The proceedings version of this paper [15] appears in *Advances in Cryptology — CRYPTO '03*, Lecture Notes in Computer Science, vol. 2729, D. Boneh, ed., Springer-Verlag, 2003. This is the full version.

A Tweakable Enciphering Mode

Shai Halevi* Phillip Rogaway†

June 3, 2003

Abstract

We describe a block-cipher mode of operation, CMC, that turns an n-bit block cipher into a tweakable enciphering scheme that acts on strings of mn bits, where $m \geq 2$. When the underlying block cipher is secure in the sense of a strong pseudorandom permutation (PRP), our scheme is secure in the sense of tweakable, strong PRP. Such an object can be used to encipher the sectors of a disk, in-place, offering security as good as can be obtained in this setting. CMC makes a pass of CBC encryption, xors in a mask, and then makes a pass of CBC decryption; no universal hashing, nor any other non-trivial operation beyond the block-cipher calls, is employed. Besides proving the security of CMC we initiate a more general investigation of tweakable enciphering schemes, considering issues like the non-malleability of these objects.

Key words: Block-cipher usage, cryptographic standards, disk encryption, modes of operation, provable security, sector-level encryption, symmetric encryption.

*IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA, `shaih@watson.ibm.com` `http://www.research.ibm.com/people/s/shaih/`

†Department of Computer Science, University of California, Davis, CA 95616, USA, and Department of Computer Science, Faculty of Science, Chiang Mai University, 50200 Thailand, `rogaway@cs.ucdavis.edu` `http://www.cs.ucdavis.edu/~rogaway`

Contents

1 Introduction

Enciphering schemes. Suppose you want to encrypt the contents of a disk, but the encryption is to be performed by a low-level device, such as a disk controller, that knows nothing of higher-level concepts like files and directories. The disk is partitioned into fixed-length sectors and the encrypting device is given one sector at a time, in arbitrary order, to encrypt or decrypt. The device needs to operate on sectors as they arrive, independently of the rest. Each ciphertext must have the same length as its plaintext, typically 512 bytes. When the plaintext disk sector P is put to the disk media at location T what is stored on the media should be a ciphertext $C = \mathcal{E}_K^T(P)$ that depends not only on the plaintext P and the key K, but also on the location T, which we call the *tweak*. Including the dependency on T allows that identical plaintext sectors stored at different places on the disk will have computationally unrelated ciphertexts.

The envisioned attack-model is a chosen plaintext/ciphertext attack: the adversary can learn the ciphertext C for any plaintext P and tweak T, and it can learn the plaintext P for any ciphertext C and tweak T. Informally, we want a *tweakable*, *strong*, *pseudorandom permutation* (PRP) that operates on a *wide blocksize* (like 512 bytes). We call such an object an *enciphering scheme*. We want to construct the enciphering scheme from a standard block cipher, such as AES, giving a *mode of operation*. The problem is one of current interest for standardization [16]. We seek an algorithm that is simple, and is efficient in both hardware and software.

Naor-Reingold approach. Naor and Reingold give an elegant approach for making a strong PRP on N bits from a block cipher on $n < N$ bits [23, 24]. Their *hash–encipher–hash* paradigm involves applying to the input an *invertible blockwise-universal hash-function*, enciphering the result (say in ECB mode), and then applying yet another invertible blockwise-universal hash-function. Their work stops short of fully specifying a mode of operation, but in [23] they come closer, showing how to make the invertible blockwise-universal hash-function out of an xor-universal hash-function. So the problem, one might imagine, is simply to *instantiate* the approach [23], selecting an appropriate xor-universal hash function from the literature.[1]

It turns out not to be so simple. Despite many attempts to construct a desirable hash function to use with the hash–encipher–hash approach, we could find no desirable realization. We wanted a hash function that was simple and more efficient, per byte, across hardware and software, than AES.[2] The collision bound should be about 2^{-128} (degrading with the length of messages). Many techniques were explored,[3] but nothing with the desired constellation of characteristics was ever found. We concluded that while making a wide-blocksize, strong PRP had "in principal" been reduced to a layer of block-cipher calls plus two "cheap" layers of universal hashing, the story, in practice, was that the "cheap" hashing layers would come to dominate the total cost in hardware, software, or both.

Our contributions. Our main contribution is a simple, practical, completely-specified enciphering mode. CMC starts with a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and turns it into an enciphering

[1] Adding in a tweak is not a problem; it can be done following the approach of [20]. See Section 6 as well.

[2] After all, one could always CBC MAC over AES to create a universal hash function—a possibility that we never convincingly outdid for creating the hash function of hash-encipher-hash. And to "beat" the construction of this paper one would want to hash at least *twice as fast* as AES (i.e., some 7–8 cycle/byte on a Pentium processor [2]).

[3] For example, software implementations of polynomial evaluation [10] in GF(2^{128}) are slower and more complex than one AES call, even when one multiplicand is a key-dependent constant. The Toeplitz construction of [19] is still slower in software. Bernstein's hash-127 [7] needs good hardware support for a fast implementation. Methods like MMH [14] and NH [8] are building blocks (the range and collision probability is not as desired) that achieve good performance only when well-supported by the hardware and when supplemented by complementary techniques.

scheme CMC[E]: $\mathcal{K}' \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{T} = \{0,1\}^n$ and $\mathcal{M}$ contains strings with any number (at least two) of n-bit blocks. See Figures 1 and 2 for a preview. CMC stands for CBC–Mask–CBC.

CMC uses $2m+1$ block-cipher calls. No "non-elementary" operations are used—in particular, no form of universal hashing is employed. The mode is highly symmetric: deciphering is the same as enciphering except that one uses the inverse block cipher E_K^{-1} in place of E_K. We prove that CMC[E] is secure, in the sense of a tweakable, strong PRP. This assumes that E itself is secure as a strong PRP. The actual results are quantitative, with the usual quadratic degradation in security.

Apart from the specific scheme, we investigate, more generally, the underlying goal. We show that being secure as a tweakable, strong, PRP implies the appropriate versions of indistinguishability [3, 13] and non-malleability [4, 12] under a chosen-ciphertext attack. Following Liskov, Rivest and Wagner [20], we show how tweaks can be cheaply added to the untweaked version of the primitive.

Joux's attack. In an earlier, unpublished, manuscript we described a different version of CMC mode [25]. Although the algorithmic change between the old and new mode is small, its consequences are not: the old mode was *wrong*, as recently shown by Antoine Joux [17]. His simple and clever attack is described in Appendix A. In the same appendix we describe the bug in the proof that corresponds to the attack. This paper fixes the mode and its proof.

Other prior work. Efforts to construct a block cipher with a large blocksize from one with a smaller blocksize go back to Luby and Rackoff [21], whose work can be viewed as building a $2n$-bit block cipher from an n-bit one. They also put forward the notion of a PRP and a strong ("super") PRP. The concrete-security treatment of PRPs begins with Bellare, Kilian, and Rogaway [5]. The notion of a tweakable block-cipher is due to Liskov, Rivest and Wagner [20]. Earlier work by Schroeppel describes a block cipher that was already designed to incorporate a tweak [26]. The first attempt to directly construct an nm-bit block cipher from an n-bit one is due to Zheng, Matsumoto and Imai [27], who give a Feistel-type construction. Bellare and Rogaway [6] give an enciphering mode that works on messages of varying lengths but is not a strong PRP. Another enciphering scheme that is potentially a strong PRP appears in unpublished work of Bleichenbacher and Desai [9]. Yet another suggestion we have seen [16] is forward-then-backwards PCBC mode [22]. The mode is easily broken in the sense of a strong PRP, but the possibility of a simple, two-layer, CBC-like mode helped to motivate us. A different approach for disk-sector encipherment is to build a wide-blocksize block cipher from scratch. Such attempts include BEAR, LION, and Mercy [1, 11].

Afterwards. Recent work by the authors has focused on providing a fully parallelizable enciphering scheme having serial efficiency comparable to that of CMC. We shall report on that work elsewhere. The proceedings version of the current paper appears as [15].

