Tweakable Blockciphers with Beyond Birthday-Bound Security

Will Landecker[1], Thomas Shrimpton[1], and R. Seth Terashima[1]

Dept. of Computer Science, Portland State University
{landeckw,teshrim,seth}@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.

Keywords: tweakable blockcipher, beyond birthday bound, pseudorandom function, message authentication code, unforgeability.

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 a 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

[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.

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 be more efficient than changing a key.

A NEW CONSTRUCTION WITH BBB SECURITY: CLRW2. Our main technical result is the first TBC construction that has tweakable strong-PRP security beyond the birthday bound, admits essentially arbitrary tweaks, and does not require per-invocation rekeying of any of the underlying objects.We call this the *Chained* LRW2 (CLRW2) construction, since it can be written as $\mathsf{LRW2}[H,E]_{h_2,K_2}(T, \mathsf{LRW2}[H,E]_{h_1,K_1}(T,X))$; see Figure 1.

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 tweakable strong-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.

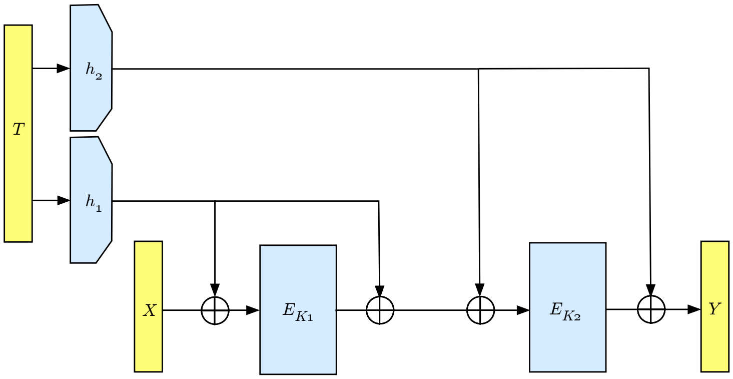

Fig. 1. The CLRW2 Construction.

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 a 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 exhibit a TBC $\widetilde{E}$ that is unforgeable, yet TBC-MAC over $\widetilde{E}$ is easily forged.

UNFORGEABILITY PRESERVATION OF TBC CONSTRUCTIONS. A final, small contribution of this work is to address the question: What if one wants only to assume access to cryptographic primitives that are unforgeable (i.e. unpredictable), rather than pseudorandom? No previous work addresses the provable security of TBC constructions starting from blockciphers with this weaker security assumption. We begin this effort by considering the two TBC constructions from LRW. 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.) For space reasons, these results appear in Appendix E. 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 tweakable strong-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, and thus 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 $X \| Y$ for their concatenation. When $X \in \{0,1\}^*$ we write $|X|$ for its length and, if $1 \leq i < j \leq |X|$ we write $X[i..j]$ for the substring running from its i^{th} to j^{th} characters, or the empty string ε otherwise. 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. For a tuple of strings $(X_1, X_2, \ldots, X_r)$ we define $|(X_1, X_2, \ldots, X_r)| = |X_1 \| X_2 \| \cdots \| X_r|$. 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 strings $X, Y \in (\{0,1\}^n)^+$ we define the predicate $\mathsf{CommonPF}_i(X, Y)$ to be true if and only if X and Y agree on their first i blocks of n-bits, i.e. $X_j = Y_j$ for all $1 \leq j \leq i$ where $X_j, Y_j \in \{0,1\}^n$. When $\mathcal{X} \subseteq (\{0,1\}^n)^+$ and $M \in (\{0,1\}^n)^+$, we also define $\mathsf{Prefix}_{\mathcal{X}}(M)$ to be the function that returns the blockwise longest common prefix that M shares with strings in $\mathcal{X}$. 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

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

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

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

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 $K\xleftarrow{\$}\{0,1\}^k$, $\pi\xleftarrow{\$}\mathrm{Perm}(n)$, and $\mathit{\Pi}\xleftarrow{\$}\mathrm{BC}(\mathcal{T},n)$. Then we define

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

to be (respectively) the PRP, strong PRP, tweakable-PRP, and tweakable strong-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 2-XOR-universal (ϵ-AXU$_2$) if for all distinct $X,X'\in\mathcal{D}$ and $Y\in\mathcal{R}$, $\Pr\Big[\,K\xleftarrow{\$}\mathcal{K}\;:\;F_K(X)\oplus F_K(X')=Y\,\Big]\;\le\;\epsilon.$

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 $E\colon\{0,1\}^k\times\{0,1\}^n\to\{0,1\}^n$ is blockcipher, 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, 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*

$$\mathbf{Adv}_{\widetilde{E}}^{\widetilde{\mathrm{sprp}}}(A)\le 2\mathbf{Adv}_E^{\mathrm{sprp}}(B)+\frac{6q^3\hat{\epsilon}^2}{1-q^3\hat{\epsilon}^2}.\;\blacksquare$$

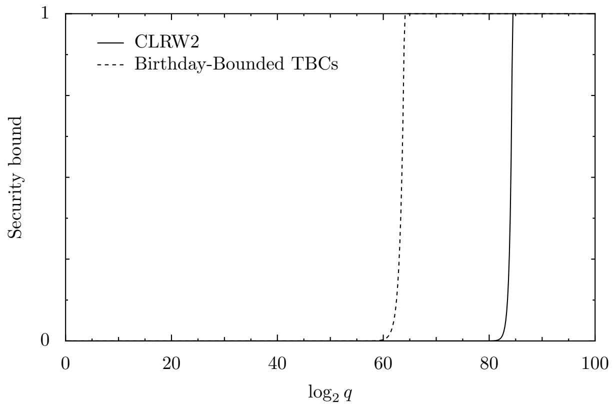

Fig. 2. The maximum advantage of an adversary making q queries agaisnt CLRW2 (solid line) and constructions limited by the birthday bound, $q^2/2^n$ (dashed line). Here, $n = 128$, $\epsilon = 2^{-n}$, and we have assumed the $\mathbf{Adv}_E^{\mathrm{sprp}}(B)$ term is negligible.

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. Now the additive 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 $\mathbf{Adv}_E^{\mathrm{sprp}}(B)$ 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

[2] We note that $\mathbf{Adv}_E^{\mathrm{sprp}}(B)$ 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 ($k = 56, n = 64$).

reversed. 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,

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

Given a set $S \subseteq \{0,1\}^n$ and a string $x \in \{0,1\}^n$ we define $S \oplus x = \{s \oplus x \,:\, s \in S\}$. 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, integers are zero, and all other variable types are undefined (equal to $\perp$).

Game *G1* (refer to Appendix A to see the games used in this proof) simulates $\widetilde{E}$ exactly by lazily sampling values for π_1 and π_2. Note that there is a certain symmetry between the encryption and decryption oracles. This symmetry arises from the fact that $\widetilde{E}$ is the dual of $\widetilde{E}^{-1}$, in the sense that $\widetilde{E}^{-1}_{h_1,h_2,\pi_1,\pi_2}(Y,T) = \widetilde{E}_{h_2,h_1,\pi_2^{-1},\pi_1^{-1}}(Y,T)$.

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). When arguing that transitions between games are correct in this sense, we will exploit the above symmetry 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.

In Game *G2*, we change what happens when there is a collision at the first block cipher: we sample Y_i from the ideal distribution, but raise a bad flag if we also encounter a collision at the input of second block cipher (bad_1) or if Y_i is already in the defined range (bad_2). See Figure 6. Game *G3* is identical to Game *G2*, except Y_i is not reassigned after a bad flag is set. Hence $\Pr\left[\, A^{G1} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G2} \Rightarrow 1 \,\right] \le \Pr\left[\, A^{G3} \Rightarrow 1 \,\right] + \Pr\left[\, A^{G3} \,:\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right]$.

Next we modify the section of code in Game *G3* that is executed when no collision occurs at π_1; i.e., when $X_i \oplus h_1(T_i) \ne X_j \oplus h_1(T_j)$ for all $j < i$. Note that the random variables Z and Y_i are dependent. In Game *G3*, Z is chosen before Y_i, but as long as the joint distribution as preserved we may reverse this order. The resulting game will be equivalent to Game *G3*. As always, the decryption oracle will be modified in a similar manner.

To describe the correct distribution for Y_i, partition $\{0,1\}^n$ into four sets, S_1, S_2, S_3 and S_4. These sets are defined with respect to an oracle query (T_i, X_i) such that no collision occurs at π_1; that is, such that $X_i \oplus h_1(T_i) \notin \mathit{Dom}(\pi_1)$. (When referring to $\mathit{Dom}(\cdot)$ outside of pseudocode, we refer to the set of points at which the function is defined at the instant the adversary makes its i^{th} oracle call [and similarly for $\mathit{Rng}(\cdot)$]; the game currently being used to define the oracle should be clear from context). For $y \in \{0,1\}^n$, we say y is *permissible* when $y \in \mathcal{Y}_i$, and y is *possible* when $\Pr\left[\, Y_i = y \,\right] > 0$, given our assumption that $X_i \oplus h_1(T_i) \notin \mathit{Dom}(\pi_1)$ and the oracles' execution histories for the first $i - 1$ queries.

Let S_4 be the set of all non-permissible values. Note that if y is not permissible (it has been returned on a query that used tweak T_i), then y is not possible (since $\widetilde{E}(T_i, \cdot)$ is a permutation and queries are distinct); hence S_4 is a subset of the impossible values. Let S_3 be the set of impossible values that *are* permissible.

We now subdivide the set of possible values based on the conditional branch on Line 317 in Game *G3*. Some values for Y_i will only be returned if the choice of Z causes a collision at π_2, while others can only be assigned in the absence of such a collision; the former will be S_2, the latter S_1. More formally, one can see that y is not possible if and only if $y \oplus h_2(T_i) \in \mathit{Rng}(\pi_2)$ and $\pi_2^{-1}(y \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i) \in \mathit{Rng}(\pi_1)$ Therefore let $S_1 = \{y \,:\, y \oplus h_2(T_i) \notin \mathit{Rng}(\pi_2)\}$, and let S_2 be the set of all other possible values.

In summary,

$$\begin{aligned}S_1 &= \{y \,:\, y \oplus h_2(T_i) \notin \mathit{Rng}(\pi_2)\} \\ S_2 &= \left\{y \,:\, y \oplus h_2(T_i) \in \mathit{Rng}(\pi_2), \pi_2^{-1}(y \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i) \in \overline{\mathit{Rng}(\pi_1)}\right\} \\ S_3 &= \mathcal{Y}_i \setminus (S_1 \cup S_2) \\ &= \left\{y \,:\, y \oplus h_2(T_i) \in \mathit{Rng}(\pi_2),\ \pi_2^{-1}(y \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i) \in \mathit{Rng}(\pi_1)\right\} \setminus \overline{\mathcal{Y}_i} \\ S_4 &= \overline{\mathcal{Y}_i} = \{Y_j \,:\, j < i, T_j = T_i\}\,.\end{aligned}$$

When these sets are used in pseudocode, it is understood that they are defined at the time the oracle call is made; although $\mathit{Rng}(\pi_1)$ (for example) may change as code executes, S_2 does not change until the next query. When referred to by a decryption oracle, the definitions for these sets are the same up to the previously mentioned duality.

