sum_{i=1}^{r} \alpha_i(\alpha_i - 1)\right) + \sum_{i=1}^{r}\binom{\alpha_i}{2}\frac{(2\ell+1)(2\ell)}{2^n}$$

where B runs in time $t' = t + \mathcal{O}(q\ell)$ and asks at most $q' = q\ell$ queries. Specifically, if A is nonce-respecting, $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{TBCMAC2}[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B)$. ∎

The proof of Theorem 3 appears in the full version.

5 Unforgeability-Preservation of TBC-MAC

TBC-MAC preserves the unforgeability of its underlying TBC when the TBC-MAC inputs are prefix-free. Since, qualitatively, this amounts to a new application of an existing result by Maurer and Sjödin [22], we defer our proof until the full version.

Theorem 4. ($\mathsf{TBCMAC}^{\mathsf{pf}}$ preserves UF-CMA.) *Fix $k, n > 0$, and let $\widetilde{E}\colon \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a TBC. Let A be an adversary for $\mathsf{TBCMAC}^{\mathsf{pf}}[\widetilde{E}]$ that runs in time t, asks q queries, these totaling σ blocks of n-bits in length. Then there exist adversaries B and C such that*

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{TBCMAC}^{\mathsf{pf}}[\widetilde{E}]}(A) \leq \frac{\sigma(\sigma-1)}{2}\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B) + \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(C)$$

where B runs in time $t_B \leq t$, asks $q_B \leq \sigma$ queries totalling $\sigma_B \leq 2\sigma$ blocks; and where C runs in time $t_C = t$, asks $q_C = \sigma$ queries totalling $\sigma_C = 2\sigma$ blocks. ∎

However, if adversaries may mount an attack using non-prefix-free inputs, it is possible to forge TBC-MAC.[3] The following lemma says that there exists a TBC $\widetilde{F}$ that is unforgeable if some underlying TBC $\widetilde{E}$ is. Liskov et al. [21] provide a TBC $\widetilde{E}$ with the required signature. The proof appears in the full version.

Lemma 1. *Let $\widetilde{E} : \{0,1\}^k \times \{0,1\}^{3n} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher. Let $\widetilde{F} : \{0,1\}^k \times \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ be a tweakable blockcipher defined by $\widetilde{F}_K(T_L \parallel T_R, X_L \parallel X_R) = X_L \oplus T_R \parallel \widetilde{E}_K(X_L \parallel T_L \parallel T_R, X_R)$. Then $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{F}}(A) \leq \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B)$ where the resources of adversaries A and B are the same.* ∎

[3] We note that Bellare and Ristenpart [3] have already shown that the Merkle-Damgård iteration is not unforgeability preserving for arbitrary inputs. However, their counterexample does not suffice here, because the compression function they build is *not* a TBC.

We now show that TBC-MAC instantiated with $\widetilde{F}$ admits efficient forging attacks if arbitrary inputs are allowed.

Theorem 5. *(*TBCMAC *is not UF-CMA preserving.) Let $\widetilde{E}$ be a tweakable blockcipher and let $\widetilde{F}$ be as defined in Lemma 1. Then there exists an adversary A that asks $q = 2$ queries totalling $\sigma = 12$ blocks of n-bits such that* $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{TBCMAC}[\widetilde{F}]}(A) = 1$. ∎

Proof. Consider the adversary A that queries $Y^1 \leftarrow \mathsf{TBCMAC}[\widetilde{F}]_K(0^{2n} \,\|\, 0^{2n})$, and then forges with $X^* = 0^{2n}$ and $Y^* = 0^n \,\|\, Y^1_L$. The forgery is valid; we leave the confirmation of this fact to the interested reader. □

Acknowledgments

The authors of this work were supported by NSF grants CNS 0627752 and CNS 0845610.

References

1. J. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. *Advances in Cryptology – CRYPTO 1999,* LNCS vol. 1666, Springer, pp. 252–269, 1999.
2. M. Bellare, O. Goldreich and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. *Advances in Cryptology – CRYPTO 1999*, LNCS vol. 1666, Springer, pp. 270-287
3. M. Bellare and T. Ristenpart. Hash functions in the dedicated-key setting: Design choices and MPP transforms. *International Colloquium on Automata, Languages, and Programming – ICALP 2007*, LNCS vol. 4596, Springer, pp. 399–410, 2007.
4. M. Bellare, K. Pietrzak, P. Rogaway. Improved security analyses for CBC MACs. *Advances in Cryptology – CRYPTO 2005*, LNCS vol. 3621, Spring, pp. 527–541, 2005.
5. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. *Advances in Cryptology – EUROCRYPT 2006*, LNCS vol. 4004, Springer, pp. 409–426, 2006.
6. J. Black and M. Cochran. MAC Reforgability. *Fast Software Encryption – FSE 2009*, LNCS vol. 5665, Springer, pp. 345–362, 2009.
7. J. Black, P. Rogaway, T. Shrimpton and M. Stam. An analysis of the blockcipher-based hash functions from PGV. *Journal of Cryptology*, vol. 23, no. 4, pp. 320–325, Springer, 2010.
8. D. Chakraborty and P. Sarkar. A new mode of encryption providing a tweakable strong pseudo-random permutation. *Fast Software Encryption – FSE 2006*, LNCS vol. 4047, Springer, pp. 293-309, 2006
9. J-S. Coron, Y. Dodis, A. Mandal and Y. Seurin. A Domain Extender for the Ideal Cipher. *Theory of Cryptography – TCC 2010*, LNCS vol. 5978, Springer, pp. 273-289, 2010.
10. P. Crowley. Mercy: A Fast Large Block Cipher for Disk Sector Encryption. *Fast Software Encryption – FSE 2000*, LNCS 19787, pp. 49-63, 2000.