2 Preliminaries

Basics. A *message space* $\mathcal{M}$ is a set of strings $\mathcal{M} = \bigcup_{i \in I} \{0,1\}^i$ for some nonempty index set $I \subseteq \mathbb{N}$. A *length-preserving permutation* is a map $\pi: \mathcal{M} \to \mathcal{M}$ where $\mathcal{M}$ is a message space and π is a permutation and $|\pi(P)| = |P|$ for all $P \in \mathcal{M}$. A *tweakable enciphering scheme,* or simply an *enciphering scheme*, is a function $\mathcal{E}: \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ (the key set) is a finite nonempty set and $\mathcal{T}$ (the tweak set) is a nonempty set and $\mathcal{M}$ is a message space and for every $K \in \mathcal{K}$ and $T \in \mathcal{T}$ we have that $\mathcal{E}(K, T, \cdot) = \mathcal{E}_K^T(\cdot)$ is a length-preserving permutation. An *untweakable enciphering scheme* is a function $\mathbb{E}: \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K}$ is a finite nonempty set and $\mathcal{M}$ is message space and $\mathbb{E}(K, \cdot) = \mathbb{E}_K(\cdot)$ is a length-preserving permutation for every $K \in \mathcal{K}$. A *block cipher* is a function $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $n \geq 1$ and $\mathcal{K}$ is a finite nonempty set and $E(K, \cdot) = E_K(\cdot)$ is a permutation for each $K \in \mathcal{K}$. The number n is the *blocksize*. An untweakable enciphering scheme can be regarded as a tweakable enciphering scheme with tweak set $\mathcal{T} = \{\varepsilon\}$ and a block cipher can be regarded as a tweakable enciphering scheme with tweak set $\mathcal{T} = \{\varepsilon\}$ and message space $\mathcal{M} = \{0,1\}^n$. The inverse of an enciphering scheme $\mathcal{E}$ is the enciphering scheme $\mathcal{D} = \mathcal{E}^{-1}$ where $X = \mathcal{D}_K^T(Y)$ if and only if $\mathcal{E}_K^T(X) = Y$. An *adversary* A is a (possibly probabilistic) algorithm with access to some oracles. Oracles are written as superscripts. By convention, the running time of an algorithm includes its description size. We let $\mathrm{Time}_f(\mu)$ be a function that bounds the worst-case time to compute f on strings that total μ bits. We write $\widetilde{O}(f)$ for $O(f(n)\lg(f(n)))$. Constants inside of O and $\widetilde{O}$ notations are absolute constants, depending only on details of the model of computation. If X and Y are strings of possibly different lengths we let $X \leftarrow\!\!\oplus\, Y$ be the string one gets by xoring the shorter string into the *beginning* of the longer string, leaving the rest of the longer string alone.

Security notions. The definitions here are adapted from [5, 20, 21]. When $\mathcal{M}$ is a message space and $\mathcal{T}$ is a nonempty set we let $\mathrm{Perm}(\mathcal{M})$ denote the set of all functions $\pi\colon \mathcal{M} \to \mathcal{M}$ that are length-preserving permutations, and we let $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ denote the set of functions $\pi\colon \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ for which $\pi(T, \cdot)$ is a length-preserving permutation for all $T \in \mathcal{T}$.

Let $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be an enciphering scheme and A be an adversary. We define the *advantage* of A in distinguishing $\mathcal{E}$ from a random, tweakable, length-preserving permutation and its inverse as

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(A) \stackrel{\mathrm{def}}{=} \Pr\left[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot,\cdot)\ \mathcal{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}^{\mathcal{T}}(\mathcal{M}):\ A^{\pi(\cdot,\cdot)\ \pi^{-1}(\cdot,\cdot)} \Rightarrow 1\right]$$

The notation above shows, in the brackets, an experiment to the left of the colon and an event to the right of the colon. We are looking at the probability of the indicated event after performing the specified experiment. By $A \Rightarrow 1$ we mean the event that A outputs the bit 1. Often we omit writing the experiment, the oracle, or the placeholder-arguments of the oracle. The tilde above the "prp" serves as a reminder that the prp is tweakable, while the $\pm$ symbol in front of the "prp" serves as a reminder that this is the "strong" (i.e., chosen plaintext/ciphertext attack) notion of security. Thus we omit the tilde for untweakable enciphering schemes and block ciphers, and we omit the $\pm$ sign to mean that the adversary is given only the first oracle from each pair.

For each "advantage notion" $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}$ we write $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(\mathcal{R})$ for the maximal value of $\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(A)$ over all adversaries A that use resources at most $\mathcal{R}$. Resources of interest are the running time t, the number of queries q, the total length of all queries μ (sometimes written as $\mu = n\sigma$ when μ is a multiple of some number n), and the length of the adversary's output ς. The name of an argument (t, t', q, etc.) will be enough to make clear what resource it refers to.

Pointless queries. There is no loss of generality in the definitions above to assume that regardless of responses that adversary A might receive from an arbitrary pair of oracles, it never repeats a query (T, P) to its left oracle, never repeats a query (T, C) to its right oracle, never asks its right oracle a query (T, C) if it earlier received a response of C to a query (T, P) from its left oracle, and never asks its left oracle a query (T, P) if it earlier received a response of P to a query (T, C) from its right oracle. We call such queries *pointless* because the adversary "knows" the answer that it should receive. A query is called *valid* if it is well-formed and not pointless. A sequence of queries and their responses is valid if every query in the sequence is valid. We assume that adversaries ask only valid queries.

The finite field $GF(2^n)$. We may think of an n-bit string $L = L_{n-1} \ldots L_1 L_0 \in \{0,1\}^n$ in any of the following ways: as an abstract point in the finite field $\mathrm{GF}(2^n)$; as the number in $[0..2^n - 1]$

whose n-bit binary representation is L; and as the polynomial $L(\mathsf{x}) = L_{n-1}\mathsf{x}^{n-1} + \cdots + L_1\mathsf{x} + L_0$. To add two points, $A \oplus B$, take their bitwise xor. To multiply two points we must fix an irreducible polynomial $P_n(\mathsf{x})$ having binary coefficients and degree n: say the lexicographically first polynomial among the irreducible degree-n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated polynomial is $P_{128}(\mathsf{x}) = \mathsf{x}^{128} + \mathsf{x}^7 + \mathsf{x}^2 + \mathsf{x} + 1$. Now multiply $A(\mathsf{x})$ and $B(\mathsf{x})$ by forming the degree $2n-2$ (or less) polynomial that is their product and taking the remainder when this polynomial is divided by $P_n(\mathsf{x})$.

Often there are simpler ways to multiply in $\mathrm{GF}(2^n)$ than the definition above might seem to suggest. In particular, given L it is easy to "double" L. We illustrate the procedure for $n = 128$, in which case $2L = L \ll 1$ if $\mathsf{firstbit}(L) = 0$, and $2L = (L \ll 1) \oplus \mathrm{Const87}$ if $\mathsf{firstbit}(L) = 1$, where Const87 is $0^{120}10000111$. Here $\mathsf{firstbit}(L)$ means L_{n-1} and $L \ll 1$ means $L_{n-2}L_{n-3} \cdots L_1 L_0\, 0$.

3 Specification of CMC Mode

We construct from block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable enciphering scheme that we denote by CMC-E or CMC$[E]$. The enciphering scheme has key space $\mathcal{K} \times \mathcal{K}$. It has tweak space $\mathcal{T} = \{0,1\}^n$. The message space $\mathcal{M} = \bigcup_{m \geq 2} \{0,1\}^{mn}$ contains any string having any number m of n-bit blocks, where $m \geq 2$. We specify in Figure 1 both the forward direction of our construction, $\mathcal{E} = \mathrm{CMC}\text{-}E$, and its inverse $\mathcal{D}$. An illustration of CMC mode is given in Figure 2. In the figures, all capitalized variables except for K and $\widetilde{K}$ are n-bit strings (keys K and $\widetilde{K}$ are elements of $\mathcal{K}$). Variable names P, C, and M are meant to suggest *plaintext*, *ciphertext*, and *mask*. When we write $\mathcal{E}_K^T(P_1 \cdots P_m)$ we mean that the incoming plaintext $P = P_1 \cdots P_m$ is silently partitioned into n-bit strings $P_1, \ldots, P_m$ (and similarly when we write $\mathcal{D}_K^T(C_1 \cdots C_m)$). It is an error to provide $\mathcal{E}$ (or $\mathcal{D}$) with a plaintext (or ciphertext) that is not mn bits for some $m \geq 2$.

4 Discussion

Basic observations. Deciphering $C = \mathcal{E}_{K\widetilde{K}}^T(P)$ produces the same mask M as enciphering P because $CCC_1 \oplus CCC_m = (PPP_1 \oplus M) \oplus (PPP_m \oplus M) = PPP_1 \oplus PPP_m$. Also note that the multiply by two in computing M cannot be dispensed with; if it were, CCC_m would not depend on PPP_1 so the mode could not be a PRP.

The CMC core. Consider the untweakable enciphering scheme $\mathbb{CMC}$ one gets by ignoring T and setting $\mathbb{T}$ to 0^n in Figures 1 and 2. The CMC algorithm can then be viewed as taking $\mathbb{CMC}$ and "adding in" a tweak according to the construction $\mathrm{CMC}_{K\widetilde{K}}^T(P) = \mathbb{T} \leftarrow\!\!\oplus\ \mathbb{CMC}_K(P \leftarrow\!\!\oplus\ \mathbb{T})$ where $\mathbb{T} = E_{\widetilde{K}}(T)$. A similar approach to modifying an untweakable enciphering scheme to create a tweakable one was used by Liskov, Rivest, and Wagner [20, Theorem 2]. See Section 6.

Symmetry. Encryption under CMC is the same as decryption under CMC except that E_K is swapped with E_K^{-1} (apart from the computation of $\mathbb{T}$). Pictorially, this high degree of symmetry can be seen by observing that if the picture in Figure 2 is rotated 180 degrees it is unchanged, apart from swapping letters P and C. Symmetry is a useful design heuristic in trying to achieve strong PRP security, as the goal itself provides the adversary with capabilities that are invariant with respect to replacing an enciphering scheme $\mathcal{E}$ by its inverse $\mathcal{D}$.