We will now compute the probability that Y_i will be in each of these sets (again, under the assumption that there is no collision at the first block cipher; i.e, that $L_i = X_i \oplus h_1(T_i) \notin \mathit{Dom}(\pi_1)$). Since S_3 and S_4 contain only impossible values, $\Pr[\, Y_i \in S_3 \cup S_4 \mid L_i \notin \mathit{Dom}(\pi_1) \,] = 0$. Let $N = \left|\overline{\mathit{Rng}(\pi_1)}\right|$. Given $y \in S_2$ and $L_i \notin \mathit{Dom}(\pi_1)$, $Y_i = y$ if and only if $Z = \pi_2^{-1}(y \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_2(T_i)$. This value is in $\overline{\mathit{Rng}(\pi_1)}$ by definition of S_2, and so this event happens with probability $1/N$. Hence,

$$\Pr[\, Y_i \in S_2 \mid L_i \notin \mathit{Dom}(\pi_1) \,] = |S_2|/N \quad \text{and} \quad \Pr[\, Y_i \in S_1 \mid L_i \oplus \notin \mathit{Dom}(\pi_1) \,] = (N - |S_2|)/N.$$

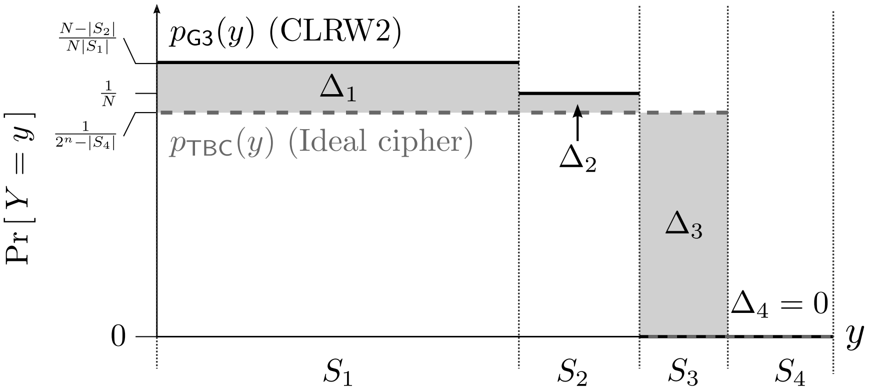

Fig. 3. When there is no collision at π_1, the distribution governing $\widetilde{E}$'s outputs is very close to the distribution an ideal cipher would provide. Horizontal scaling suggests plausible relative sizes of each $|S_k|$: likely $|S_1| \gg |S_2 \cup S_4| \gg |S_3|$. This graph is accurate for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

Ideally, Y_i would be distributed as $p_{\mathsf{TBC}}(y) := \Pr\left[\, Y \xleftarrow{\$} \mathcal{Y}_i \,;\, Y = y \,\right] = 1/(2^n - |S_4|)$ (for $y \notin S_4$) and zero otherwise. However, we have shown that if there is no collision at π_1 on the i^{th} query, then Y_i is distributed as

$$p_{\mathsf{G3}}(y) := \Pr[\, Y_i = y \mid L_i \notin \mathit{Dom}(\pi_1) \,] = \begin{cases} \frac{N-|S_2|}{N|S_1|} & \text{if } y \in S_1 \\ \frac{1}{N} & \text{if } y \in S_2 \\ 0 & \text{if } y \in S_3 \cup S_4 \end{cases}$$

See Figure 3. Now, let $V \xleftarrow{\$} \xi(p)$ denote that the random variable V is equal to one with probability p and is zero otherwise. In Game *G4* (see Figure 7), we decide whether to sample Y_i $\overline{\text{from } S_1}$ or from S_2 based on an appropriately weighted coin flip. It follows that Z is uniformly distributed from $\overline{\mathit{Rng}(\pi_1)}$; letting

$$S' = \left\{ z \in \overline{\mathit{Rng}(\pi_1)} \;:\; z \oplus h_1(T_i) \oplus h_2(T_i) \in \mathit{Dom}(\pi_2) \right\}$$

and $S'' = \overline{\mathit{Rng}(\pi_1)} \setminus S'$, we have

$$\begin{aligned} \Pr[\, Z = z \mid z \in S' \,] &= \Pr[\, V = 1 \wedge Z = z \mid z \in S' \,] + \Pr[\, V = 0 \wedge Z = z \mid z \in S' \,] \\ &= \Pr[\, Z = z | z \in S' \wedge V = 1 \,] \Pr[\, V = 1 \,] \\ &= \frac{1}{|S'|} \frac{|S_2|}{\left|\overline{\mathit{Rng}(\pi_1)}\right|} = \frac{1}{\left|\overline{\mathit{Rng}(\pi_1)}\right|}, \end{aligned}$$

and similarly for $z \in S''$. Therefore $\pi_1(L_i)$ is assigned a uniformly random value from $\overline{Rng(\pi_1)}$, as desired. Games *G3* and *G4* are therefore equivalent.

For $j = 1, 2, 3, 4$, let $\Delta_j = \sum_{y \in S_j} (|p_{\mathsf{G3}}(y) - p_{\mathsf{TBC}}(y)|)$ (see Figure 3). Because total probability is always one, $\Delta_1 + \Delta_2 = \Delta_3$. In Game *G5*, we reverse the order in which Y_i and V_i are sampled. To show that V_i follows the same distribution in Games *G4* and *G5*, we denote the corresponding random variables used by these two games as V_{G4} and V_{G5}, respectively. Let Y' be the value initially assigned to Y_i in Game *G5* (Line 612). Note that $\Pr[\, Y' \in S_3 \,] = \Delta_3$. We have:

$$\Pr[\, V_{G5} = 1 \,] = \Pr[\, Y' \in S_2 \,] + \left(\frac{\Delta_2}{\Delta_1 + \Delta_2}\right) \Pr[\, Y' \in S_3 \,] = \Pr[\, Y' \in S_2 \,] + \Delta_2 = \Pr[\, V_{G4} = 1 \,]\,.$$

The final value for Y_i is in S_2 (S_1) if and only if $V_{G5} = 1$ ($V_{G5} = 0$), in which case it is distributed uniformly among the values of this set. Hence the final value of (Y_i, V_i) has the same distribution in Games *G4* and *G5*, and so the two games are equivalent.

Game *G6* is identical to Game *G5* until bad_3 is set. Therefore $\Pr\left[\, A^{G3} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G5} \Rightarrow 1 \,\right] \leq \Pr\left[\, A^{G6} \Rightarrow 1 \,\right] + \Pr\left[\, A^{G6} : \mathsf{bad}_3 \,\right]$. Similarly, $\Pr\left[\, A^{G3} : \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] = \Pr\left[\, A^{G5} : \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] \leq \Pr\left[\, A^{G6} : \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + \Pr\left[\, A^{G6} : \mathsf{bad}_3 \,\right]$.

At this point, Y_i is always sampled from $\mathcal{Y}_i$, and once assigned, its value is never changed. Consequently, we can move this assignment to outside the **if** block. The resulting game, Game *G7*, is shown in Figure 7. Game *G7* behaves exactly as an ideal TBC, and in particular, $\Pr\left[\, A^{G7} \Rightarrow 1 \,\right] = \Pr\left[\, A^{\Pi(\cdot,\cdot),\Pi^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right]$.

We now give the adversary control over what value is assigned to Y_i (or X_i, in the case of decryption queries) in Game *G8*, but insist that it be in $\mathcal{Y}_i$ or $\mathcal{X}_i$, as appropriate. We also simplify program flow by removing the now-unnecessary variable V_i. Because the adversary can compute $\mathcal{Y}_i$ and $\mathcal{X}_i$, he may simulate the oracles of Game *G7* if desired; hence, he can set the bad flags in Game *G8* with probability at least as high as any adversary 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 the adversary is non-adaptive.

For the rest of this proof, all probabilities will be with respect to the experiment A^{G8} (unless the experiment is explicitly stated).

Let $\mathcal{Q}$ be the event that for there exist i, j, and k (with $j, k \neq i$) such that $X_i \oplus h_1(T_i) = X_j \oplus h_1(T_j)$ and $Y_i \oplus h_2(T_i) = Y_k \oplus h_2(T_k)$. That is, $\mathcal{Q}$ indicates the i^{th} query is responsible for collisions at both π_1 and π_2. Our strategy is to show that $\mathcal{Q}$ is extremely unlikely, since it requires two independent collisions involving a single query. Barring such a query, we can show that the probability of a bad flag being set is very small.

By definition of $\mathcal{Q}$ and the ϵ-AXU$_2$ property of H,

$$\Pr[\, \mathcal{Q} \,] \leq \sum_{i=1}^{q} \sum_{j,k \neq i} \Pr[\, h_1(T_j) \oplus h_1(T_i) = X_j \oplus X_i \,] \Pr[\, h_2(T_k) \oplus h_2(T_i) = Y_k \oplus Y_i \,] < q^3 \epsilon^2.$$

Define $\beta_j = \max_{\widetilde{A}} \left(\Pr\left[\, \widetilde{A}^{G8} : \mathsf{bad}_j \mid \neg\mathcal{Q} \,\right] \right)$ and $\beta_j(i) = \max_{\widetilde{A}} \left(\Pr\left[\, \widetilde{A}^{G8} : \mathsf{bad}_j \text{ on query } i \mid \neg\mathcal{Q} \,\right] \right)$. We consider the event in the latter definition to "trigger" even if it has also triggered on an earlier query. (This definition assumes q is not so large that $\Pr[\, \neg\mathcal{Q} \,] = 0$, but since our bound becomes vacuous before this threshold, this is not an issue.) When bounding $\beta_j(i)$, we will assume the i^{th} query is made to the encryption oracle; as before, the other case is symmetric.

Because bad_2 can only be set if the conditions for $\mathcal{Q}$ are met, we immediately have that $\beta_2 \leq \Pr[\, \mathcal{Q} \,] \leq q^3 \epsilon^2$.

Note that bad_1 is set on query i if and only if there exist $j, k < i$ such that

$$X_i \oplus h_1(T_i) = X_j \oplus h_1(T_j) \quad \text{and} \quad \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i) = \pi_1(L_k) \oplus h_1(T_k) \oplus h_2(T_k),$$

where we remind the reader that $L_i = X_i \oplus h_1(T_i)$. Our goal now is to bound

$$\beta_1(i) = \Pr[\, \exists k < i \,:\, \pi_1(L_i) \oplus \pi_1(L_k) = R(i,k) \mid \exists j < i \,:\, L_i = L_j \wedge \neg\mathcal{Q} \,] \cdot \Pr[\, \exists j < i \,:\, L_i = L_j \mid \neg\mathcal{Q} \,]\,,$$

where for brevity we introduce $R(i,k) = h_1(T_i) \oplus h_2(T_i) \oplus h_1(T_k) \oplus h_2(T_k)$.

Because queries are unique and $\widetilde{E}(T_i, \cdot)$ is a permutation, $L_i = L_j$ is only possible if $T_i \neq T_j$, bringing the ϵ-AXU$_2$ property into scope. Hence

$$\begin{aligned}\Pr\left[\, \exists j < i \,:\, L_i = L_j \mid \neg\mathcal{Q} \,\right] &= \frac{\Pr\left[\, \exists j < i \,:\, L_i = L_j \wedge \neg\mathcal{Q} \,\right]}{\Pr\left[\, \neg\mathcal{Q} \,\right]} \\ &\leq \frac{\Pr\left[\, \exists j < i \,:\, L_i = L_j \,\right]}{1 - q^3\epsilon^2} \leq \frac{q\epsilon}{1 - q^3\epsilon^2}.\end{aligned}$$

We now wish to bound $\Pr\left[\, \exists k < i \,:\, \pi_1(L_i) \oplus \pi_1(L_k) = R(i,k) \mid \exists j < i \,:\, L_i = L_j \wedge \neg\mathcal{Q} \,\right]$ (the other factor in our bound for $\beta_1(i)$), so assume that there is some $j < i$ such that $L_i = L_j$ and that $\neg\mathcal{Q}$.

Fix $k \in [1..i-1]$. Consider the case that $L_i = L_k$. Then $\pi_1(L_i) = R(i,k)$ is equivalent to $h_1(T_i) \oplus h_1(T_k) = h_2(T_i) \oplus h_2(T_k)$. Because queries must respect per-tweak permutivity, $T_i \neq T_k$; hence by the ϵ-AXU$_2$ property of H, in this case $\beta_1(i) \leq \epsilon$.

On the other hand, if $L_i \neq L_k$, we will trace the execution history of the game backwards to when the values eventually assigned to $\pi_1(L_i)$ and $\pi_1(L_k)$ become determined. Define $\mathrm{root}(x) = \min\{m \,:\, L_x = L_m\}$. Let $i' = \mathrm{root}(i)$, and let $k' = \mathrm{root}(k)$. Since $L_i = L_j$ for some $j < i$, it follows that $i' < i$. Therefore, by our assumption that $\mathcal{Q}$ does not occur, there is no $\ell \neq i'$ such that $Y_\ell \oplus h_2(T_\ell) = Y_{i'} \oplus h_2(T_{i'})$. Hence on query i', $\pi_1(L_i)$ is sampled from a set of size at least $2^n - 2q$; this sampling occurs indirectly through the random variable Z, itself sampled either on Line 813 or 836, depending on which oracle receives query i'.

Now we compute when the value of $\pi_1(L_k) = \pi_1(L_{k'})$ is determined. If there is no $\ell < k'$ such that $Y_\ell \oplus h_2(T_\ell) = Y_{k'} \oplus h_2(T_{k'})$, then $\pi_1(L_{k'})$ is likewise sampled indirectly from a set of size at least $2^n - 2q$. However, if such an ℓ exists, then $\pi_1(L_k) = \pi_2^{-1}(Y_{k'} \oplus h_2(T_{k'})) \oplus h_2(T_{k'}) \oplus h_1(T_{k'})$, and we are forced to backtrack further to when $\pi_2^{-1}(Y_\ell \oplus h_2(T_\ell)) = \pi_2^{-1}(Y_{k'} \oplus h_2(T_{k'}))$ was defined. Fortunately, our assumption that the conditions for $\mathcal{Q}$ are not met saves us from having to backtrack far. Let $\ell' = \min\{m \,:\, Y_\ell \oplus h_2(T_\ell) = Y_m \oplus h_2(T_m)\}$. Then $\neg\mathcal{Q}$ implies $\ell' = \mathrm{root}(\ell')$. Hence on query ℓ', $\pi_2^{-1}(Y_{\ell'} \oplus h_2(T_{\ell'})) = \pi_2^{-1}(Y_\ell \oplus h_2(T_\ell))$ is sampled, through Z, from a set of size at least $2^n - 2q$. In the first of these two cases, let $r = k'$; in the second, let $r = \ell'$. After query r completes, the value which will be assigned to $\pi_1(L_k)$ is deterministic.

Suppose without loss of generality that $i' > r$. Then $\pi_1(L_i) = \pi_1(L_k) \oplus R(i,k)$ only if on query i', $\pi_1(L_i) = \pi_1(L_{i'})$ is assigned the unique value that makes the former equation true; this happens with probability at most $1/(2^n - 2q)$.