11. I. Damgård. A design principle for hash functions. *Advances in Cryptology – CRYPTO 1989*, LNCS vol. 435, Springer, pp. 416–427, 1989.
12. Y. Dodis, K. Pietrzak and P. Puniya. A new mode of operation for block ciphers and length-preserving MACs. *Advances in Cryptology – EUROCRYPT 2008*, LNCS vol. 4965, Springer, pp. 198–219, 2008.
13. Y. Dodis and J. Steinberger. Message authentication codes from unpredictable block ciphers. *Advances in Cryptology – CRYPTO 2009*, LNCS vol. 5677, Springer, pp. 267–285, 2009.
14. S. Halevi and P. Rogaway. A tweakable cnciphering mode. *Advances in Cryptology – CRYPTO 2003*, LNCS vol. 2729, Springer, pp. 482–499, 2003.
15. S. Halevi and P. Rogaway. A parallelizable enciphering mode. *Topics in Cryptology – CT-RSA 2004*, LNCS vol. 2964, Springer, pp. 292–304, 2004.
16. S. Halevi. Invertible Universal Hashing and the TET Encryption Mode. *Advances in Cryptology – CRYPTO 2007*, LNCS vol. 4622, Springer, pp. 412-429, 2007.
17. D. Goldenberg, S. Hohenberger, M. Liskov, E.C. Schwartz and H. Seyalioglu. On Tweaking Luby-Rackoff Blockciphers. *Advances in Cryptologoy – ASIACRYPT 2007*, LNCS vol. 4833, Springer, pp. 342–356, 2007
18. J. Garay, V. Kolesnikov and R. McLellan. MAC precomputation with applications to secure memory. *12th Information Security Conference – ISC 2009*, LNCS vol. 5735, Springer, pp. 427–442, 2009.
19. E. Jaulmes, A. Joux and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption – FSE 2002*, LNCS vol. 2365, Springer, pp. 237–251, 2002.
20. T. Krovetz and P. Rogaway. The Software Performance of Authenticated-Encryption Modes. *Fast Software Encryption – FSE 2011*, LNCS vol. 6733, Springer, pp. 306–327, 2011.
21. M. Liskov, R. Rivest and D. Wagner, Tweakable block ciphers. *Advances in Cryptology – CRYPTO 2002*, LNCS vol. 2442, Springer, pp. 31–46, 2002.
22. U. Maurer and J. Sjödin. Single-key AIL-MACs from any FIL-MAC. *International Colloquium on Automata, Languages, and Programming – ICALP 2005,* LNCS vol. 3580, Springer, pp. 472–484, 2005.
23. R. Merkle. One way hash functions and DES. *Advances in Cryptology – CRYPTO '89*, LNCS vol. 435, Springer, pp. 428–446, 1989.
24. K. Minematsu. Beyond-birthday-bound security based on tweakable block cipher. *Fast Software Encryption – FSE 2009*, LNCS vol. 5665, Springer, pp. 308–326, 2009.
25. K. Minematsu. How to Thwart Birthday Attacks against MACs via Small Randomness. *Fast Software Encryption – FSE 2010*, LNCS vol. 6147, Springer, pp. 230–249, 2010.
26. B. Preneel, R. Govaerts and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. *Advances in Cryptology – CRYPTO 1993*, LNCS vol. 773, Springer, pp. 368–378, 1993.
27. P. Rogaway, M. Bellare, J. Black and T. Krovetz. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Conference on Computer and Communication Security – CCS 2001*, ACM Press, pp. 196-205, 2001.
28. P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC *Advances in Cryptology – ASIACRYPT 2004*, LNCS vol. 3329, Springer, pp. 13–31, 2004.
29. R. Schroeppel. The hasty pudding cipher. *NIST AES proposal, available at* `http://www.cs.arizona.edu/~rcs/hpc`, 1998.

30. M. Bellare, T. Kohno, S. Lucks, N. Ferguson, B. Schneier, D. Whiting, J. Callas and J. Walker. Provable Security Support for the Skein Hash Family. `http://www.skein-hash.info/sites/default/files/skein-proofs.pdf`
31. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. Internet RFC 4346, 2006.
32. P. Wang, D. Feng and W. Wu. HCTS: A varible-input-length enciphering mode. *Information Security an Cryptology – CISC 2005*, LNCS vol. 3822, Springer, pp. 175-188, 2005.
33. K. Yasuda. The Sum of CBC MACs Is a Secure PRF *Topics in Cryptology – CT-RSA 2010*, LNCS vol. 5985, pp. 366–381, 2010.
34. K. Yasuda. A New Variant of PMAC: Beyond the Birthday Bound *Advances in Cryptology – CRYPTO 2011*, LNCS vol. 6841, pp. 596–607, 2011.
35. L. Zhang, W. Wu, P. Wang, L. Zhang, S. Wu and B. Liang. Constructing rate-1 MACs from related-key unpredictable block ciphers: PGV model revisted. *Fast Software Encryption – FSE 2010*, LNCS vol. 6147, Springer, pp. 250–269, 2010.