Notice that output blocks in CMC mode are taken in reverse order from the input blocks (meaning that $CCC_i = PPP_{m+1-i} \oplus M$ instead of $CCC_i = PPP_i \oplus M$). This was done for purposes of symmetry: if one had numbered output blocks in the "forward" direction then deciphering would be quite different from enciphering. As an added benefit, the reverse-numbering may improve the

Algorithm $\mathcal{E}^T_{K\widetilde{K}}(P_1 \cdots P_m)$

$\mathbb{T} \leftarrow E_{\widetilde{K}}(T)$
$PPP_0 \leftarrow \mathbb{T}$
for $i \leftarrow 1$ **to** m **do**
  $PP_i \leftarrow P_i \oplus PPP_{i-1}$
  $PPP_i \leftarrow E_K(PP_i)$

$M \leftarrow 2\,(PPP_1 \oplus PPP_m)$
for $i \in [1 \mathinner{..} m]$ **do**
  $CCC_i \leftarrow PPP_{m+1-i} \oplus M$

$CCC_0 \leftarrow 0^n$
for $i \in [1 \mathinner{..} m]$ **do**
  $CC_i \leftarrow E_K(CCC_i)$
  $C_i \leftarrow CC_i \oplus CCC_{i-1}$

$C_1 \leftarrow C_1 \oplus \mathbb{T}$
return $C_1 \cdots C_m$

Algorithm $\mathcal{D}^T_{K\widetilde{K}}(C_1 \cdots C_m)$

$\mathbb{T} \leftarrow E_{\widetilde{K}}(T)$
$CCC_0 \leftarrow \mathbb{T}$
for $i \leftarrow 1$ **to** m **do**
  $CC_i \leftarrow C_i \oplus CCC_{i-1}$
  $CCC_i \leftarrow E_K^{-1}(CC_i)$

$M \leftarrow 2\,(CCC_1 \oplus CCC_m)$
for $i \in [1 \mathinner{..} m]$ **do**
  $PPP_i \leftarrow CCC_{m+1-i} \oplus M$

$PPP_0 \leftarrow 0^n$
for $i \in [1 \mathinner{..} m]$ **do**
  $PP_i \leftarrow E_K^{-1}(PPP_i)$
  $P_i \leftarrow PP_i \oplus PPP_{i-1}$

$P_1 \leftarrow P_1 \oplus \mathbb{T}$
return $P_1 \cdots P_m$

Figure 1: Enciphering (left) and deciphering (right) under $\mathcal{E} = \mathrm{CMC}[E]$, where $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher. The tweak is $T \in \{0,1\}^n$ and the plaintext is $P = P_1 \cdots P_m$ and the ciphertext is $C = C_1 \cdots C_m$.

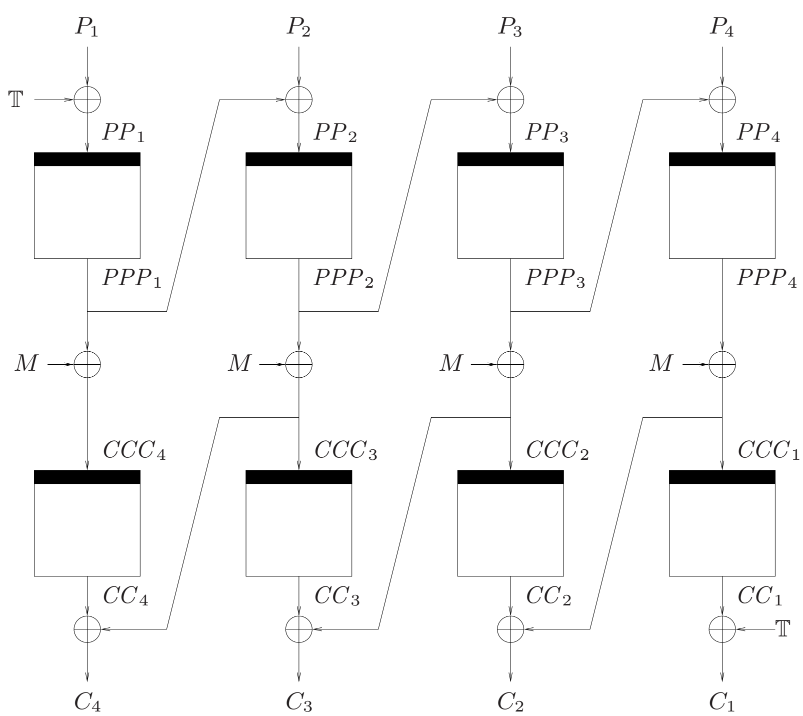

Figure 2: Enciphering under CMC mode for a message of $m = 4$ blocks. The boxes represent E_K. We set mask $M = 2\,(PPP_1 \oplus PPP_m)$. This value can also be computed as $M = 2\,(CCC_1 \oplus CCC_m)$. We set $\mathbb{T} = E_{\widetilde{K}}(T)$ where T is the tweak.

cache-interaction characteristics of CMC by improving locality of reference. That said, an application is always free to write its output according to whatever convention it wishes, and an application with limited memory may prefer to write its output as $C_m \cdots C_1$.

Re-orienting the bottom layer. It is tempting to orient the second block-cipher layer in the opposite direction as the first, thinking that this improves symmetry. But if one were to use E_K^{-1} in the second layer then CMC would become an involution, and thus easily distinguishable from a random permutation.

Limitations. CMC has the following limitations: (1) The mode is not parallelizable. (2) The sector size must be a multiple of the blocksize. (3) In order to make due with $2m+1$ block-cipher calls one needs $\Theta(nm)$ bits of extra memory. Alternatively, one can use $\Theta(n)$ bits of memory, but then one needs $3m+1$ block-cipher calls and one should output the blocks in reverse order. (4) The key for CMC is longer than the key for the underlying block cipher; to keep things simple, we have done nothing to "collapse keys" for this mode. (5) Both directions of the block cipher are used to decipher, due to the one block-cipher call used for producing $\mathbb{T}$ from T.

All of the above limitations could potentially be addressed. In particular, in Appendix B we show a variant of CMC that does not suffer from the last two limitations. Further limitations are inherent characteristics of the type of object that is being constructed. Namely: (a) a good PRP necessarily achieves less than semantic security: repetitions of plaintexts that share a tweak are manifest in the ciphertexts. (b) A PRP must process the entire plaintext before emitting the first bit of ciphertext (and it must process the entire ciphertext before emitting the first block of plaintext). Depending on the context, these limitations can be significant.

5 Security of CMC

The concrete security of the CMC is summarized in the following theorem. The theorem relates the advantage that an adversary has in attacking CMC-E to the advantage that an adversary can get in attacking the underlying block cipher E.

Theorem 1 [CMC security] Fix $n, t, q \geq 1$, $m \geq 2$, and a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let message space $\mathcal{M} = \{0,1\}^{mn}$ and let $\sigma = mq$. Let CMC and $\mathbb{CMC}$ be the modes with the indicated message space. Then

$$\mathbf{Adv}^{\pm\mathrm{prp}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(n\sigma) \leq \frac{5\,\sigma^2}{2^n} \tag{1}$$

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{CMC}[\mathrm{Perm}(n)]}(n\sigma) \leq \frac{7\,\sigma^2}{2^n} \tag{2}$$

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{CMC}[E]}(t, n\sigma) \leq \frac{7\,\sigma^2}{2^n} + 2\,\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t', 2\sigma) \tag{3}$$

where $t' = t + O(n\sigma)$. $\square$

Although we defined CMC and $\mathbb{CMC}$ to have message space $\bigcup_{m\geq 2}\{0,1\}^{mn}$ the theorem restricts messages to one particular length, mn bits for some m. In other words, proven security is for a fixed-input-length (FIL) cipher and not a variable-input-length (VIL) one. We believe that, in fact, security also holds in the sense of a VIL cipher, but we do not at this time provide a proof. All other results in this paper are done for arbitrary (VIL) message spaces.

The heart of Theorem 1 is Equation (1), which is proven in Appendix D. Equation (2) follows immediately using Theorem 2, as given below. Equation (3) embodies the standard way to pass from the information-theoretic setting to the complexity-theoretic one.

Since the proof of Equation (1) is long, let us try to get across some basic intuition for it. Refer to Figure 2 (but ignore the $\mathbb{T}$, as we are only considering $\mathbb{CMC}$). Suppose the adversary asks to encipher some new four-block plaintext P. Plaintext P must be different from all previous plaintexts, so it has some first block where it is different, say P_3. This will usually result in PP_3 being *new*—some value not formerly acted on by the block cipher π. This, in turn, will result in PPP_3 being nearly uniform, and this will propagate to the right, so that PPP_4 will be nearly uniform as well. The values PPP_1 and PPP_2 will usually have been different from each other, and they'll usually be different from the freshly chosen PPP_3 and PPP_4 values. Now $M = 2(PPP_1 \oplus PPP_4)$ and so M will be nearly uniform due to the presence of PPP_4. When we add M to the PPP_i values we will get a bunch of sums CCC_i that are almost always new and distinct. This in turn will cause the vector of CC_i-values to be uniform, which will cause C to be uniform. The argument for a decryption query is symmetric.

Though it is ultimately the above intuition that the proof formalizes, one must be careful, as the experience with the Joux-attack drives home [17]. One must be sure that an adversary cannot, by cutting and pasting parts of plaintexts and ciphertexts, force any nontrivial repetitions in intermediate values.