Let $\hat{\epsilon} = \max(\epsilon, 1/(2^n - 2q))$. Then $\Pr\left[\, \pi_1(L_i) \oplus \pi_1(L_k) = R(i,k) \mid \exists j < i \,:\, L_i = L_j \wedge \neg\mathcal{Q} \,\right] \leq \hat{\epsilon}$. We have

$$\beta_1 \leq \sum_{i=1}^{q} \beta_1(i) \leq \sum_{i=1}^{q} \sum_{k=1}^{i-1} \frac{q\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}} < \frac{q^3\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}^2}.$$

If the encryption oracle query (T_i, X_i, Y_i) would cause bad_3 to be set, then one can see by inspection that the decryption oracle query (T_i, Y_i, X_i) would set bad_1. Therefore the upper bound we derived for $\beta_1(i)$ may also be used for $\beta_3(i)$, and as a consequence, the upper bound for β_1 also bounds β_3.

By the fundamental lemma of game playing,

$$\begin{aligned}\Pr\left[\, A^{\widetilde{E}_{h_1,h_2,\pi_1,\pi_2}(\cdot,\cdot), \widetilde{E}^{-1}_{h_1,h_2,\pi_1,\pi_2}(\cdot,\cdot)} \Rightarrow 1 \,\right] &= \Pr\left[\, A^{\mathit{G1}} \Rightarrow 1 \,\right] \\ &\leq \Pr\left[\, A^{\mathit{G3}} \Rightarrow 1 \,\right] + \Pr\left[\, A^{\mathit{G3}} \,:\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] \\ &\leq \Pr\left[\, A^{\mathit{G6}} \Rightarrow 1 \,\right] + \Pr\left[\, A^{\mathit{G6}} \,:\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + 2\Pr\left[\, A^{\mathit{G6}} \,:\, \mathsf{bad}_3 \,\right] \\ &\leq \Pr\left[\, A^{\mathit{G7}} \Rightarrow 1 \,\right] + \Pr\left[\, A^{\mathit{G8}} \,:\, \mathsf{bad}_1 \vee \mathsf{bad}_2 \,\right] + 2\Pr\left[\, A^{\mathit{G8}} \,:\, \mathsf{bad}_3 \,\right] \\ &\leq \Pr\left[\, A^{\mathit{G7}} \Rightarrow 1 \,\right] + \beta_1 + \Pr\left[\, \mathcal{Q} \,\right] + 2(\beta_1 + \Pr\left[\, \mathcal{Q} \,\right]) \\ &\leq \Pr\left[\, A^{\Pi(\cdot,\cdot), \Pi^{-1}(\cdot,\cdot)} \Rightarrow 1 \,\right] + \frac{6q^3\hat{\epsilon}^2}{1 - q^3\hat{\epsilon}^2}.\end{aligned}$$

Thus by a standard argument, there exists a B such that

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

This completes 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 the values $\widetilde{E}(T_i, X') \oplus \widetilde{E}(T_i, X'')$, 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 β_1 required us to trace back through a game's execution history to determine when π_1 became defined at particular points; 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 $M \mapsto M \parallel 10^r$ 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 Appendix B .

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. ($\mathsf{TBCMAC2}$ 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, $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{TBCMAC2}[\widetilde{E}]}(A) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(B)$. ▌

Proof. The second claim follows immediately from the first, since $\alpha_i = 1$ for all i if A is nonce-respecting. So we prove the first claim.

We omit proof of the standard switch from the complexity-theoretic to the information-theoretic setting, wherein adversary B simulates the PRF experiment for $\mathsf{TBCMAC2}$ over $\widetilde{E}_K$ or $\mathit{\Pi}$, depending upon its own oracle. The remainder of the proof is the core technical piece, which proceeds by a sequence of code-based games.

For ease of notation, we write $\mathsf{TBCMAC2}[\mathit{\Pi}]$ instead of $\mathsf{TBCMAC2}[\mathrm{BC}(n{+}b{+}1, n)]_{\mathit{\Pi}}$ with the understanding that $\mathit{\Pi}$ is a uniform element from $\mathrm{BC}(n{+}b{+}1, n)$. Also, in the psuedocode we use $\mathit{Dom}(\mathit{\Pi}(T, \cdot))$ to denote the set of domain points under the (lazily sampled) random permutation $\mathit{\Pi}(T, \cdot)$ that have been assigned a corresponding range value. Likewise, the set $\overline{\mathit{Rng}}(\mathit{\Pi}(T, \cdot))$ denotes the set of stings in $\{0,1\}^n$ that have not yet been associated to any domain point under $\mathit{\Pi}(T, \cdot)$. Games are shown in Figure 4.

Game $G0$ faithfully simulates $\mathsf{TBCMAC2}[\mathit{\Pi}]$, using lazy sampling to establish the random $\mathit{\Pi}$. Game $G1$ does likewise, but always returns a uniform random string. Letting $\mathsf{bad} = \mathsf{bad}_1 \vee \mathsf{bad}_2$, we see that these games are identical-until-bad, so $\Pr\left[\, A^{G0} \Rightarrow 1 \,\right] - \Pr\left[\, A^{G1} \Rightarrow 1 \,\right] \leq \Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right]$. Since $G1$ returns a random string no matter what the value of bad, we drop unneccessary instructions and move to game $G2$ with $\Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right] = \Pr\left[\, A^{G2} \text{ sets } \mathsf{bad} \,\right] \leq \Pr\left[\, A^{G2} \text{ sets } \mathsf{bad}_1 \,\right] + \Pr\left[\, A^{G2} \text{ sets } \mathsf{bad}_2 \,\right]$. It is easy to see that the first summand is at most $(1/2^{n+1}) \sum_{i=1}^{r} (\alpha_i)(\alpha_i - 1)$ by a union bound. Game $G3$ is the same as $G2$ with the setting of bad_1 removed. Thus we have $\Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right] \leq (1/2^{n+1})(\sum_{i=1}^{r} \alpha_i(\alpha_i - 1)) + \Pr\left[\, A^{G3} \text{ sets } \mathsf{bad}_2 \,\right]$.

The setting of bad_2 is more complicated to analyze, because the adversary controls M^s_b and N^s. We see that bad_2 is set only if for some $1 \leq r < s \leq q$ we have $T^r_{b_r - 1} = T^s_{b_s - 1}$ and $N^r = N^s$. Notice that in terms of setting bad_2 the particular value assigned to $\mathit{\Pi}(T^s_{b_s - 1} \,\|\, N^s, M^s_{b_s})$ is irrelevant. It only matters that the domain point has been previously assigned a value. Moreover, these values are never used in the for-loop (because $N^s = 1 \,\|\, N$). So the adversary could itself have selected up front the values $T^s_{b_s}$ to be returned and, also up front, picked the q pairs $(\mathtt{N}, \mathtt{M})$ that optimize the chance of bad_2. So in $G4$ we no longer sample $T^s_{b_s}$, and no longer return any value; we simply mark domain points as defined. Thus $\Pr\left[\, A^{G3} \text{ sets } \mathsf{bad}_2 \,\right] \leq \Pr\left[\, A^{G4} \text{ sets } \mathsf{bad}_2 \,\right]$. Fix the optimal set of $(\mathtt{N}, \mathtt{M})$ for setting bad_2 in $G4$. Finally in game $G5$ we delay the setting of bad_2 until the end, exchange the procedure F for a for-loop over the fixed nonce-message pairs, and sample the entire ideal cipher $\mathit{\Pi}$ at the beginning instead of using lazy sampling. We note that $G5$ sets bad_2 at least as often as does $G4$, since the latter only requires a tweak collision. Hence $\Pr\left[\, A^{G4} \text{ sets } \mathsf{bad}_2 \,\right] \leq \Pr\left[\, G5 \text{ sets } \mathsf{bad}_2 \,\right]$.

Once $\mathit{\Pi}$ is fixed, the order in which the $(\mathtt{N}^i, \mathtt{M}^i)$, $i \in [s]$, are put through the for-loop does not matter, so we assume that they are ordered lexicographically by their first component. Thus we can think of the messages as being processed in groups "named" by their common nonce value. By assumption, there are r such groups with sizes $\alpha_1, \ldots, \alpha_r$, respectively. Let $\mathsf{Coll}_1, \ldots, \mathsf{Coll}_r$ be the events that bad_2 is set by members of the corresponding groups. Then we have $\Pr\left[\, G5 \text{ sets } \mathsf{bad}_2 \,\right] \leq \sum_{i=1}^{r} \Pr\left[\, \mathsf{Coll}_i \text{ in } G5 \,\right]$. Collecting up results we have

$$\Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right] \leq (2/2^n) \sum_{i=1}^{r} \alpha_i(\alpha_i - 1) + \sum_{i=1}^{r} \Pr\left[\, \mathsf{Coll}_i \text{ in } G5 \,\right] \; .$$

At this point we notice that if $\alpha_1 = \cdots = \alpha_r = 1$, then all terms on the right side are zero. Thus if the attack is nonce-respecting, then our reduction is as tight as possible. In general, every $i \in [r]$ for which $\alpha_i = 1$ contributes zero to the right side. Assume that $\alpha_i > 1$ for some $i \in [r]$. The probability that Coll_i is bounded as follows. Consider any two messages $\mathtt{M}, \mathtt{M}'$ that are associated to the same nonce. By Lemma 2 we know the probability that these collide is at most $(2\ell)^2/2^n$ since each message is at most ℓ blocks long. By taking a union bound over all such pairs of messages we obtain

$$\Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right] \le (2/2^n)\sum_{i=1}^{r}\alpha_i(\alpha_i - 1) + \sum_{i=1}^{r}\binom{\alpha_i}{2}(2\ell+1)(2\ell)/2^n$$

and our proof is complete. □

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 to Appendix D .

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*

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

where B runs in time $t_B \le t$, asks $q_B \le \sigma$ queries totalling $\sigma_B \le 2\sigma$ 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 $\widetilde{F}$,shown in Figure 13, that is unforgeable if some underlying TBC $\widetilde{E}$ is. Liskov et al. [21] provide a TBC $\widetilde{E}$ with the required signature.

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*

$$\widetilde{F}_K(T_L \,\|\, T_R, X_L \,\|\, X_R) = X_L \oplus T_R \,\|\, \widetilde{E}_K(X_L \,\|\, T_L \,\|\, T_R, X_R).$$

Then $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{F}}(A) \le \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B)$ where the resources of adversaries A and B are the same. ▌

Proof. $\widetilde{F}$ is a tweakable blockcipher because, given the tweak T and key K, one can invert $\widetilde{F}_K(T, X) = Y$ by evaluating $X = Y_L \oplus T_R \,\|\, \widetilde{E}_K^{-1}(Y_L \oplus T_R \,\|\, T_L \,\|\, T_R, Y_R)$. Therefore $\widetilde{F}_K(T, \cdot)$ is a permutation for every tweak T.

Now, let B be a forging adversary for $\widetilde{E}$ that runs as follows. Adversary B runs A, responding to A's queries by using its own oracle for $\widetilde{E}$. When A outputs its forgery $((T^*, X^*), Y^*)$, let B output $(X_L^* \,\|\, T_L^* \,\|\, T_R^*, X_R^*, Y_R^*)$ as its own forgery.