6 Transforming an Untweakable Enciphering Scheme to a Tweakable One

Let $E\colon \widetilde{\mathcal{K}} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathbb{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be an untweakable enciphering scheme where the message space $\mathcal{M}$ contains no string of length less than n bits. We construct a tweakable enciphering scheme $\mathcal{E} = \mathbb{E} \triangleleft E$ where $\mathcal{E}\colon (\mathcal{K} \times \widetilde{\mathcal{K}}) \times \{0,1\}^n \times \mathcal{M} \to \mathcal{M}$. The construction is $\mathcal{E}^T_{K\widetilde{K}}(M) = \mathbb{T} \leftarrow\!\!\oplus\ \mathbb{E}_K(M \leftarrow\!\!\oplus\ \mathbb{T})$ where $\mathbb{T} = E_{\widetilde{K}}(T)$. (Recall that $\leftarrow\!\!\oplus$ just means to xor in the shorter string at the beginning.) Notice that the cost of adding in the tweak is one block-cipher call and two n-bit xors, regardless of the length of the sector being enciphered or deciphered. Also notice that $\mathrm{CMC} = \mathbb{CMC} \triangleleft E$.

The specified construction is similar to that of Liskov, Rivest and Wagner [20, Theorem 2] but, instead of a PRP E, those authors used an xor-universal hash function. One can view a secure block cipher as being "computationally" xor-universal, and try to conclude the security of the construction in that way. But we have also broadened the context to include enciphering schemes whose input is not a string of some fixed length, and so it seems better to prove the result from scratch. We show that $\mathcal{E} = \mathbb{E} \triangleleft E$ is secure (as a tweakable, strong, enciphering scheme) as long as $\mathbb{E}$ is secure (as an untweakable, strong enciphering scheme) and E is secure (as a PRP). The proof is given in Appendix E.

Theorem 2 [Adding in a tweak] Let $E\colon \widetilde{\mathcal{K}} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let $\mathbb{E}\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be an untweakable enciphering scheme whose message space $\mathcal{M}$ has a shortest string of $N \geq n$ bits. Then

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{E}\triangleleft E}(t, q, \mu) \;\leq\; \frac{q^2}{2^n} + \frac{q^2}{2^N} + \mathbf{Adv}^{\pm\mathrm{prp}}_{\mathbb{E}}(t', q, \mu) + \mathbf{Adv}^{\mathrm{prp}}_{E}(t', q) \tag{4}$$

where $t' = t + \widetilde{O}(\mu + q\mathrm{Time}_E + \mathrm{Time}_{\mathbb{E}}(\mu))$. □

7 Indistinguishability and Nonmalleability of Tweakable Enciphering Schemes

The definition we have given for the security of an enciphering scheme is simple and natural, but it is also quite far removed from any natural way to say that an encryption scheme does what it should do. In this section we explore two notions of security that speak more directly about the

When query	gets an answer of	then these queries are no longer allowed:
$\mathcal{E}(T_0, P_0;\ T_1, P_1)$	C	$\mathcal{D}(T_0, C,\ \cdot, \cdot)$ $\quad \mathcal{D}(\cdot, \cdot,\ T_1, C)$ $\mathcal{E}(T_0, P_0,\ \cdot, \cdot)$ $\quad \mathcal{E}(\cdot, \cdot,\ T_1, P_1)$
$\mathcal{D}(T_0, C_0,\ T_1, C_1)$	P	$\mathcal{E}(T_0, P,\ \cdot, \cdot)$ $\quad \mathcal{E}(\cdot, \cdot,\ T_1, P)$ $\mathcal{D}(T_0, C_0,\ \cdot, \cdot)$ $\quad \mathcal{D}(\cdot, \cdot,\ T_1, C_1)$

Table 1: Disallowed queries. The dot refers to an arbitrary argument—all are disallowed.

privacy and integrity of an enciphering scheme. First we give a definition of *indistinguishability* and then we give a definition for the *nonmalleability*. We show that, as one would expect, security in the sense of a tweakable PRP implies both of these notions, and by tight reductions.

Indistinguishability. To define the indistinguishability of a tweakable enciphering scheme $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ we adapt the left-or-right notion from [3]. We imagine the following game. At the onset of the game we select at random a key K from $\mathcal{K}$ and a bit b. The adversary is then given access to two oracles, $\mathcal{E} = \mathcal{E}_K^b$ and $\mathcal{D} = \mathcal{D}_K^b$. The attacker can query the $\mathcal{E}$-oracle with any 4-tuple $(T_0, P_0,\ T_1, P_1)$ where $T_0, T_1 \in \mathcal{T}$ and P_0 and P_1 are equal-length strings in $\mathcal{M}$. The oracle returns $\mathcal{E}_K(T_b, P_b)$. Alternatively, the adversary can query the $\mathcal{D}$ oracle with a 4-tuple $(T_0, C_0,\ T_1, C_1)$ where $T_0, T_1 \in \mathcal{T}$ and C_0 and C_1 are equal-length strings in $\mathcal{M}$. The oracle returns $\mathcal{D}_K(T_b, C_b)$ where $\mathcal{D}$ is the inverse of $\mathcal{E}$. The adversary wants to identify the bit b. We must disallow the adversary from asking queries that will allow it to win trivially. The disallowed queries are given in Table 1.

The advantage of the adversary in guessing the bit b is defined by

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{ind}}}(A) \stackrel{\mathrm{def}}{=} \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K}:\ A^{\mathcal{E}_K^1\, \mathcal{D}_K^1} \Rightarrow 1\,] \;-\; \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K}:\ A^{\mathcal{E}_K^0\, \mathcal{D}_K^0} \Rightarrow 1\,]$$

We now show a tight equivalence between the PRP-security of a tweakable enciphering scheme and its indistinguishability. In Theorem 3 we show that PRP-security implies indistinguishability, and in Theorem 4 we show the converse. The proofs are in Appendices F and G, respectively.

Theorem 3 [$\pm\widetilde{\mathbf{prp}}$**-security** $\Rightarrow$ $\pm\widetilde{\mathbf{ind}}$**-security**] Let $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be an enciphering scheme whose message space $\mathcal{M}$ consists of strings of length at least n bits. Then for any t, q, μ,

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{ind}}}(t, q, 2\mu) \le 2\,\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(t', q, \mu) + \frac{2q^2}{2^n - q}$$

where $t' = t + O(\mu)$. □

Theorem 4 [$\pm\widetilde{\mathbf{ind}}$**-security** $\Rightarrow$ $\pm\widetilde{\mathbf{prp}}$**-security**] Let $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be an enciphering scheme. Then for any t, q, μ, we have $\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(t, q, \mu) \le \mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{ind}}}(t', q, 2\mu)$, where $t' = t + \widetilde{O}(\mu)$. □

Nonmalleability. Nonmalleability is an important cryptographic goal that was first identified and investigated by Dolev, Dwork, and Naor [12]. Informally, an encryption scheme is nonmalleable if an adversary cannot modify a ciphertext C to create a ciphertext C^* where the plaintext P^* of C^* is related to the plaintext P of C. In this section we define the nonmalleability of a tweakable enciphering scheme with respect to a chosen-ciphertext attack and we show that $\pm\widetilde{\mathrm{prp}}$-security implies nonmalleability. The result mirrors the well-known result that indistinguishability of a probabilistic encryption scheme under a chosen-ciphertext attack implies its nonmalleability under the same kind of attack [4, 12].

Fix an enciphering scheme $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ and an adversary A. Consider running A with two oracles: an enciphering oracle $\mathcal{E}_K(\cdot,\cdot)$ and a deciphering oracle $\mathcal{D}_K(\cdot,\cdot)$, where $\mathcal{D} = \mathcal{E}^{-1}$ and K is chosen randomly from $\mathcal{K}$. After A has made all of its oracle queries and halted, we define a number of sets:

- *Known plaintexts.* For every $T \in \mathcal{T}$ we define the $\mathcal{P}^T$ as the set of all P such that A asked $\mathcal{E}_K$ to encipher (T, P) or A asked $\mathcal{D}_K$ to decipher some (T, C) and A got back an answer of P. Thus $\mathcal{P}^T$ is the set of all plaintexts P associated to T that the adversary already "knows".
- *Known ciphertexts.* For every $T \in \mathcal{T}$ we define $\mathcal{C}^T$ as the set of all C such A asked $\mathcal{D}_K$ to decipher (T, C) or A asked $\mathcal{E}_K$ to encipher some (T, P) and A got back an answer of C. Thus $\mathcal{C}^T$ is the set of all ciphertexts C associated to T that the adversary already "knows".
- *Plausible plaintexts.* For every $T \in \mathcal{T}$ and $C \in \mathcal{M}$ we define $\mathcal{P}^T(C)$ as the singleton set $\{\mathcal{D}_K^T(C)\}$ if $C \in \mathcal{C}^T$ and as $\{0,1\}^{|C|} \setminus \mathcal{P}^T$ otherwise. Thus $\mathcal{P}^T(C)$ is the set of all plaintexts P for which the adversary should regard it as plausible that $C = \mathcal{E}_K^T(P)$.

With enciphering scheme $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ and adversary A still fixed, we consider the following two games, which we call games Real and Ideal. Both games begin by choosing a random key $K \stackrel{\$}{\leftarrow} \mathcal{K}$ and letting the adversary A interact with oracles $\mathcal{E}_K$ and $\mathcal{D}_K$ where $\mathcal{D} = \mathcal{E}^{-1}$. Just before termination, after the adversary has asked all the queries that it will ask, it outputs a three-tuple (T, C, f) where $T \in \mathcal{T}$ and $C \in \mathcal{M}$ and f is the encoding of a predicate $f\colon \mathcal{M} \to \{0,1\}$ (we do not distinguish between the predicate and its encoding). Now for game Real we set $P \leftarrow \mathcal{D}_K^T(C)$ and for game Ideal we set $P \stackrel{\$}{\leftarrow} \mathcal{P}^T(C)$. Finally, we look at the event that $f(P) = 1$. Formally, we define the advantage of A, in the sense of nonmalleability under a chosen-ciphertext attack, as follows:

$$
\begin{aligned}
\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{nm}}}(A) \;=\; & \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K};\, (T, C, f) \stackrel{\$}{\leftarrow} A^{\mathcal{E}_K(\cdot,\cdot)\,\mathcal{D}_K(\cdot,\cdot)};\, P \leftarrow \mathcal{D}_K^T(C):\; f(P) = 1] - \\
& \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K};\, (T, C, f) \stackrel{\$}{\leftarrow} A^{\mathcal{E}_K(\cdot,\cdot)\,\mathcal{D}_K(\cdot,\cdot)};\, P \stackrel{\$}{\leftarrow} \mathcal{P}^T(C):\; f(P) = 1]
\end{aligned}
$$

We emphasize that in game Ideal (the second experiment) the set $\mathcal{P}^T(C)$ depends on the oracle queries asked by A and the answers returned to it (even though this is not reflected in the notation). For the resource-bounded version of $\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{nm}}}$ we let the running time t include the running time to compute $f(P)$. We have the following result, the proof of which appears in Appendix H.

Theorem 5 [$\pm\widetilde{\mathbf{prp}}$**-security** $\Rightarrow$ $\pm\widetilde{\mathbf{nm}}$**-security**] Let $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ be an enciphering scheme. Then for any t, q, μ, ς

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{nm}}}(t, q, \mu, \varsigma) \le 2\, \mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(t', q+1, \mu+\varsigma)$$

where $t' = t + \widetilde{O}(\mu + \varsigma)$. $\square$

Acknowledgments

The authors thank Jim Hughes for posing this problem to each of us and motivating our work on it. Shai thanks Hugo Krawczyk and Charanjit Jutla for discussions regarding candidates for implementing the Naor-Reingold approach and regarding the security proof for CMC. Phil thanks John Black for useful conversations on this problem, and Mihir Bellare, who promptly broke his first two-layer attempts. Phil received support from NSF grant CCR-0085961 and a gift from CISCO Systems. This work was carried out while Phil was at Chiang Mai University, Thailand.

References

[1] R. Anderson and E. Biham. Two practical and provably secure block ciphers: BEAR and LION. In *Fast Software Encryption, Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 113–120, 1996. www.cs.technion.ac.il/∼biham/.

[2] K. Aoki and H. Lipmaa. Fast implementations of AES candidates. In *Third AES Candidate Conference*, 2000. See www.tcs.hut.fi/∼helger for the latest data.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1998.

[5] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000. www.cs.ucdavis.edu/∼rogaway.

[6] M. Bellare and P. Rogaway. On the construction of variable-input-length ciphers. In *Fast Software Encryption—6th International Workshop—FSE '99*, volume 1635 of *Lecture Notes in Computer Science*, pages 231–244. Springer-Verlag, 1999. www.cs.ucdavis.edu/∼rogaway.

[7] D. Bernstein. Floating point arithmetic and message authentication. Available at http://cr.yp.to/hash127.html, 2000.

[8] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In M. Wiener, editor, *Advances in Cryptology– CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.

[9] D. Bleichenbacher and A. Desai. A construction of a super-pseudorandom cipher. Manuscript, February 1999.

[10] L. Carter and M. Wegman. Universal hash functions. *J. of Computer and System Sciences*, 18(2):143–154, 1979.

[11] P. Crowley. Mercy: A fast large block cipher for disk sector encryption. In B. Schneier, editor, *Fast Software Encryption: 7th International Workshop*, volume 1978 of *Lecture Notes in Computer Science*, pages 49–63, New York, USA, Apr. 2000. Springer-Verlag. www.ciphergoth.org/crypto/mercy.

[12] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. Earlier version in STOC 91.

[13] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, Apr. 1984.

[14] S. Halevi and H. Krawczyk. MMH: Software message authentication in the Gbit/second rates. In E. Biham, editor, *Fast Software Encryption (FSE '97)*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 1997.

[15] S. Halevi and P. Rogaway. A tweakable enciphering mode. In D. Boneh, editor, *Advances in Cryptology – CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003. Proceedings version of this paper.

[16] J. Hughes. Chair of the IEEE Security in Storage Working Group. Working group homepage at www.siswg.org. Call for algorithms can be found at www.mail-archive.com/cryptography@wasabisystems.com/msg02102.html, May 2002.

[17] A. Joux. Cryptanalysis of the EMD mode of operation. In *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[18] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search. *Journal of Cryptology*, 14(1):17–35, 2001. Earlier version in CRYPTO '96. www.cs.ucdavis.edu/∼rogaway.

[19] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer-Verlag, 1994.

[20] M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science. Springer-Verlag, 2002. www.cs.berkeley.edu/∼daw/.

[21] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. of Computation*, 17(2), April 1988.

[22] C. Meyer and S. Matyas. *Cryptography: A new dimension in computer security*. John Wiley and Sons, 1982.

[23] M. Naor and O. Reingold. A pseudo-random encryption mode. Manuscript, available from www.wisdom.weizmann.ac.il/∼naor/.

[24] M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1):29–66, 1999. (Earlier version in STOC '97.) Available from www.wisdom.weizmann.ac.il/∼naor/.

[25] P. Rogaway. The EMD mode of operation (a tweaked, wide-blocksize, strong PRP). Cryptology ePrint Archive, Report 2002/148, Oct. 2002. Early (buggy) version of the CMC algorithm. http://eprint.iacr.org/.

[26] R. Schroeppel. The hasty pudding cipher. AES candidate submitted to NIST. www.cs.arizona.edu/∼rcs/hpc, 1999.

[27] Y. Zheng, T. Matsumoto, and H. Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 461–480. Springer-Verlag, 1989.

A The Joux Attack

In an early, unpublished version of the current paper [25] the scheme CMC, then called EMD, worked a little bit differently: instead of computing $\mathbb{T} = \mathbb{E}_K(T)$ and xoring $\mathbb{T}$ into P_1 and C_1, we simply xored T into the mask M, setting $M = 2(PPP_1 \oplus PPP_m) \oplus T$. We claimed—incorrectly—that the scheme was secure (as a tweakable, strong PRP). Antoine Joux [17] noticed that the scheme was wrong, pointing out that it is easy to distinguish the mode and its inverse from a tweakable truly random permutation and its inverse. Below is (a slightly simplified variant of) his attack:

1. The adversary picks an arbitrary tweak T and an arbitrary 4-block plaintext $P_1P_2P_3P_4$. It encrypts $(T,\ P_1P_2P_3P_4)$, obtaining ciphertext $C_1C_2C_3C_4$, and it encrypts $(T+1,\ P_1P_2P_3P_4)$, obtaining a different ciphertext $C_1'C_2'C_3'C_4'$.

2. The adversary now decrypts $(T, C_1(C_2'+1)(C_3+1)C_4)$, obtaining plaintext $P_1''P_2''P_3''P_4''$.

If $P_1'' = P_1$ then the adversary outputs 1 (it guesses that it has a "real" enciphering oracle; otherwise, the adversary answers 0 (it knows that it has a "fake" enciphering oracle). It is easy to see that this attack has advantage of nearly 1.

What went wrong? Clearly the provided proof had a bug. The bug turns out not to be a particularly interesting one. On the 14-th page of the proof [25] begins a detailed case analysis. The case denoted X1–X5 was incorrect: two random variables are said to rarely collide, but with an appropriate choice of constants the random variables become degenerate (constants) and *always* collide. The same happens for case Y1–Y5. The current paper restructures the case analysis.

Our earlier manuscript [25] also mentioned a parallelizable mode that we called EME. Joux also provides an attack on EME, using the tweak in a manner similar to the attack on CMC. We later found that, as opposed to CMC, the EME scheme remains insecure even as an untweakable PRP. Thus one cannot repair EME simply by using a different method of incorporating the tweak.

B A "natively tweakable" variant of CMC

Two minor drawbacks of the tweakable CMC mode is that it uses two keys of the underlying cipher (rather than just one), and that both directions of the block cipher are used to decipher. Both of these drawbacks are due to the "generic" way in which the tweak is incorporated into CMC. We now describe a variant, denoted CMC$'$, that avoids these drawbacks (at the cost of one more block encryption), by "natively" incorporating the tweak in the computation. We specify in Figure 3 both the forward direction of this variant, $\mathcal{E} = \mathrm{CMC}'\text{-}E$, and its inverse $\mathcal{D}$. An illustration is given in Figure 4.