If A's forgery is valid, it must be the case that

$$\begin{aligned} Y^* =& \widetilde{F}_K(T^*, X^*) \\ =& X_L^* \oplus T_R^* \,\|\, \widetilde{E}_K(X_L^* \,\|\, T_L^* \,\|\, T_R^*, X_R^*) \\ \text{hence } Y_R^* =& \widetilde{E}_K(X_L^* \,\|\, T_L^* \,\|\, T_R^*, X_R^*) \end{aligned}$$

[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.

```
Games $\boxed{\text{G0}}$, $G1$

procedure $F(N, M)$:
$s \leftarrow s + 1$
$N^s \leftarrow 1 \,\|\, N$
$M^s_1, \ldots, M^s_{b_s} \xleftarrow{n} M^s \leftarrow M$
$T^s_0 \leftarrow IV$
for $i = 1$ to $b_s - 1$ do
  if $M^s_i \in Dom(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$ then
    $T^s_i \leftarrow \Pi(T^s_{i-1} \,\|\, \mathbf{0}, M^s_i)$
  else
    $T^s_i \xleftarrow{\$} \overline{Rng}(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$
  $\Pi(T^s_{i-1} \,\|\, \mathbf{0}, M^s_i) \leftarrow T^s_i$
$\tau^s \leftarrow T^s_{b_s} \xleftarrow{\$} \{0,1\}^n$
if $T^s_{b_s} \in Rng(\Pi(T^s_{b_s-1} \,\|\, N^s, \cdot))$ then
  $\mathsf{bad}_1 \leftarrow \mathsf{true}$ $\boxed{; T^s_{b_s} \xleftarrow{\$} \overline{Rng}(\Pi(T^s_{b_s-1} \,\|\, N^s, \cdot))}$
if $M^s_{b_s} \in Dom(\Pi(T^s_{b_s-1} \,\|\, N^s, \cdot))$ then
  $\mathsf{bad}_2 \leftarrow \mathsf{true}$ $\boxed{; T^s_{b_s} \leftarrow \Pi(T^s_{b_s-1} \,\|\, N^s, M^s_{b_s})}$
$\Pi(T^s_{b_s-1} \,\|\, N^s, M^s_{b_s}) \leftarrow T^s_{b_s}$
$\boxed{\textbf{if } \mathsf{bad}_1 \vee \mathsf{bad}_2 \textbf{ then } \text{Return } T^s_{b_s}}$
Return $\tau^s$
```

```
Games $\boxed{G2}$, $G3$

procedure $F(N, M)$:
$s \leftarrow s + 1$
$N^s \leftarrow 1 \,\|\, N$
$M^s_1, \ldots, M^s_{b_s} \xleftarrow{n} M^s \leftarrow M$
$T^s_0 \leftarrow IV$
for $i = 1$ to $b_s - 1$ do
  if $M^s_i \in Dom(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$ then
    $T^s_i \leftarrow \Pi(T^s_{i-1} \,\|\, \mathbf{0}, M^s_i)$
  else
    $T^s_i \xleftarrow{\$} \overline{Rng}(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$
    $\Pi(T^s_{i-1} \,\|\, \mathbf{0}, M^s_i) \leftarrow T^s_i$
$T^s_{b_s} \xleftarrow{\$} \{0,1\}^n$
$\boxed{\textbf{if } T^s_{b_s} \in Rng(\Pi(T^s_{b_s-1} \,\|\, N^s, \cdot)) \textbf{ then } \mathsf{bad}_1 \leftarrow \mathsf{true}}$
if $M^s_{b_s} \in Dom(\Pi(T^s_{b_s-1} \,\|\, N^s, \cdot))$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
$\Pi(T^s_{b_s-1} \,\|\, N^s, M^s_{b_s}) \leftarrow T^s_{b_s}$
Return $T^s_{b_s}$
```

```
Game $G4$

procedure $F(\mathtt{N}, \mathtt{M})$:
$s \leftarrow s + 1$
$\mathtt{N}^s \leftarrow 1 \,\|\, \mathtt{N}$
$\mathtt{M}^s_1, \ldots, \mathtt{M}^s_{b_s} \xleftarrow{n} \mathtt{M}^s$
$T^s_0 \leftarrow IV$
for $i = 1$ to $b_s - 1$ do
  if $\mathtt{M}^s_i \in Dom(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$ then
    $T^s_i \leftarrow \Pi(T^s_{i-1} \,\|\, \mathbf{0}, \mathtt{M}^s_i)$
  else
    $T^s_i \xleftarrow{\$} \overline{Rng}(\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \cdot))$
    $\Pi(T^s_{i-1} \,\|\, \mathbf{0}, \mathtt{M}^s_i) \leftarrow T^s_i$
if $\mathtt{U}[T^s_{b_s-1} \,\|\, \mathtt{N}^s, \mathtt{M}^s_{b_s}] = \mathsf{defined}$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
$\mathtt{U}[T^s_{b_s-1} \,\|\, \mathtt{N}^s, \mathtt{M}^s_{b_s}] \leftarrow \mathsf{defined}$
```

```
Game $G5$

$\Pi \xleftarrow{\$} \mathrm{BC}(n + b + 1, n)$
for $s = 1$ to $q$ do
  $\mathtt{N}^s \leftarrow 1 \,\|\, \mathtt{N}$
  $\mathtt{M}^s_1, \ldots, \mathtt{M}^s_{b_s} \xleftarrow{n} \mathtt{M}^s$
  $T^s_0 \leftarrow IV$
  for $i = 1$ to $b_s - 1$ do
    $T^s_i \leftarrow \Pi(T^s_{i-1} \,\|\, \mathbf{0}, \mathtt{M}^s_i)$

if $(\exists r \neq s)((T^s_{b_r-1} \,\|\, \mathtt{N}^r) = (T^s_{b_s-1} \,\|\, \mathtt{N}^s))$
  then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
```

Fig. 4. Games for Theorem 3. We write $\mathbf{0}$ for $0 \,\|\, 0^b$, and IV is some fixed string.

It must also be the case that A's final query (T^*, X^*) be a new query. This implies that $(X^*_L \,\|\, T^*_L \,\|\, T^*_R, X^*_R)$ is a new query as well, as we have only rearranged the forgery's n-bit blocks. Clearly, then, B successfully forges whenever A does. Furthermore, B makes the same number of queries as A does. Therefore, if A runs in time t making q queries totalling σ blocks, $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{F}}(A) \leq \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B)$, where the resources of B are $t' = t$, $q' = q$ queries, these totalling $\sigma' = \sigma$ blocks. This is due to the fact that each of A's q queries contain a $2n$-bit tweak and a $2n$-bit message (a total of $4n$ bits per query), and each of B's q queries contain a $3n$-bit tweak and an n-bit message (also totalling $4n$ bits per query). □

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

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.

A Games for Theorem 1

This appendix contains the games used in the proof of Theorem 1.

GAME *G1*

```
Procedure $\widetilde{E}(T, X)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $L_i \leftarrow X_i \oplus h_1(T_i)$
  if $L_i \in Dom(\pi_1)$ then
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    if $M_i \in Dom(\pi_2)$ then
      $Y_i \leftarrow \pi_2(M_i) \oplus h_2(T_i)$
    else
      $Y_i \overset{\$}{\leftarrow} \overline{Rng(\pi_2)} \oplus h_2(T_i)$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  else
    $Z \overset{\$}{\leftarrow} \overline{Rng(\pi_1)}; \pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    if $M_i \in Dom(\pi_2)$ then
      $Y_i \leftarrow \pi_2(M_i) \oplus h_2(T_i)$
    else
      $Y_i \overset{\$}{\leftarrow} \overline{Rng(\pi_2)} \oplus h_2(T_i)$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
return $Y_i$

Procedure $\widetilde{E}^{-1}(T, Y)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $N_i \leftarrow Y_i \oplus h_2(T_i)$
  if $N_i \in Rng(\pi_2)$ then
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    if $M_i \in Rng(\pi_1)$ then
      $X_i \leftarrow \pi_1^{-1}(M_i) \oplus h_1(T_i)$
    else
      $X_i \overset{\$}{\leftarrow} \overline{Dom(\pi_1)} \oplus h_1(T_i)$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  else
    $Z \overset{\$}{\leftarrow} \overline{Dom(\pi_2)}; \pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    if $M_i \in Rng(\pi_1)$ then
      $X_i \leftarrow \pi_1^{-1}(M_i) \oplus h_1(T_i)$
    else
      $X_i \overset{\$}{\leftarrow} \overline{Dom(\pi_1)} \oplus h_1(T_i)$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
return $X_i$
```

Fig. 5. Game *G1* simulates $\widetilde{E}$ by using lazy sampling to define the random permutations.

GAMES $\boxed{G2}$, *G3*

```
Procedure $\widetilde{E}(T, X)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $L_i \leftarrow X_i \oplus h_1(T_i)$
  if $L_i \in Dom(\pi_1)$ then
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    $Y_i \xleftarrow{\$} \mathcal{Y}_i$
    if $M_i \in Dom(\pi_2)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
      $\boxed{Y_i \leftarrow \pi_2(M_i) \oplus h_2(T_i)}$
    else
      if $Y_i \oplus h_2(T_i) \in Rng(\pi_2)$ then
        $\mathsf{bad}_2 \leftarrow \mathsf{true}$
        $\boxed{Y_i \xleftarrow{\$} \overline{Rng(\pi_2)} \oplus h_2(T_i)}$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  else
    $Z \xleftarrow{\$} \overline{Rng(\pi_1)}; \pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    if $M_i \in Dom(\pi_2)$ then
      $Y_i \leftarrow \pi_2(M_i) \oplus h_2(T_i)$
    else
      $Y_i \xleftarrow{\$} \overline{Rng(\pi_2)} \oplus h_2(T_i)$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
return $Y_i$
```

```
Procedure $\widetilde{E}^{-1}(T, Y)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $N_i \leftarrow Y_i \oplus h_2(T_i)$
  if $N_i \in Rng(\pi_2)$ then
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    $X_i \xleftarrow{\$} \mathcal{X}_i$
    if $M_i \in Rng(\pi_1)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
      $\boxed{X_i \leftarrow \pi_1^{-1}(M_i) \oplus h_1(T_i)}$
    else
      if $X_i \oplus h_1(T_i) \in Dom(\pi_1)$ then
        $\mathsf{bad}_2 \leftarrow \mathsf{true}$
        $\boxed{X_i \xleftarrow{\$} \overline{Dom(\pi_1)} \oplus h_1(T_i)}$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  else
    $Z \xleftarrow{\$} \overline{Dom(\pi_2)}; \pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    if $M_i \in Rng(\pi_1)$ then
      $X_i \leftarrow \pi_1^{-1}(M_i) \oplus h_1(T_i)$
    else
      $X_i \xleftarrow{\$} \overline{Dom(\pi_1)} \oplus h_1(T_i)$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
return $X_i$
```

GAME *G4*

```
Procedure $\widetilde{E}(T, X)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $L_i \leftarrow X_i \oplus h_1(T_i)$
  if $L_i \in Dom(\pi_1)$ then
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    $Y_i \xleftarrow{\$} \mathcal{Y}_i$
    if $M_i \in Dom(\pi_2)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else
      if $Y_i \oplus h_2(T_i) \in Rng(\pi_2)$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  else
    $V_i \xleftarrow{\$} \xi(|S_2| \,/\, |\overline{Rng(\pi_1)}|)$
    if $V_i = 1$ then
      $Y_i \xleftarrow{\$} S_2$
    else if $V_i = 0$
      $Y_i \xleftarrow{\$} S_1$
    if $Y_i \in S_2$ then
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    else if $Y_i \in S_1$
      $Z \xleftarrow{\$} \overline{Rng(\pi_1)} \setminus (Dom(\pi_2) \oplus h_2(T_i) \oplus h_1(T_i))$
      $\pi_2(Z \oplus h_1(T_i) \oplus h_2(T_i)) \leftarrow Y_i \oplus h_2(T_i)$
    $\pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
return $Y_i$
```

```
Procedure $\widetilde{E}^{-1}(T, Y)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $N_i \leftarrow Y_i \oplus h_2(T_i)$
  if $N_i \in Rng(\pi_2)$ then
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    $X_i \xleftarrow{\$} \mathcal{X}_i$
    if $M_i \in Rng(\pi_1)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else
      if $X_i \oplus h_1(T_i) \in Dom(\pi_1)$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  else
    $V_i \xleftarrow{\$} \xi(|S_2| \,/\, |\overline{Dom(\pi_2)}|)$
    if $V_i = 1$ then
      $X_i \xleftarrow{\$} S_2$
    else if $V_i = 0$
      $X_i \xleftarrow{\$} S_1$
    if $X_i \in S_2$ then
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    else if $X_i \in S_1$
      $Z \xleftarrow{\$} \overline{Dom(\pi_2)} \setminus (Rng(\pi_1) \oplus h_1(T_i) \oplus h_2(T_i))$
      $\pi_1^{-1}(Z \oplus h_2(T_i) \oplus h_1(T_i)) \leftarrow X_i \oplus h_1(T_i)$
    $\pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
return $X_i$
```

Fig. 6. Game *G2* resamples invalid Y_i values, and so behaves identically to Game *G1*. Game *G3*, which excludes boxed commands, behaves as like an ideal TBC for encryption queries that cause collisions at π_1 (and decryption queries that cause collisions at π_2). Game *G4* behaves identically to Game *G3*, except that the order in which Z and Y_i are sampled is reversed. This happens by assigning Y_i a value based on a weighted coin toss (V_i).

GAMES $\boxed{G5}$, *G6*

```
Procedure $\widetilde{E}(T, X)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $L_i \leftarrow X_i \oplus h_1(T_i)$
  if $L_i \in Dom(\pi_1)$ then
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    $Y_i \xleftarrow{\$} \mathcal{Y}_i$
    if $M_i \in Dom(\pi_2)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else
      if $Y_i \oplus h_2(T_i) \in Rng(\pi_2)$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  else
    $Y_i \xleftarrow{\$} \mathcal{Y}_i$
    if $Y_i \in S_1$ then
      $V_i \leftarrow 0$
    else if $Y_i \in S_2$
      $V_i \leftarrow 1$
    else if $Y_i \in S_3$
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $\boxed{V_i \xleftarrow{\$} \xi(\Delta_2/(\Delta_1 + \Delta_2))}$
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
      // $V_i = \perp$ here in Game G6, so neither
      // branch of the following if block executes
      if $V_i = 1$ then
        $Y_i \xleftarrow{\$} S_2$
      else if $V_i = 0$
        $Y_i \xleftarrow{\$} S_1$
    if $Y_i \in S_2$ then
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    else if $Y_i \in S_1$
      $Z \xleftarrow{\$} \overline{Rng(\pi_1)} \setminus (Dom(\pi_2) \oplus h_2(T_i) \oplus h_1(T_i))$
      $\pi_2(Z \oplus h_1(T_i) \oplus h_2(T_i)) \leftarrow Y_i \oplus h_2(T_i)$
    $\pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
return $Y_i$
```

```
Procedure $\widetilde{E}^{-1}(T, Y)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $N_i \leftarrow Y_i \oplus h_2(T_i)$
  if $N_i \in Rng(\pi_2)$ then
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    $X_i \xleftarrow{\$} \mathcal{X}_i$
    if $M_i \in Rng(\pi_1)$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else
      if $X_i \oplus h_1(T_i) \in Dom(\pi_1)$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  else
    $X_i \xleftarrow{\$} \mathcal{X}_i$
    if $X_i \in S_1$ then
      $V_i \leftarrow 0$
    else if $X_i \in S_2$
      $V_i \leftarrow 1$
    else if $X_i \in S_3$
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $\boxed{V_i \xleftarrow{\$} \xi(\Delta_2/(\Delta_1 + \Delta_2))}$
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
      // $V_i = \perp$ here in Game G6, so neither
      // branch of the following if block executes
      if $V_i = 1$ then
        $X_i \xleftarrow{\$} S_2$
      else if $V_i = 0$
        $X_i \xleftarrow{\$} S_1$
    if $X_i \in S_2$ then
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    else if $X_i \in S_1$
      $Z \xleftarrow{\$} \overline{Dom(\pi_2)} \setminus (Rng(\pi_1) \oplus h_1(T_i) \oplus h_2(T_i))$
      $\pi_1^{-1}(Z \oplus h_2(T_i) \oplus h_1(T_i)) \leftarrow X_i \oplus h_1(T_i)$
    $\pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
return $X_i$
```

Fig. 7. In Game *G5*, the order in which V_i and Y_i are reversed. Game *G6* behaves identically to Game *G5* until bad_3 is set (i.e., until $Y_i \in S_3$).

Game *G7*

```
**Procedure** $\widetilde{E}(T, X)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $Y_i \xleftarrow{\$} \mathcal{Y}_i$
  $L_i \leftarrow X_i \oplus h_1(T_i)$
  **if** $L_i \in \mathit{Dom}(\pi_1)$ **then**
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    **if** $M_i \in \mathit{Dom}(\pi_2)$ **then**
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    **else**
      **if** $Y_i \oplus h_2(T_i) \in \mathit{Rng}(\pi_2)$ **then** $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  **else**
    **if** $Y_i \in S_1$ **then**
      $V_i \leftarrow 0$
    **else if** $Y_i \in S_2$
      $V_i \leftarrow 1$
    **else**
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    **if** $V_i = 1$ **then**
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    **else if** $V_i = 0$
      $Z \xleftarrow{\$} \overline{\mathit{Rng}(\pi_1)} \setminus (\mathit{Dom}(\pi_2) \oplus h_2(T_i) \oplus h_1(T_i))$
      $\pi_2(Z \oplus h_1(T_i) \oplus h_2(T_i)) \leftarrow Y_i \oplus h_2(T_i)$
    $\pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
**return** $Y_i$
```

```
**Procedure** $\widetilde{E}^{-1}(T, Y)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $X_i \xleftarrow{\$} \mathcal{X}_i$
  $N_i \leftarrow Y_i \oplus h_2(T_i)$
  **if** $N_i \in \mathit{Rng}(\pi_2)$ **then**
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    **if** $M_i \in \mathit{Rng}(\pi_1)$ **then**
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    **else**
      **if** $X_i \oplus h_1(T_i) \in \mathit{Dom}(\pi_1)$ **then** $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  **else**
    **if** $X_i \in S_1$ **then**
      $V_i \leftarrow 0$
    **else if** $X_i \in S_2$
      $V_i \leftarrow 1$
    **else**
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    **if** $V_i = 1$ **then**
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    **else if** $V_i = 0$
      $Z \xleftarrow{\$} \overline{\mathit{Dom}(\pi_2)} \setminus (\mathit{Rng}(\pi_1) \oplus h_1(T_i) \oplus h_2(T_i))$
      $\pi_1^{-1}(Z \oplus h_2(T_i) \oplus h_1(T_i)) \leftarrow X_i \oplus h_1(T_i)$
    $\pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
**return** $X_i$
```

Fig. 8. Game *G7* is identical to Game *G6*, but simplifies some of the program flow.

GAME *G8*

```
**Procedure** $\widetilde{E}(T, X, Y)$:
  $i \leftarrow i + 1; X_i \leftarrow X; T_i \leftarrow T$
  $Y_i \leftarrow Y$
  $L_i \leftarrow X_i \oplus h_1(T_i); N_i \leftarrow Y_i \oplus h_2(T_i)$
  **if** $L_i \in Dom(\pi_1)$ **then**
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
    **if** $M_i \in Dom(\pi_2)$ **then**
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    **else**
      **if** $Y_i \oplus h_2(T_i) \in Rng(\pi_2)$ **then** $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_2(M_i) \leftarrow Y_i \oplus h_2(T_i)$
  **else**
    **if** $Y_i \in S_1$ **then**
      $Z \xleftarrow{\$} \overline{Rng(\pi_1)} \setminus (Dom(\pi_2) \oplus h_2(T_i) \oplus h_1(T_i))$
      $\pi_2(Z \oplus h_1(T_i) \oplus h_2(T_i)) \leftarrow Y_i \oplus h_2(T_i)$
    **else if** $Y_i \in S_2$
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    **else**
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $Z \leftarrow \pi_2^{-1}(Y_i \oplus h_2(T_i)) \oplus h_2(T_i) \oplus h_1(T_i)$
    $\pi_1(L_i) \leftarrow Z$
    $M_i \leftarrow \pi_1(L_i) \oplus h_1(T_i) \oplus h_2(T_i)$
**return** $Y_i$

**Procedure** $\widetilde{E}^{-1}(T, Y, X)$:
  $i \leftarrow i + 1; Y_i \leftarrow Y; T_i \leftarrow T$
  $X_i \leftarrow X$
  $N_i \leftarrow Y_i \oplus h_2(T_i); L_i \leftarrow X_i \oplus h_1(T_i)$
  **if** $N_i \in Rng(\pi_2)$ **then**
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
    **if** $M_i \in Rng(\pi_1)$ **then**
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    **else**
      **if** $X_i \oplus h_1(T_i) \in Dom(\pi_1)$ **then** $\mathsf{bad}_2 \leftarrow \mathsf{true}$
      $\pi_1^{-1}(M_i) \leftarrow X_i \oplus h_1(T_i)$
  **else**
    **if** $X_i \in S_1$ **then**
      $Z \xleftarrow{\$} \overline{Dom(\pi_2)} \setminus (Rng(\pi_1) \oplus h_1(T_i) \oplus h_2(T_i))$
      $\pi_1^{-1}(Z \oplus h_2(T_i) \oplus h_1(T_i)) \leftarrow X_i \oplus h_1(T_i)$
    **else if** $X_i \in S_2$
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    **else**
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$
      $Z \leftarrow \pi_1(X_i \oplus h_1(T_i)) \oplus h_1(T_i) \oplus h_2(T_i)$
    $\pi_2^{-1}(N_i) \leftarrow Z$
    $M_i \leftarrow \pi_2^{-1}(N_i) \oplus h_2(T_i) \oplus h_1(T_i)$
**return** $X_i$
```

Fig. 9. Game *G8* gives the advesary control over Y_i values. Such an adversary can set bad flags at least as easily as adversaries for Game *G7* can. Additionally, adversaries for Game *G8* are, without loss of generality, non-adaptive.

B Proof of Theorem 2

```
Games $\boxed{G0}$, $G1$

procedure $F(M)$:
$s \leftarrow s+1$; $M^s \leftarrow M$
$M_1^s, \ldots, M_{b_s}^s \xleftarrow{n} M^s$
$P \leftarrow \mathsf{Prefix}_{\mathcal{M}}(M^s)$; $p \leftarrow |P|$; $T_p^s \leftarrow \mathtt{T}[P]$
for $i = p+1$ to $b_s$
  $T_i^s \xleftarrow{\$} \{0,1\}^n$
  if $M_i^s \in Dom(\Pi(T_{i-1}^s, \cdot))$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{; T_i^s \leftarrow \Pi(T_{i-1}^s, M_i^s)}$
  if $T_i^s \in Rng(\Pi(T_{i-1}^s, \cdot))$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ $\boxed{; T_i^s \xleftarrow{\$} \overline{Rng}(\Pi(T_{i-1}^s, \cdot))}$
  $\Pi(T_{i-1}^s, M_i^s) \leftarrow T_i^s$
  $\mathtt{T}[M_1^s \cdots M_i^s] \leftarrow T_i^s$
$\mathcal{M} \xleftarrow{\cup} M^s$
Return $T_{b_s}^s$
```

```
Game $G2$

procedure $F(M)$:
$s \leftarrow s+1$; $M^s \leftarrow M$
$M_1^s, \ldots, M_{b_s}^s \xleftarrow{n} M^s$
$P \leftarrow \mathsf{Prefix}_{\mathcal{M}}(M^s)$; $p \leftarrow |P|$; $T_p^s \leftarrow \mathtt{T}[P]$
for $i = p+1$ to $b_s$
  $T_i^s \xleftarrow{\$} \{0,1\}^n$
  if $M_i^s \in Dom(\Pi(T_{i-1}^s, \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\Pi(T_{i-1}^s, M_i^s) \leftarrow T_i^s$
  $\mathtt{T}[M_1^s \cdots M_i^s] \leftarrow T_i^s$
$\mathcal{M} \xleftarrow{\cup} M^s$
Return $T_{b_s}^s$
```

```
Game $G3$

procedure $F(M)$:
$s \leftarrow s+1$; $M^s \leftarrow M$
$M_1^s, \ldots, M_{b_s}^s \xleftarrow{n} M^s$
$P \leftarrow \mathsf{Prefix}_{\mathcal{M}}(M^s)$; $p \leftarrow |P|$; $T_p^s \leftarrow \mathtt{T}[P]$
for $i = p+1$ to $b_s$
  if $M_i^s \in Dom(\Pi(T_{i-1}^s, \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\Pi(T_{i-1}^s, M_i^s) \leftarrow \mathsf{defined}$
  $\mathtt{T}[M_1^s \cdots M_i^s] \leftarrow T_i^s \xleftarrow{\$} \{0,1\}^n$
$\mathcal{M} \xleftarrow{\cup} M^s$
Return $T_{b_s}^s$
```

```
Game $G4$

procedure $F(M)$:
$s \leftarrow s+1$; $M^s \leftarrow M$
$M_1^s, \ldots, M_{b_s}^s \xleftarrow{n} M^s$
$P \leftarrow \mathsf{Prefix}_{\mathcal{M}}(M^s)$; $p \leftarrow |P|$; $T_p^s \leftarrow \mathtt{T}[P]$
for $i = p+1$ to $b_s$
  if $M_i^s \in Dom(\Pi(T_{i-1}^s, \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\Pi(T_{i-1}^s, M_i^s) \leftarrow \mathsf{defined}$
  $\mathtt{T}[M_1 \cdots M_i] \leftarrow T_i^s \xleftarrow{\$} \{0,1\}^n$
$\mathcal{M} \xleftarrow{\cup} M^s$
```

```
Game $G5$

for $s = 1$ to $q$
$\mathtt{M}_1^s, \ldots, \mathtt{M}_{b_s}^s \xleftarrow{n} \mathtt{M}^s$
$P \leftarrow \mathsf{Prefix}_{\mathcal{M}}(\mathtt{M}^s)$; $p \leftarrow |P|$; $T_p^s \leftarrow \mathtt{T}[P]$
for $i = p+1$ to $b_s$
  if $\mathtt{M}_i^s \in Dom(\Pi(T_{i-1}^s, \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\Pi(T_{i-1}^s, \mathtt{M}_i^s) \leftarrow \mathsf{defined}$
  $\mathtt{T}[\mathtt{M}_1^s \cdots \mathtt{M}_i^s] \leftarrow T_i^s \xleftarrow{\$} \{0,1\}^n$
$\mathcal{M} \xleftarrow{\cup} \mathtt{M}^s$
```

```
Game $G6$

for all $X \in (\{0,1\}^n)^+)$, $\mathtt{T}[X] \xleftarrow{\$} \{0,1\}^n$
for $s = 1$ to $q$
$\mathtt{M}_1^s, \ldots, \mathtt{M}_{b_s}^s \xleftarrow{n} \mathtt{M}^s$
$P^s \leftarrow \mathsf{Prefix}_{\mathcal{M}}(\mathtt{M}^s)$; $p^s \leftarrow |P^s|$
if $\mathtt{M}_{p^s+1}^s \in Dom(\Pi(\mathtt{T}[P^s], \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
$\Pi(\mathtt{T}[P^s], \mathtt{M}_{p^s+1}^s) \leftarrow \mathsf{defined}$
for $i = p^s + 1$ to $b_s - 1$
  if $\mathtt{M}_{i+1}^s \in Dom(\Pi(\mathtt{T}[\mathtt{M}_1^s \ldots \mathtt{M}_i^s], \cdot))$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\Pi(\mathtt{T}[\mathtt{M}_1^s \ldots \mathtt{M}_i^s], \mathtt{M}_{i+1}^s) \leftarrow \mathsf{defined}$
$\mathcal{M} \xleftarrow{\cup} \mathtt{M}^s$
```

Fig. 10. Games for the proof of Theorem 2. We define $\mathtt{T}[\varepsilon] = 0^n$. The set $\mathcal{M}$ is initially empty.