C A Useful Lemma — $\pm\widetilde{\mathbf{prp}}$-security $\Leftrightarrow$ $\pm\widetilde{\mathbf{rnd}}$-security

Before proving security for CMC, we provide a little lemma that says that a (tweakable) truly random permutation and its inverse looks very much like a pair of oracles that just return random bits (assuming you never ask pointless queries). So instead of analyzing indistinguishability from a random permutation we might as well analyze indistinguishability from random bits.

Let $\mathcal{E}\colon \mathcal{K}\times\mathcal{T}\times\mathcal{M}\to\mathcal{M}$ be a tweaked block-cipher and let $\mathcal{D}$ be its inverse. The advantage of distinguishing $\mathcal{E}$ from random bits, $\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{rnd}}}$, is

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{rnd}}}(A) \;=\; \Pr[K \xleftarrow{\$} \mathcal{K}:\ A^{\mathcal{E}_K(\cdot,\cdot)\ \mathcal{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\,A^{\$(\cdot,\cdot)\ \$(\cdot,\cdot)} \Rightarrow 1\,]$$

where $\$(T,M)$ returns a random string of length $|M|$. We insist that A makes no pointless queries, regardless of oracle responses, and A asks no query (T,M) outside of $\mathcal{T}\times\mathcal{M}$. We extend the definition above in the usual way to its resource-bounded versions. We have the following:

Lemma 6 [$\pm\widetilde{\mathbf{prp}}$**-security** $\approx$ $\pm\widetilde{\mathbf{rnd}}$**-security**] Let $\mathcal{E}\colon\mathcal{K}\times\mathcal{T}\times\mathcal{M}\to\mathcal{M}$ be a tweaked block-cipher and let $q\ge 1$. Then

$$|\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{prp}}}(q) - \mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\mathrm{rnd}}}(q)| \;\le\; q(q-1)/2^{N+1}$$

where N is the length of a shortest string in $\mathcal{M}$.

Algorithm $\mathcal{E}_K^T(P_1 \cdots P_m)$

$CCC_0 \leftarrow PPP_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** m **do**
  $PP_i \leftarrow P_i \oplus PPP_{i-1}$
  $PPP_i \leftarrow E_K(PP_i)$

$MC \leftarrow E_K(PPP_m \oplus T)$
$M \leftarrow 2\,(PPP_1 \oplus MC)$
$MP \leftarrow PPP_1 \oplus M$

$CCC_1 \leftarrow MC \oplus M$
for $i \in [2\,..\,m-1]$ **do**
  $CCC_i \leftarrow PPP_{m+1-i} \oplus M$
$CCC_m \leftarrow E_K(MP) \oplus T$

for $i \in [1\,..\,m]$ **do**
  $CC_i \leftarrow E_K(CCC_i)$
  $C_i \leftarrow CC_i \oplus CCC_{i-1}$

return $C_1 \cdots C_m$

Algorithm $\mathcal{D}_K^T(C_1 \cdots C_m)$

$PPP_0 \leftarrow CCC_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** m **do**
  $CC_i \leftarrow C_i \oplus CCC_{i-1}$
  $CCC_i \leftarrow E_K^{-1}(CC_i)$

$MP \leftarrow E_K^{-1}(CCC_m \oplus T)$
$M \leftarrow 2\,(CCC_1 \oplus MP)$
$MC \leftarrow CCC_1 \oplus M$

$PPP_1 \leftarrow MP \oplus M$
for $i \in [2\,..\,m-1]$ **do**
  $PPP_i \leftarrow CCC_{m+1-i} \oplus M$
$PPP_m \leftarrow E_K^{-1}(MC) \oplus T$

for $i \in [1\,..\,m]$ **do**
  $PP_i \leftarrow E_K^{-1}(PPP_i)$
  $P_i \leftarrow PP_i \oplus PPP_{i-1}$

return $P_1 \cdots P_m$

Figure 3: Enciphering (left) and deciphering (right) under $\mathcal{E} = \mathrm{CMC}'[E]$, where $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a block cipher. The tweak is $T \in \{0,1\}^n$ and the plaintext is $P = P_1 \cdots P_m$ and the ciphertext is $C = C_1 \cdots C_m$.

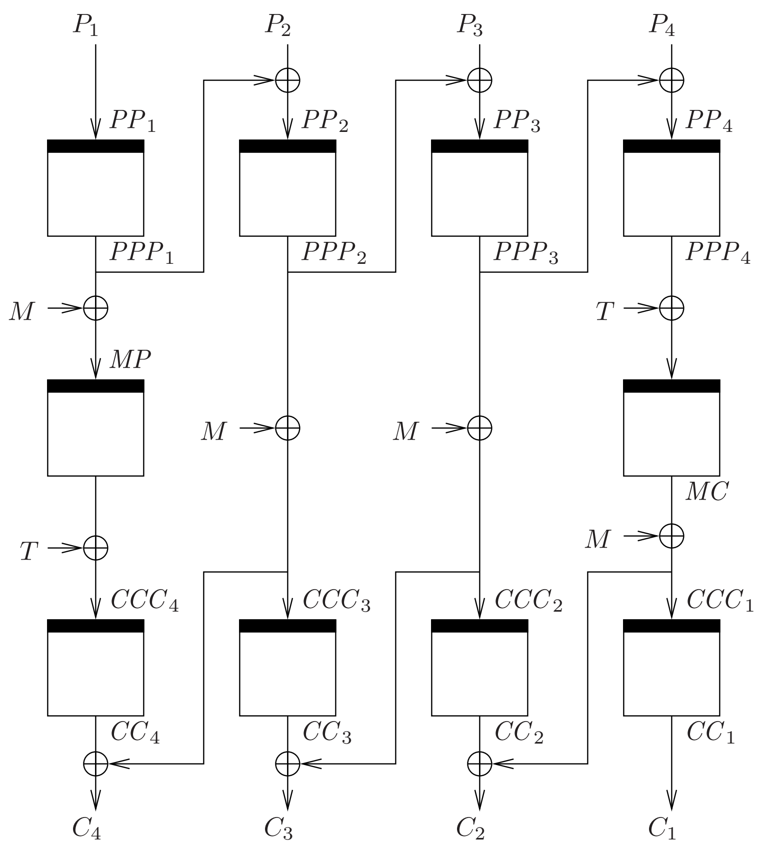

Figure 4: Enciphering under CMC' mode for a message of $m = 4$ blocks. The boxes represent E_K. We set mask $M = 2\,(PPP_1 \oplus MC)$. This value can also be computed as $M = 2\,(CCC_1 \oplus MP)$.

$\square$

The proof follows the well-known argument relating PRP-security to PRF-security (e.g., [5]). Namely, let A be an adversary that interacts with an oracle $F\,F'$. Assume that A makes no purposeless queries and at most q queries overall. Let $\mathcal{X}$ be the multiset of strings which are either asked to F or answered by F', and let $\mathcal{Y}$ be the multiset of strings which are either asked to F' or answered by F. When $F\,F' = \$\,\$$ let C be the event that some string in $\mathcal{X}$ appears twice or some string in $\mathcal{Y}$ appears twice, and let C_i be the event that the adversary's i^{th} query, when added to $\mathcal{X}$ or $\mathcal{Y}$, was already in that set. Then

$$\begin{aligned}
|\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(A) - \mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathcal{E}}(A)| &= |\Pr[A^{\mathcal{E}_K\,\mathcal{D}_K} \Rightarrow 1] - \Pr[A^{\pi\,\pi^{-1}} \Rightarrow 1] - \Pr[A^{\mathcal{E}_K\,\mathcal{D}_K} \Rightarrow 1] + \Pr[A^{\$\,\$} \Rightarrow 1]| \\
&= |\Pr[A^{\$\,\$} \Rightarrow 1] - \Pr[A^{\pi\,\pi^{-1}} \Rightarrow 1]| \\
&= |\Pr[A^{\$\,\$} \Rightarrow 1\,|\mathsf{C}]\Pr[\mathsf{C}] + \Pr[A^{\$\,\$} \Rightarrow 1\,|\overline{\mathsf{C}}](1-\Pr[\mathsf{C}]) - \Pr[A^{\pi\,\pi^{-1}} \Rightarrow 1]| \\
&\le |zy + x(1-y) - x| \\
&= |y(z-x)| \\
&\le y
\end{aligned}$$

where $x = \Pr[A^{\pi\,\pi^{-1}} \Rightarrow 1]|$ and $y = \Pr[\mathsf{C}]$ and $z = \Pr[A^{\$\,\$} \Rightarrow 1\,|\mathsf{C}]$. Now $y = \Pr[\mathsf{C}] \le \sum_{i=1}^{q} \Pr[\mathsf{C}_i] \le (1 + \cdots + (q-1))/2^n \le q(q-1)/2^{N+1}$ as the i-th query will cause C_i with probability at most $(i-1)/2^N$.

D Proof of Theorem 1 — Security of $\mathbb{CMC}$

Our proof of security for $\mathbb{CMC}$ is divided into two parts: (1) a game-substitution argument, reducing the analysis of $\mathbb{CMC}$ to the analysis of a simpler probabilistic game; and (2) analyzing that game. We also use Lemma 6 from above.