Proof. We omit proof of the standard switch from the complexity-theoretic to the information-theoretic setting, wherein adversary B simulates the PRF experiment for $\mathsf{TBCMAC}[\widetilde{E}]$ or $\mathsf{TBCMAC}[\mathit{\Pi}]$, depending upon its own oracle. The remainder of the proof is the core technical piece, which proceeds by a sequence of code-based games.

Game $G0$ faithfully implements $\mathsf{TBCMAC}[\mathit{\Pi}]$. Here and throughout we use $Dom(\mathit{\Pi}(T,\cdot))$ to denote the set of domain points under the (lazily sampled) random permutation $\mathit{\Pi}(T,\cdot)$ that have been assigned a corresponding range value. Likewise, the set $\overline{Rng}(\mathit{\Pi}(T,\cdot))$ denotes the set of stings in $\{0,1\}^n$ that have not yet been associated to any domain point under $\mathit{\Pi}(T,\cdot)$. Game $G1$ omits the boxed statements, and hence implements a random function ρ with the same domain as TBCMAC and n-bit outputs. Thus $\Pr\left[\, A^{\mathsf{TBCMAC}[\mathit{\Pi}]} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G0} \Rightarrow 1 \,\right]$ and $\Pr\left[\, A^{\rho} \Rightarrow 1 \,\right] = \Pr\left[\, A^{G1} \Rightarrow 1 \,\right]$, and these games are identical-until-bad [5], so we have that $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{TBCMAC}[\mathit{\Pi}]}(A) \leq \Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right]$. The probability that $\mathsf{bad} \leftarrow \mathtt{true}$ on line 106 of $G1$ is at most $(0+1+\cdots+(\ell q-1))/2^n \leq .5(\ell q)^2/2^n$. So in $G2$ we simply remove the check for colliding range points under $\mathit{\Pi}$ and have the bound $\Pr\left[\, A^{G1} \text{ sets } \mathsf{bad} \,\right] \leq \Pr\left[\, A^{G2} \text{ sets } \mathsf{bad} \,\right] + .5(\ell q)^2/2^n$. Notice that in game $G2$ the actual values assigned to $\mathit{\Pi}$ do not impact the setting of bad. In moving to game $G3$ we simply associate a distinguished value defined to domain points that have been touched, and move the random sampling of T^s_i to the first place that it is needed (line 307). Now, in $G3$ the adversary learns only the output of a random function on input M, i.e. $T^s_{b_s}$ for query M^s; inside the game these returned values are only written into $\mathtt{T}[M^s_1 \cdots M^s_{b_s}]$. But since queries are restricted to be prefix-free, these final values are never again used by the game, so we can simply return to the adversary a "dummy" uniform random point Z^s that is never used elsewhere. We do this in game $G4$. Thus $\Pr\left[\, A^{G3} \text{sets } \mathsf{bad} \,\right] \leq \Pr\left[\, A^{G4} \text{sets } \mathsf{bad} \,\right]$.

At this point the adversary receives values that are independent of the setting of bad, so we can assume that the adversary just picks q random values itself and not bother to return the Z^s at all. Moreover, at this point we can assume without loss that the adversary is deterministic and has hardcoded into it the string of coins that maximize the probability that it sets bad by choice of the queries $M^1,\ldots,M^q$. Let $\mathtt{M}^1,\ldots,\mathtt{M}^\mathtt{q}$ be these queries. This moves us to game $G5$, where the adversary is no longer present, so we replace the procedure $F(M)$ by a for-loop over these fixed messages. We have $\Pr\left[\, A^{G4} \text{sets } \mathsf{bad} \,\right] \leq \Pr\left[\, G5 \text{sets } \mathsf{bad} \,\right]$.

Finally we move to game $G6$. Here we make a number of structural changes that do not impact the probability of setting bad. First we unroll the first loop of the for-loop in $G5$ on line 502. We no longer sample values T^s_i and assign these to $\mathtt{T}$ inside the for-loop, instead we set $\mathtt{T}$ to have uniform random values for every "message" $X \in (\{0,1\}^n)^+$; thus in particular all of the random assignments that were made in $G5$ are made here. Wherever T^s_i would have been used before, we use directly the corresponding values of $\mathtt{T}$. The prefixes P are now indexed by s (i.e. P^s) as are the prefix (block)lengths p. Finally, the inner for-loop from $G5$ is reindexed. It is straightforward to verify that $\Pr\left[\, G5 \text{ sets } \mathsf{bad} \,\right] \leq \Pr\left[\, G6 \text{ sets } \mathsf{bad} \,\right]$.

What remains is an analysis of the ways in which bad can be set in $G6$. There are four cases to consider.

Case 1: For some $1 \leq r < s \leq q$ we have $(\mathtt{T}[P^r], \mathtt{M}^r_{p^r+1}) = (\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1})$. If $P^r = P^s = \varepsilon$ then M^r and M^s differ in their first block, so $\Pr\left[\, (\mathtt{T}[P^r], \mathtt{M}^r_{p^r+1}) = (\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1}) \,\right] = 0$ because $\mathtt{M}^r_1 \neq \mathtt{M}^s_1$. If $P^r = \varepsilon$ and $\mathrm{P}^s \neq \varepsilon$ then $\Pr\left[\, (0^n, \mathtt{M}^r_1) = (\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1}) \,\right] = 2^{-n}$ becauase $\mathtt{T}[P^s]$ is uniformaly random. If $P^r \neq \varepsilon$ and $P^s = \varepsilon$ the same reasoning applies symmetrically. If $P^r \neq \varepsilon$ and $P^s \neq \varepsilon$ then $\Pr\left[\, (\mathtt{T}[P^r], \mathtt{M}^r_{p^r+1}) = (\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1}) \,\right] = 2^{-n}$ unless $P^r = P^s$, so assume this. But then $\mathtt{M}^r_{p^r+1} \neq \mathtt{M}^s_{p^s+1}$, since otherwise there would have been a longer common prefix.

Case 2: For some $1 \leq r < s \leq q$ and $p_r + 1 \leq i \leq b_r - 1$ we have $(\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1}) = (\mathtt{T}[\mathtt{M}^r_1 \cdots \mathtt{M}^r_i], \mathtt{M}^r_{i+1})$. If $P^s = \varepsilon$ then we have $\Pr\left[\, (0^n, \mathtt{M}^s_{p^s+1}) = (\mathtt{T}[\mathtt{M}^r_1 \cdots \mathtt{M}^r_i], \mathtt{M}^r_{i+1}) \,\right] = 2^{-n}$. If $P^s \neq \varepsilon$ then $\Pr\left[\, (\,\right] (\mathtt{T}[P^s], \mathtt{M}^s_{p^s+1}) = (\mathtt{T}[\mathtt{M}^r_1 \cdots \mathtt{M}^r_i], \mathtt{M}^r_{i+1})) = 2^{-n}$ unless $P^s = \mathtt{M}^r_1 \cdots \mathtt{M}^r_i$. But then $\mathtt{M}^s_{p^s+1} \neq \mathtt{M}^r_{i+1}$, since otherwise there would have been a longer common prefix.

Case 3: For some $1 \leq r < s \leq q$ and $p_s + 1 \leq i \leq b_s - 1$ we have $(\mathtt{T}[P^r], \mathtt{M}^s_{p^r+1}) = (\mathtt{T}[\mathtt{M}^s_1 \cdots \mathtt{M}^s_i], \mathtt{M}^s_{i+1})$. This is argued as in case 2, so $\Pr\left[\, (\mathtt{T}[P^r], \mathtt{M}^s_{p^r+1}) = (\mathtt{T}[\mathtt{M}^s_1 \cdots \mathtt{M}^s_i], \mathtt{M}^s_{i+1}) \,\right] \leq 2^{-n}$.

Case 4: For some $1 \leq r < s \leq q$, some $p_r + 1 \leq i \leq b_r - 1$, and some $p_s + 1 \leq j \leq b_s - 1$, we have $(\mathtt{T}[\mathtt{M}^r_1 \cdots \mathtt{M}^r_i], \mathtt{M}^r_{i+1}) = (\mathtt{T}[\mathtt{M}^s_1 \cdots \mathtt{M}^s_j], \mathtt{M}^s_{j+1})$. The probability of this is at most 2^{-n} unless $i = j$ and $\mathtt{M}^r_1 \cdots \mathtt{M}^r_i = \mathtt{M}^s_1 \cdots \mathtt{M}^s_j$. But this leads to contradictory values for p^r and p^s.

Thus in every case the probability of $\mathsf{bad} \leftarrow \mathtt{true}$ is at most 2^{-n}. Since there are at most $.5(\ell q)^2$ opportunities for bad to be set, we have $\Pr\left[\, G6 \text{ sets } \mathsf{bad} \,\right] \leq .5(\ell q)^2/2^n$. Collecting up our results from each game step, we have our claimed bound. □

C Collision Resistance Lemma for TBC-MAC

Here we give a simple result showing that TBC-MAC over an ideal cipher Π is collision resistant. We begin by observing that $\mathsf{TBCMAC}[\mathrm{BC}(n,n)]$ can be viewed as an iterated hash with compression function $f^{\Pi}(T,M) = \Pi(T,M)$. Let H^{Π} be the TBC-MAC construction for a particular $\Pi \in \mathrm{BC}(n,n)$. Then we define the collision-finding advantage of A attacking TBC-MAC as

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{TBCMAC}}(A) = \Pr\left[\, \Pi \xleftarrow{\$} \mathrm{BC}(n,n)\, ;\, (M,M') \xleftarrow{\$} A^{\Pi} \;:\; H^{\Pi}(M) = H^{\Pi}(M') \,\right]$$

and we make the convention that when A outputs (M,M') it has already made all queries necessary to evaluate $H^{\Pi}(M)$ and $H^{\Pi}(M')$. We note that this differs from the usual ideal-cipher notion of collision-resistance because the adversary has access only to Π, and *not* its inverse. This notion will be sufficient for our needs.

Similarly, let $f^{\Pi}(T,M) = \Pi(T,M)$ be the compression function of the TBC-MAC iteration for some particular $\Pi \in \mathrm{BC}(n,n)$. We define the collision-finding advantage of A in attacking this compression function as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{comp}}_{f}(A) = \Pr\Big[\,&\Pi \xleftarrow{\$} \mathrm{BC}(n,n)\, ;\, (T,M),(T'M') \xleftarrow{\$} A^{\Pi} \\ &: ((T,M) \neq (T',M') \wedge f^{\Pi}(T,M) = f^{\Pi}(T',M')) \vee f^{\Pi}(T,M) = 0^n \Big]\end{aligned}$$

where we note that the adversary wins by making a collision (again using only Π, not its inverse) or by finding a preimage of the IV. Now a standard argument about the Merkle-Damgård iteration gives us that for any collision-finding adversary A asking q messages of length at most ℓ blocks, there exists an adversary B such that $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{TBCMAC}}(A) \leq \mathbf{Adv}^{\mathrm{comp}}_{f}(B)$, where B asks at most $q\ell$ queries. Moreover, a simple union bound argument shows that for any adversary B making $q\ell$ queries, we have $\mathbf{Adv}^{\mathrm{comp}}_{f}(B) \leq (q\ell+1)(q\ell)/2^n$. Thus we have the following lemma.

Lemma 2. *Fix $n > 0$ and let A be a collision-finding adversary for TBC-MAC that asks q messages of length at most ℓ blocks. Then $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{TBCMAC}}(A) \leq (q\ell+1)(q\ell)/2^n$.* ▌

D Proof of Theorem 4

Proof. The proof essentially analyzes two cases: either the messages queried by A to its oracle result in some output of $\widetilde{E}$ repeating (possibly across queries), or all values output by $\widetilde{E}$ are distinct. In the latter case, we will show there is an adversary C that immediately turns A's forgery into a forgery for $\widetilde{E}$ (in particular, the last call to $\widetilde{E}$ in evaluating A's forgery). In the former case, we show that there exists an adversary that turns $\widetilde{E}$-collisions into $\widetilde{E}$ forgeries.

Consider Game $G0$ in Figure 11. As the boxed instructions are not executed in this game and various bookkeeping objects (e.g. $\mathcal{W}$, j, bad) are never surfaced, it is clear that the procedure Mac correctly implements the $\mathsf{TBCMAC}^{\mathsf{pf}}[\widetilde{E}]$ oracle expected by A in the UF-CMA experiment. Likewise, the procedure **Finalize** returns 1 iff the "winning" event of the UF-CMA experiment occurs. Thus we have $\mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathsf{TBCMAC}^{\mathsf{pf}}\widetilde{E}}(A) = \Pr\left[\, G0^A \Rightarrow 1 \,\right]$. Now since Games $G0$ and $G1$ are *identical-until-bad* games, the fundamental lemma of game playing [5] gives us that

$$\Pr\left[\, G0^A \Rightarrow 1 \,\right] - \Pr\left[\, G1^A \Rightarrow 1 \,\right] \leq \Pr\left[\, G1^A \text{ sets } \mathsf{bad} \,\right].$$

Let forges be the event that **Finalize** returns 1, i.e. the adversary manages to forge. We note that the event forges does not occur in Game $G1$ if $\mathsf{bad} \leftarrow \mathsf{true}$ because execution is halted in this case.