D.1 The Game-Substitution Sequence

Let n, m, and q all be fixed, and $\sigma = mq$. Let A be an adversary that asks q oracle queries (none pointless), each of nm bits. Our first major goal is to describe a probability space, NON2, this probability space depending on constants derived from A, and to define an event on the probability space, denoted NON2 sets *bad*, for which $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(A) \le 2\cdot\Pr[\text{NON2 sets } bad] + \sigma^2/2^n$. Later we bound $\Pr[\text{NON2 sets } bad]$ and, putting that together with Lemma 6, we will get Equation (1) of Theorem 1. The rest of Theorem 1 follows easily, as explained in Section 5. Game NON2 is obtained by a game-substitution argument, as carried out in works like [18]. The goal is to simplify the rather complicated setting of A adaptively querying its oracles and to arrive at a simpler setting where there is no adversary and no interaction—just a program that flips coins and a flag *bad* that does or does not get set.

Game CMC1. We describe the attack of A against $\mathbb{CMC}[\mathrm{Perm}(n)]$ as a probabilistic game in which the permutation π is chosen "on the fly", as needed to answer the queries of A. Initially, the partial function $\pi\colon \{0,1\}^n \to \{0,1\}^n$ is everywhere undefined. When we need $\pi(X)$ and π isn't yet defined at X we choose this value randomly among the available range values. When we need $\pi^{-1}(Y)$ and there is no X for which $\pi(X)$ has been set to Y we likewise choose X at random from the available domain values. As we fill in π its domain and its range thus grow. In the game we keep track of the domain and range of π by maintaining two sets, Domain and Range, that include all the points for which π is already defined. We let $\overline{\text{Domain}}$ and $\overline{\text{Range}}$ be the complement of these sets relative to $\{0,1\}^n$. The game, denoted CMC1, is shown in Figure 5. Since game CMC1 accurately

Initialization:

000 $\quad bad \leftarrow \mathsf{false}$; $\quad \text{Domain} \leftarrow \text{Range} \leftarrow \emptyset$; $\quad$ **for** all $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$

Respond to the s-th adversary query as follows:

An encipher query, $\mathsf{Enc}(P_1^s \cdots P_m^s)$:

110 $\quad$ Let $u[s]$ be the largest value in $[0\,..\,m]$ s.t. $P_1^s \cdots P_{u[s]}^s = P_1^r \cdots P_{u[s]}^r$ for some $r < s$

111 $\quad PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; $\quad$ **for** $i \leftarrow 1$ **to** $u[s]$ **do** $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$, $\quad PPP_i^s \leftarrow PPP_i^r$

112 $\quad$ **for** $i \leftarrow u[s]+1$ **to** m **do**

113 $\qquad PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$

114 $\qquad PPP_i^s \xleftarrow{\$} \{0,1\}^n$; $\quad$ **if** $PPP_i^s \in \text{Range}$ **then** $bad \leftarrow \mathsf{true}$ $,\ PPP_i^s \xleftarrow{\$} \overline{\text{Range}}$

115 $\qquad$ **if** $PP_i^s \in \text{Domain}$ **then** $bad \leftarrow \mathsf{true}$ $,\ PPP_i^s \leftarrow \pi(PP_i^s)$

116 $\qquad \pi(PP_i^s) \leftarrow PPP_i^s$, $\text{Domain} \leftarrow \text{Domain} \cup \{PP_i^s\}$, $\text{Range} \leftarrow \text{Range} \cup \{PPP_i^s\}$

120 $\quad M^s \leftarrow 2\,(PPP_1^s \oplus PPP_m^s)$; $\quad$ **for** $i \in [1\,..\,m]$ **do** $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus M^s$

130 $\quad$ **for** $i \leftarrow 1$ **to** m **do**

131 $\qquad CC_i^s \xleftarrow{\$} \{0,1\}^n$; $\quad$ **if** $CC_i^s \in \text{Range}$ **then** $bad \leftarrow \mathsf{true}$ $,\ CC_i^s \xleftarrow{\$} \overline{\text{Range}}$

132 $\qquad$ **if** $CCC_i^s \in \text{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$ $,\ CC_i^s \leftarrow \pi(CCC_i^s)$

133 $\qquad C_i^s \leftarrow CC_i^s \oplus CCC_{i-1}^s$

134 $\qquad \pi(CCC_i^s) \leftarrow CC_i^s$, $\text{Domain} \leftarrow \text{Domain} \cup \{CCC_i^s\}$, $\text{Range} \leftarrow \text{Range} \cup \{CC_i^s\}$

140 $\quad$ **return** $C_1 \cdots C_m$

A decipher query, $\mathsf{Dec}(C_1^s \cdots C_m^s)$:

210 $\quad$ Let $u[s]$ be the largest value in $[0\,..\,m]$ s.t. $C_1^s \cdots C_{u[s]}^s = C_1^r \cdots C_{u[s]}^r$ for some $r < s$

211 $\quad CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; $\quad$ **for** $i \leftarrow 1$ **to** $u[s]$ **do** $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$, $\quad CCC_i^s \leftarrow CCC_i^r$

212 $\quad$ **for** $i \leftarrow u[s]+1$ **to** m **do**

213 $\qquad CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$

214 $\qquad CCC_i^s \xleftarrow{\$} \{0,1\}^n$; $\quad$ **if** $CCC_i^s \in \text{Domain}$ **then** $bad \leftarrow \mathsf{true}$ $,\ CCC_i^s \xleftarrow{\$} \overline{\text{Domain}}$

215 $\qquad$ **if** $CC_i^s \in \text{Range}$ **then** $bad \leftarrow \mathsf{true}$ $,\ CCC_i^s \leftarrow \pi^{-1}(CC_i^s)$

216 $\qquad \pi(CCC_i^s) \leftarrow CC_i^s$, $\text{Domain} \leftarrow \text{Domain} \cup \{CCC_i^s\}$, $\text{Range} \leftarrow \text{Range} \cup \{CC_i^s\}$

220 $\quad M^s \leftarrow 2\,(CCC_1^s \oplus CCC_m^s)$; $\quad$ **for** $i \in [1\,..\,m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus M^s$

230 $\quad$ **for** $i \leftarrow 1$ **to** m **do**

231 $\qquad PP_i^s \xleftarrow{\$} \{0,1\}^n$; $\quad$ **if** $PP_i^s \in \text{Domain}$ **then** $bad \leftarrow \mathsf{true}$ $,\ PP_i^s \xleftarrow{\$} \overline{\text{Domain}}$

232 $\qquad$ **if** $PPP_i^s \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$ $,\ PP_i^s \leftarrow \pi^{-1}(PPP_i^s)$

233 $\qquad P_i^s \leftarrow PP_i^s \oplus PPP_{i-1}^s$

234 $\qquad \pi(PP_i^s) \leftarrow PPP_i^s$, $\text{Domain} \leftarrow \text{Domain} \cup \{PP_i^s\}$, $\text{Range} \leftarrow \text{Range} \cup \{PPP_i^s\}$

240 $\quad$ **return** $P_1 \cdots P_m$

Figure 5: Game CMC1 describes the attack of A on $\mathbb{CMC}[\text{Perm}(n)]$, where the permutation π is chosen "on the fly" as needed. Game RND1 is the same as game CMC1 except we do not execute the shaded statements.

represent the attack scenario, we have that

$$\Pr[\,A^{\mathbb{E}_\pi\,\mathbb{D}_\pi} \Rightarrow 1\,] \quad = \quad \Pr[\,A^{\mathrm{CMC1}} \Rightarrow 1\,] \tag{5}$$

Looking ahead to the game substitution sequence, we have structured the code in Figure 5 in a way that makes it easier to present the following games. In particular, here are some things to note about this code:

- *The notation* $u[s]$*.* When handling the s-th adversary query we need to focus on the first "new block" in that query. Suppose the s^{th} query is an encipher query, $P_1^s \dots P_m^s$. Then we look for the first index i such that $P_1^s \dots P_i^s$ is not a prefix of any other plaintext that was encountered in the game so far. Since we do CBC encryption at the first layer, the index i represents the first time we expect to choose a new value for π (all previous values must already be defined). For convenience, we define $u[s]$ to be the index immediately prior to this "first new block". Namely, we define $u[s]$ to give the length, in blocks, of the longest prefix of plaintext blocks that has *already* been seen:

$$u[s] \stackrel{\text{def}}{=} \max\{\; i \;:\; \exists\, r < s,\ \text{s.t. } P_1^s \dots P_i^s = P_1^r \dots P_i^r \;\}$$

  The value of $u[s]$ is understood to be 0 if $s = 1$ (no prior queries) or $s > 1$ but all prior plaintext blocks P_1^r differ from P_1^s. Note that the plaintexts being considered here is both the plaintexts asked in encipher queries and plaintexts returned by decipher queries. We use the symmetric definition for decipher queries.
- *Filling in* π *and* π^{-1} *values.* When we need to define π on what is likely to be a new domain point X, setting $\pi(X) \leftarrow Y$ for some Y, we do not do the natural thing of first checking if π has a value at X and, if not, choosing a random value from $\overline{\text{Range}}$ as Y. Instead, we first sample Y from $\{0,1\}^n$; then *re-sample*, this time from $\overline{\text{Range}}$, if the initially chosen sample Y was already in the range of π; finally, if π already had a value at X, then we forget about Y and use the original value. We behave analogously for $\pi^{-1}(Y)$ values. In Figure 5 we highlight the places where we have to reset a choice we tentatively made. Whenever we do so we set a flag *bad*. The flag *bad* is never seen by the adversary A that interacts with the CMC1 game—it is only present to facilitate the subsequent analysis.