Now, for $\ell \in [1..\sigma-1]$ let Coll_ℓ denote the event that $\mathsf{bad} \leftarrow \mathsf{true}$ occurs in Game $G1$ on the ℓ-th call to $\widetilde{E}$. Let $\mathsf{Coll} = \mathsf{Coll}_1 \vee \mathsf{Coll}_2 \vee \cdots \vee \mathsf{Coll}_{(\sigma-1)}$, and notice that if Coll is true then exactly one of the Coll_ℓ is true because Game $G1$ halts. Conditioning on Coll and using the above, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathsf{TBCMAC}^{\mathsf{pf}}[\widetilde{E}]}(A) &\leq \Pr_1\left[\, \mathsf{forges} \,\right] + \Pr_1\left[\, \mathsf{Coll} \,\right] \\ &\leq \Pr_1\left[\, \mathsf{forges} \mid \overline{\mathsf{Coll}} \,\right] + \Pr_1\left[\, \mathsf{Coll} \,\right]\end{aligned}$$

where $\Pr_1\left[\,\cdot\,\right]$ is the probability measured when A is in Game $G1$.

```
Game $G0$                                        $\boxed{\text{Game } G1}$

procedure Initialize:
10 $K \xleftarrow{\$} \{0,1\}^k$
11 $T_0 \leftarrow 0^n; \mathcal{W} \leftarrow \{T_0\}; j \leftarrow 0$

procedure Mac(M):
12 $j \leftarrow j+1; M^j \leftarrow M$
13 $M^j_1, \ldots, M^j_{b_j} \xleftarrow{n} M^j$
14 for $i = 1$ to $b_j$ do
15     $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M^j_i)$
16     if $T_i \in \mathcal{W}$ and $\nexists h < j$ such that $\mathsf{CommonPF}_i(M^j, M^h)$
17         then bad ← true; $\boxed{\textsc{Halt}}$
18     $\mathcal{W} \leftarrow \mathcal{W} \cup \{T_i\}$
19 Return $T_{b_j}$

procedure Finalize$(M, \tau)$:
20 $M^q \leftarrow M$
21 if $\exists j \in [1..q-1]$ such that $M^q = M^j$ then return 0
22 $M^q_1, \ldots, M^q_{b_q} \xleftarrow{n} M^q$
23 $T_0 \leftarrow 0^n$
24 for $i = 1$ to $b_q - 1$ do
25     $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M^q_i)$
26     if $T_i \in \mathcal{W}$ and $\nexists h < q$ such that $\mathsf{CommonPF}_i(M^q, M^h)$
27         then bad ← true; $\boxed{\textsc{Halt}}$
28 if $\widetilde{E}_K(T_{b_q-1}, M^q_{b_q}) = \tau$ then Return 1 else Return 0.
```

```
adversary $B_\ell^{\widetilde{E}_K(\cdot,\cdot)}$:
10 $s \leftarrow 1; T_0 \leftarrow 0^n; \mathcal{W} \leftarrow \{T_0\}$

When $A$ asks query $M$
12 $M_1, \ldots, M_b \xleftarrow{n} M$
13 for $i = 1$ to $b$ do
14     if $s = \ell$ then
15         $T \xleftarrow{\$} \mathcal{W}$
16         Return $(T, M_i)$
17     $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M_i)$
18     $\mathcal{W} \leftarrow \mathcal{W} \cup \{T_i\}$
19     $s \leftarrow s + 1$
20 pass $T_b$ to $A$

When $A$ halts execution with output $(M^*, \tau^*)$,
21 $M^*_1, \ldots M^*_{b^*} \xleftarrow{n} M^*$
22 for $i = 1$ to $b^* - 1$ do
23     if $s = \ell$ then
24         $T \xleftarrow{\$} \mathcal{W}$
25         Return $(T, M^*_i)$
26     $T_i \leftarrow \widetilde{E}_K(T_{i-1}, M^*_i)$
27     $\mathcal{W} \leftarrow \mathcal{W} \cup \{T_i\}$
28     $s \leftarrow s + 1$
```

Fig. 11. Games for the proof of Theorem 4. Game $G0$ excludes the boxed text (lines 17 and 27). Game $G1$ includes the boxed text. Adversary B_ℓ is for forging the TBC in the event that $\mathsf{bad} \leftarrow \mathsf{true}$ on the ℓ-th message block in Game $G0$, where $\ell \in [\sigma - 1]$. Recall our convention that all boolean flags are initially false.

Consider the case that the event forges occurs in $G1$. Then necessarily $G1$-line 29 executes (the game did not halt), so in fact $\Pr_1[\,\mathsf{forges}\,] = \Pr_1\left[\,\mathsf{forges} \mid \overline{\mathsf{Coll}}\,\right]$. In this case there exists an obvious $\widetilde{E}$-forging adversary C (based on the code of game G1) that outputs $((T_{b_q-1}, M^q_{b_q}), \tau)$ as its forgery. We claim that if event $(\mathsf{forges} \mid \overline{\mathsf{Coll}})$ occurs in $G1$, then C successfully forges $\widetilde{E}$. To see this, it suffices to show that C never asks $(T_{b_q-1}, M^q_{b_q})$ to its $\widetilde{E}$-oracle during its execution. Assume not, and consider the message blocks $M^h_1, M^h_2, ..., M^h_w$ that immediately preceded the first query of $(T_{b_q-1}, M^q_{b_q})$. Either $M^h_1, \ldots, M^h_w = M^q_1, \ldots, M^q_{b_q-1}$ or $M^h_1, \ldots, M^h_w \neq M^q_1, \ldots, M^q_{b_q-1}$. In the former case M^q is a prefix of some previously queried message, which violates the prefix-free encoding of the messages. In the latter case, Coll_ℓ must have occurred for some ℓ, which contradicts our assumption that $\overline{\mathsf{Coll}}$ is true. Hence $\Pr_1\left[\,\mathsf{forges} \mid \overline{\mathsf{Coll}}\,\right] \leq \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(C)$. The adversary C perfectly simulates an oracle $\mathsf{TBCMAC}^{\mathsf{pf}}[\widetilde{E}]$ for A, making a single query for each block of a message queried by A. Thus C's resources are defined by $t_C = t$, $q_C = \sigma$, and $\sigma_C = 2\sigma$ which comes from the fact that C must query the tweak in addition to the message when simulating $\mathsf{TBCMAC}^{\mathsf{pf}}$ on each block of A's queries.

Thus it remains to bound $\Pr_1[\,\mathsf{Coll}\,]$ to finish our proof. For $\ell \in [1..\sigma - 1]$ let B_ℓ be the adversary described in Figure 11. Recall that if Coll is true, then exactly one of Coll_ℓ is true, and for this ℓ there is some $0 \leq p < \ell$ such that $T_\ell = T_p$. Let $(T_{\ell-1}, M_\ell)$ be the query to $\widetilde{E}$ that gave rise to T_ℓ. In this case, there is a $1/\ell$ chance that B_ℓ correctly guesses the index p. If it does and $(T_{\ell-1}, M_\ell)$ has not already been queried, then $((T_{\ell-1}, M_\ell), T_p)$ is a valid $\widetilde{E}$-forgery. (Note that if $(T_{\ell-1}, M_\ell)$ had already been queried, then $\mathsf{Coll}_{\ell-1}$ would have held, contradicting our assumption that Coll_ℓ holds.)

Therefore, $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B_\ell) = \Pr(B_\ell \text{ correctly guesses } T_p \wedge \mathsf{Coll}_\ell)$. The fact that B_ℓ guesses T_p uniformly from the set $\mathcal{W}$, where $|\mathcal{W}| = \ell$, gives us

$$\begin{aligned}\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B_\ell) &\geq \Pr(B_\ell(\text{ correctly guesses } T_p) \wedge \mathsf{Coll}_\ell) \\ &= \Pr(B_\ell(\text{ correctly guesses } T_p)|\mathsf{Coll}_\ell)\Pr(\mathsf{Coll}_\ell) = \frac{1}{\ell}\Pr(\mathsf{Coll}_\ell).\end{aligned}$$

Thus,

$$\Pr(\mathsf{Coll}) \leq \sum_{\ell=1}^{\sigma-1}\Pr(\mathsf{Coll}_\ell) \leq \left(\sum_{\ell=1}^{\sigma-1}\ell\right)\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B) = \frac{\sigma(\sigma-1)}{2}\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B)$$

where $B = B_\ell$ for the value of ℓ that maximizes the advatange. This B runs in time at most $t_B = \max_\ell(t_\ell) \leq t$, asks at most $q_B = \max_\ell(q_\ell) \leq \sigma$ queries, these totaling at most $\sigma_B = \max_\ell(\sigma_\ell) \leq 2\sigma$ which comes from the fact that B_ℓ must query the tweak in addition to the message when simulating $\mathsf{TBCMAC}^{\mathsf{pf}}$ on each block of A's queries. Pulling together results leads immediately to the theorem statement. □

E LRW1 and LRW2 Do Not Preserve Unforgeability

In this section we show that the two TBC constructions from Liskov et al. [21] do not preserve unforgeability. Specifically: given a PRP-secure blockcipher, we construct an unforgeable blockcipher (with a larger domain) that, when used in either of the constructions from Liskov et al. [21], yields a TBC that is easily forged. Curiously, we use one of the LRW constructions in the first step of the process, i.e. building an unforgeable blockcipher from a PRP.

BREAKING LRW1. For what follows, we refer the reader to Figure 12 for a visual representation of the various constructions. First we construct a blockcipher from a TBC. Let $\widetilde{E}\colon \{0,1\}^n \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher. We define $F^{\widetilde{E}}\colon \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ by $F^{\widetilde{E}}_K(X_L \,\|\, X_R) = X_R \,\|\, \widetilde{E}_K(X_L, X_R)$, where X_L and X_R are the leftmost and rightmost n bits of X, respectively. It is clear that the Feistel-like $F^{\widetilde{E}}$ is invertible for any key K, and therefore a permutation, since the preimage of $Y_L \,\|\, Y_R$ is given by $E^{-1}_K(Y_L, Y_R) \,\|\, Y_L$. Thus $F^{\widetilde{E}}$ is a blockcipher with an n-bit key and $2n$-bit blocksize.

If the underlying TBC $\widetilde{E}$ is unforgeable, then clearly so is the blockcipher $F^{\widetilde{E}}$, since forging the latter implies forging the former. Likewise, it is straightforward to prove that a secure tweakable-PRP is unforgeable. We state these simple results as lemmas; we omit the obivous reductions.

Lemma 3. *Let $\widetilde{E}\colon \{0,1\}^n \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ be a tweakable blockcipher, and let $F^{\widetilde{E}}$ be defined as above. Then $\mathbf{Adv}^{\text{uf-cma}}_{F^{\widetilde{E}}}(A) \leq \mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(B)$, where the resources of A and B are identical.* ▮

Lemma 4. *Fix $n > 0$, and 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 totalling σ blocks. Then there exists adversary B such that $\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(A) \leq \mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(B) + \frac{1}{2^n - q}$, where B runs in time t, asks $q+1$ queries, these totalling σ blocks.* ▮

Thus $F^{\widetilde{E}}$ is unforgeable if $\widetilde{E}$ is a secure tweakable-PRP. Now we leverage the results of Liskov et al. [21] to build a tweakable-PRP from a PRP. Specifically, let $E\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\mathsf{LRW1}[E]\colon \{0,1\}^n \times (\{0,1\}^n \times \{0,1\}^n) \to \{0,1\}^n$ be defined by $\mathsf{LRW1}[E]_K(T,X) = E_K(T \oplus E_K(X))$. LRW show that $\mathbf{Adv}^{\widetilde{\text{prp}}}_{\mathsf{LRW1}[E]}(A) \leq \mathbf{Adv}^{\text{prp}}_{E}(B) + \Theta(Q^2/2^n)$, where A makes Q queries to its oracle, and the resources of B are essentially those of A. This result, combined with Lemmas 3 and 4 gives the following.

Lemma 5. *Fix $n > 0$. Let $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let $\mathsf{LRW1}[E]_K(T,X) = E_K(T \oplus E_K(X))$ be defined as above. Let A run in time t, and ask q queries of total length σ blocks. Then there exists a B such that*

$$\mathbf{Adv}^{\text{uf-cma}}_{F^{\mathsf{LRW1}[E]}}(A) \leq \mathbf{Adv}^{\text{prp}}_{E}(B) + \Theta((q+1)^2/2^n)$$

where B runs in time t, asking $q+1$ queries, these totalling σ blocks. ▮

Now we can proceed to the main point, that the TBC $\mathsf{LRW1}[F^{\mathsf{LRW1}[E]}]$ is easily forged.

Theorem 6. *(*$\mathsf{LRW1}$ *does not preserve unforgeability.) Fix $n > 0$. Let $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let* $\mathsf{LRW1}[E]$ *be the tweakable blockcipher construction defined above. Let blockcipher $F^{\mathsf{LRW1}[E]} : \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ be defined (as above) by*

$$F_K^{\mathsf{LRW1}[E]}(X_L \parallel X_R) = X_R \parallel \mathsf{LRW1}[E]_K(X_L, X_R) = X_R \parallel E_K(X_L \oplus E_K(X_R))).$$

Then there exists an A that achieves $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{LRW1}[F^{\mathsf{LRW1}[E]}]}(A) = 1$ by asking $q = 3$ queries totalling $\sigma = 12$ blocks. ▌

Proof. To avoid writing $\mathsf{LRW1}[F^{\mathsf{LRW1}[E]}]_K$ repeatedly, we let $\widetilde{E}_K$ stand for this. Adversary A asks queries $Y^1 \leftarrow \widetilde{E}_K(0^{2n}, 0^{2n})$, and $Y^2 \leftarrow \widetilde{E}_K(0^{2n}, 1^{2n})$ and then outputs $((T^*, X^*), Y^*)$ where $(T^*, X^*) = (1^n \parallel Y_L^1 \oplus Y_L^2, 1^{2n})$ and $Y^* = Y^1$ as its forgery. We note immediately that the forgery (T^*, X^*) has never been queried, since $T^1 = T^2 = 0^{2n} \neq T^*$.

Now we show that A indeed produces a valid forgery. First, we have that

$$\begin{aligned} Y^1 &= E_K(E_K(0^n)) \parallel E_K(E_K(E_K(E_K(0^n)))), \\ Y^2 &= E_K(1^n \oplus E_K(1^n)) \parallel E_K(1^n \oplus E_K(E_K(1^n \oplus E_K(1^n)))) \end{aligned}$$

and hence

$$\begin{aligned} \widetilde{E}_K(T^*, X^*) &= \widetilde{E}_K(1^n \parallel Y_L^1 \oplus Y_L^2, 1^{2n}) \\ &= \widetilde{E}_K(1^n \parallel E_K(E_K(0^n)) \oplus E_K(1^n \oplus E_K(1^n)), 1^{2n}) \end{aligned}$$

which we must show is exactly $Y^* = Y^1$. Let U denote the output of the first round of E_K calculated in the $\mathsf{LRW1}$ construction, and let V denote the input to the second round of E_K. Then we have both $U^* = X_R^* \parallel E_K(X_L^* \oplus E_K(X_R^*)) = 1^n \parallel E_K(1^n \oplus E_K(1^n))$ and

$$\begin{aligned} V^* &= U_L^* \oplus T_L^* \parallel U_R^* \oplus T_R^* \\ &= 1^n \oplus 1^n \parallel E_K(1^n \oplus E_K(1^n)) \oplus E_K(E_K(0^n)) \oplus E_K(1^n \oplus E_K(1^n) \\ &= 0^n \parallel E_K(E_K(0^n)) \end{aligned}$$

finally giving

$$Y^* = V_R^* \parallel E_K(V_L^* \oplus E_K(V_R^*)) = E_K(E_K(0^n)) \parallel E_K(E_K(E_K(E_K(0^n)))) = Y^1$$

which completes the proof. □

We note that if the two blockcipher calls in $\mathsf{LRW1}$ were keyed with distinct keys, the proof is easily modified to cover this case, too. (We present the result with one key because this is the construction proposed by LRW.)

BREAKING $\mathsf{LRW2}$. Liskov et al. [21] give a second construction that yields a tweakable-PRP from a blockcipher and an ϵ-AXU$_2$ hash function. The $\mathsf{LRW2}$ construction is as follows. Given a blockcipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and an ϵ-AXU$_2$ hash function family H, the construction $\mathsf{LRW2}[H, E] : (\mathcal{K} \times \{0,1\}^n) \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ is defined by $\mathsf{LRW2}[H, E]_{(h,K)}(T, X) = h(T) \oplus E_K(X \oplus h(T))$. We will show that there exists an ϵ-AXU$_2$ hash function family and an unforgeable blockcipher such that $\mathsf{LRW2}$ admits an efficient forging attack that succeeds with high probablility. In particular, fix an ϵ-AXU$_2$ hash function family H mapping from t bits to n bits. Let $H' : \mathcal{K} \times \{0,1\}^t \to \{0,1\}^{2n}$ be the family defined by $h'(T) = 0^n \parallel h(T)$ for each $h \in H$. We immediately have the following lemma, given without proof.

Lemma 6. *Fix $t, n > 0$ and let $H : \mathcal{K} \times \{0,1\}^t \to \{0,1\}^n$ be a family of ϵ-AXU$_2$ hash functions, and let H' be defined as above. Then H' is also ϵ-AXU$_2$.* ▌

This will be our ϵ-AXU$_2$ hash function. For the blockcipher, we reuse the $F^{\mathsf{LRW1}[E]}$ construction. We have already shown in Theorem 5 that this blockcipher is unforgeable if E is a secure PRP. Our next result shows that $\mathsf{LRW2}[H', F^{\mathsf{LRW1}[E]}]$ does not preserve this unforgeability. See Figure 14 for a drawing of the construction.

Theorem 7. *(*$\mathsf{LRW2}$ *does not preserve unforgeability.) Fix* $t, n > 0$*. Let* $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be a blockcipher. Let* $F^{\mathsf{LRW1}[E]}$ *be defined as in Theorem 6. Let* $H\colon \mathcal{K} \times \{0,1\}^t \to \{0,1\}^n$ *be a family of* ϵ*-AXU*$_2$ *hash functions, and let* H' *be defined as above. Then there exists an adversary* A *that asks* $q = 3$ *queries, and achieves* $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{LRW2}[H', F^{\mathsf{LRW1}[E]}]}(A) > 1 - \epsilon$*.* ▌

Proof. To avoid writing $\mathsf{LRW2}[H', F^{\mathsf{LRW1}[E]}]_{(h',K)}$ repeatedly, we write $\overline{K}$ for the pair (h', K), and write $\widetilde{E}_{\overline{K}}$ for $\mathsf{LRW2}[H', F^{\mathsf{LRW1}[E]}]_{(h',K)}$.

Adversary A queries $Y^1 \leftarrow \widetilde{E}_{\overline{K}}(0^{2n}, 0^{2n})$ and $Y^2 \leftarrow \widetilde{E}_{\overline{K}}(1^{2n}, 0^{2n})$. A then forges with $((T^*, X^*), Y^*)$, where $Y^* = Y^2_L \parallel Y^1_L \oplus Y^2_L \oplus Y^2_R$, and $(T^*, X^*) = (0^{2n}, 0^n \parallel Y_1^L \oplus Y_2^L)$.

To aid in our analysis, we note that

$$\widetilde{E}_{\overline{K}}(T, X) = X_R \oplus h(T) \parallel h(T) \oplus E_K(X_L \oplus E_K(X_R \oplus h(T))$$

and so $Y^1 = h(0^{2n}) \parallel h(0^{2n}) \oplus E_K(E_K(h(0^{2n})))$, and $Y^2 = h(1^{2n}) \parallel h(1^{2n}) \oplus E_K(E_K(h(1^{2n})))$.

In order to prove the theorem statement, we must prove that $((T^*, X^*), Y^*)$ is a valid forgery with probability $1 - \epsilon$. To this end, we first show that $\widetilde{E}_{\overline{K}}(T^*, X^*) = Y^*$, and next that (T^*, X^*) is a new query.

To the first point, we note that

$$\begin{aligned}
\widetilde{E}_{\overline{K}}(T^*, X^*) &= \widetilde{E}_{\overline{K}}(0^{2n}, 0^n \parallel Y_1^L \oplus Y_2^L) \\
&= \widetilde{E}_{\overline{K}}(0^{2n}, 0^n \parallel h(0^{2n}) \oplus h(1^{2n})) \\
&= h(0^{2n}) \oplus h(1^{2n}) \oplus h(0^{2n}) \parallel h(0^{2n}) \oplus E_K(0^n \oplus E_K(h(0^{2n}) \oplus h(1^{2n}) \oplus h(0^{2n}))) \\
&= h(1^{2n}) \parallel h(0^{2n}) \oplus E_K(E_K(h(1^{2n}))) \\
&= Y^2_L \parallel Y^1_L \oplus Y^2_L \oplus Y^2_R
\end{aligned}$$

Therefore the event $\widetilde{E}_{\overline{K}}(T^*, X^*) = Y^*$ is always true. To the second point, let new-msg denote the event that (T^*, X^*) is a new query. We note that new-msg occurs if and only if $Y_1^L \oplus Y_2^L \neq 0^n$. Thus we have

$$\Pr[\text{new-msg}] = 1 - \Pr[Y_1^L \oplus Y_2^L = 0^n] = 1 - \Pr[h(0^{2n}) \oplus h(1^{2n}) = 0^n] > 1 - \epsilon.$$

where the probability is taken over the random choice of function $h \in \mathcal{H}$. Thus we have

$$\begin{aligned}
\mathbf{Adv}^{\text{uf-cma}}_{\widetilde{E}}(A) &= \Pr\left[\left(\widetilde{E}_{\overline{K}}(T^*, X^*) = Y^*\right) \wedge \text{new-msg}\right] = \Pr[\text{new-msg}] \\
&> 1 - \epsilon \quad \square
\end{aligned}$$

We note that this counterexample is easily strengthened to the case that H is a collision-resistant hash function, rather than merely ϵ-AXU$_2$.

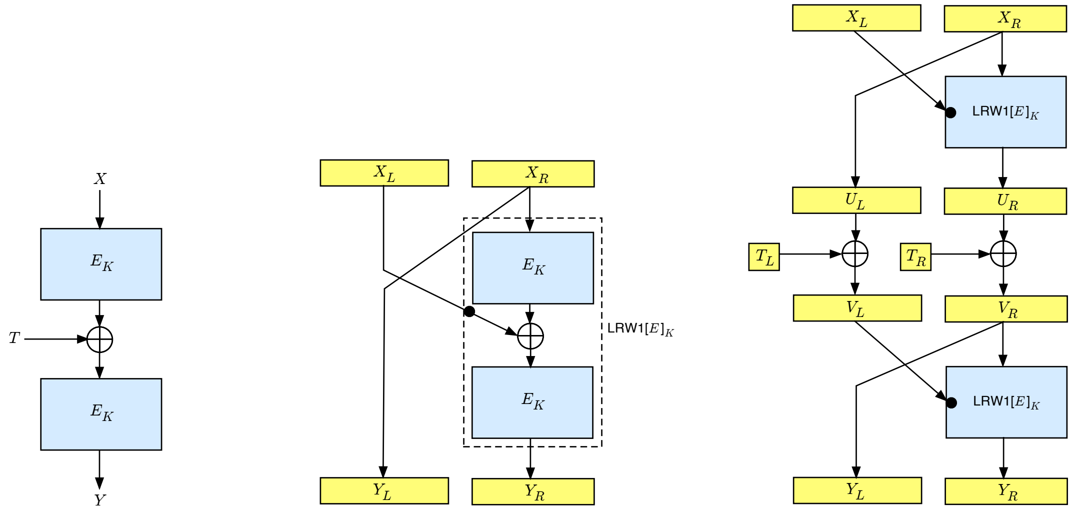

Fig. 12. Figures for Theorem 6. **Left**: The $\mathsf{LRW1}[E]$ construction of an n-bit TBC from an n-bit BC. **Center**: Using $\mathsf{LRW1}[E]$ to build a $2n$-bit BC $F^{\mathsf{LRW1}[E]}$. **Right**: The $\mathsf{LRW1}[F^{\mathsf{LRW1}[E]}]$ TBC construction. The filled-in dot denotes the tweak input.

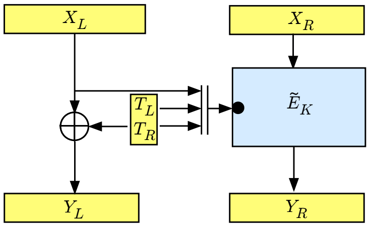

Fig. 13. The $\widetilde{F}$ construction from Theorem 5

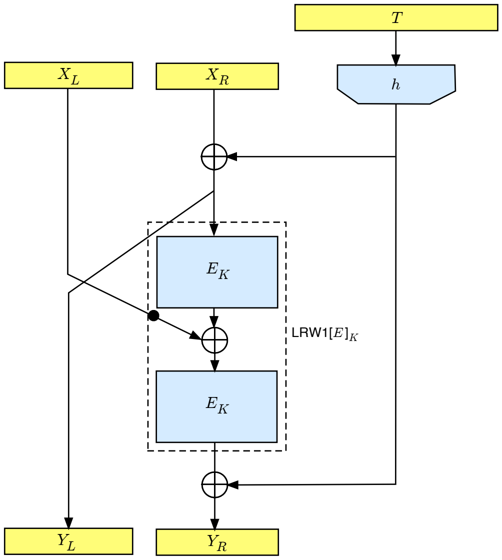

Fig. 14. The $\mathsf{LRW2}[H', F^{\mathsf{LRW1}[E]}]$ TBC construction, used in Theorem 7. Note that $h'(T) = 0^n \,\|\, h(T)$, but for compactness we do not draw the 0^n block or the exclusive-or (with 0^n) operations.