Game RND1. We next modify game CMC1 by omitting the statements that immediately follow the setting of *bad* to true. (This is the usual trick under the game-substitution approach.) Namely, where before we were making some consistency checks after each random choice $\pi(X) = Y \xleftarrow{\$} \{0,1\}^n$ (to see if this value of Y was already in use, or if π was already defined at X), we now omit all these checks. This means that π may end up not being a permutation, and moreover we may reset its value on previously chosen points.

Still, the games CMC1 and RND1 are syntactically identical apart from what happens after the setting of the flag *bad* to true. Once the flag *bad* is set to true the subsequent behavior of the game does not impact the probability that an adversary A interacting with the game can set the flag *bad* to true. This is exactly the setup used in the game-substitution method to conclude that

$$\Pr[\,A^{\mathrm{CMC1}} \Rightarrow 1\,] - \Pr[\,A^{\mathrm{RND1}} \Rightarrow 1\,] \quad \le \quad \Pr[\,A^{\mathrm{RND1}} \text{ sets } \mathit{bad}\,] \tag{6}$$

Game RND2. We now make two adversarially-invisible changes to game RND1. First, we note that the function π (and its inverse) are never used in game RND1, so we just remove them from the code. Next, instead of choosing $CC_i^s \xleftarrow{\$} \{0,1\}^n$ and then setting $C_i^s \leftarrow CC_i^s \oplus CCC_{i-1}^s$ (in

Initialization:

000 $\quad bad \leftarrow \mathsf{false}; \quad \text{Domain} \leftarrow \text{Range} \leftarrow \emptyset;$

Respond to the s-th adversary query as follows:

AN ENCIPHER QUERY, $\mathsf{Enc}(P_1^s \cdots P_m^s)$:

110 $\quad$ Let $u[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $P_1^s \cdots P_{u[s]}^s = P_1^r \cdots P_{u[s]}^r$ for some $r < s$

111 $\quad PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n;\quad \textbf{for } i \leftarrow 1 \textbf{ to } u[s] \textbf{ do } PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s, \quad PPP_i^s \leftarrow PPP_i^r$

112 $\quad \textbf{for } i \leftarrow u[s]+1 \textbf{ to } m \textbf{ do}$

113 $\qquad PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$

114 $\qquad PPP_i^s \xleftarrow{\$} \{0,1\}^n;\quad \textbf{if } PPP_i^s \in \text{Range } \textbf{then } bad \leftarrow \mathsf{true}$

115 $\qquad \textbf{if } PP_i^s \in \text{Domain } \textbf{then } bad \leftarrow \mathsf{true}$

116 $\qquad \text{Domain} \leftarrow \text{Domain} \cup \{PP_i^s\},\ \text{Range} \leftarrow \text{Range} \cup \{PPP_i^s\}$

120 $\quad M^s \leftarrow 2\,(PPP_1^s \oplus PPP_m^s);\quad \textbf{for } i \in [1 \mathinner{..} m] \textbf{ do } CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus M^s$

130 $\quad \textbf{for } i \leftarrow 1 \textbf{ to } m \textbf{ do}$

131 $\qquad C_i^s \xleftarrow{\$} \{0,1\}^n$

132 $\qquad CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s;\quad \textbf{if } CC_i^s \in \text{Range } \textbf{then } bad \leftarrow \mathsf{true}$

133 $\qquad \textbf{if } CCC_i^s \in \text{Domain } \textbf{then } bad \leftarrow \mathsf{true}$

134 $\qquad \text{Domain} \leftarrow \text{Domain} \cup \{CCC_i^s\},\ \text{Range} \leftarrow \text{Range} \cup \{CC_i^s\}$

140 $\quad \textbf{return } C_1 \cdots C_m$

A DECIPHER QUERY, $\mathsf{Dec}(C_1^s \cdots C_m^s)$, IS TREATED SYMMETRICALLY

Figure 6: Game RND2 is indistinguishable from Game RND1 but chooses some of its variables in different order.

lines 131–133), we will now choose $C_i^s \xleftarrow{\$} \{0,1\}^n$ and then set $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$. Clearly, this change preserves the distribution of all these variables. The analogous comments apply to the choice of PP_i^s and P_i^s in lines 231–233 of game RND1; we could just as well have chosen P_i^s at random and defined PP_i^s using it. Game RND2 is described in Figure 6. (In that figure we did not bother describing the decipher queries, as they are completely symmetric to the encipher queries.) Since games RND1 and RND2 define the same distribution over all their variables, it follows in particular that

$$\Pr[\,A^{\mathrm{RND1}} \text{ sets } bad\,] = \Pr[\,A^{\mathrm{RND2}} \text{ sets } bad\,] \quad \text{and} \quad \Pr[\,A^{\mathrm{RND1}} \Rightarrow 1\,] = \Pr[\,A^{\mathrm{RND2}} \Rightarrow 1\,] \tag{7}$$

Next we note that in game RND2 we return nm random bits in response to any m-block Enc-query. Similarly, we return nm random bits in response to any m-block Dec-query. Thus RND2 provides an adversary with an identical view to a pair of random-bit oracles,

$$\Pr[\,A^{\mathrm{RND2}} \Rightarrow 1\,] \quad = \quad \Pr[\,A^{\pm\widetilde{\mathrm{rnd}}} \Rightarrow 1\,] \tag{8}$$

Combining Equations 5, 6, 7, and 8, we thus have that

$$\begin{aligned}
\mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(A) &= \Pr[\,A^{\mathrm{CMC1}} \Rightarrow 1\,] - \Pr[\,A^{\mathrm{RND2}} \Rightarrow 1\,] \\
&= \Pr[\,A^{\mathrm{CMC1}} \Rightarrow 1\,] - \Pr[\,A^{\mathrm{RND1}} \Rightarrow 1\,] \\
&\le \Pr[\,A^{\mathrm{RND1}} \text{ sets } bad\,] \\
&= \Pr[\,A^{\mathrm{RND2}} \text{ sets } bad\,] \qquad (9)
\end{aligned}$$

Our task is thus to bound $\Pr[\,A^{\mathrm{RND2}} \text{ sets } bad\,]$.

Respond to the s-th adversary query as follows:

An encipher query, $\mathsf{Enc}(P_1^s \cdots P_m^s)$:

$ty^s \leftarrow \mathsf{Enc}$

$C^s = C_1^s \cdots C_m^s \xleftarrow{\$} \{0,1\}^{nm}$

return C^s

A decipher query, $\mathsf{Dec}(C_1^s \cdots C_m^s)$:

$ty^s \leftarrow \mathsf{Dec}$

$P^s = P_1^s \cdots P_m^s \xleftarrow{\$} \{0,1\}^{nm}$

return P^s

Finalization:

First phase

$\mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \emptyset$ // These are *multisets*—a point can be in $\mathfrak{D}$ or $\mathfrak{R}$ more than once

for $s \leftarrow 1$ **to** q **do**

  if $ty^s = \mathsf{Enc}$ **then**

    Let $u[s]$ be the largest value in $[0 \,..\, m]$ s.t. $P_1^s \cdots P_{u[s]}^s = P_1^r \cdots P_{u[s]}^r$ for some $r < s$

    $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $u[s]$ **do** $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow PPP_i^r$

    for $i \leftarrow u[s]+1$ **to** m **do**

      $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$

      $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$

    $M^s \leftarrow 2\,(PPP_1^s \oplus PPP_m^s)$

    for $i \leftarrow 1$ **to** m **do**

      $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus M^s$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$

      $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC_i^s\}$

  if $ty^s = \mathsf{Dec}$ **then** *behave analogously*

Second phase

$bad \leftarrow$ (some value appears more than once in $\mathfrak{D}$)
  or (some value appears more than once in $\mathfrak{R}$)

Figure 7: Game RND3 is adversarially indistinguishable from game RND2 but defers the setting of bad.

Game RND3. Next we reorganize game RND2 so as to separate out (i) choosing random values to return to the adversary, (ii) defining intermediate variables, and (iii) setting the flag bad.

We already remarked that game RND2 returns mn random bits in response to any m-block query. Now, in game RND3, shown in Figure 7, we make that even more clear by choosing the necessary $C^s = C_1^s \cdots C_m^s$ or $P^s = P_1^s \cdots P_m^s$ response just as soon as the s^{th} query is made. Nothing else is done at that point except for recording if the adversary made an Enc query or a Dec query.

When the adversary finishes all of its oracle queries and halts, we execute the "finalization" step of game RND3. First, we go over all the variables of the game and determine their values, just as we do in game RND2. While doing so, we collect all the values in the sets Domain and Range, this time viewing them as *multisets* $\mathfrak{D}$ and $\mathfrak{R}$, respectively. When we are done setting values to all the variables, we go back and look at $\mathfrak{D}$ and $\mathfrak{R}$. The flag bad is set if (and only if) any of these multisets contains some value more than once. This procedure is designed to set bad under exactly the same conditions as in game RND2. The following is thus clear:

$$\Pr[\, A^{\text{RND2}} \text{ sets } bad\,] \;=\; \Pr[\, A^{\text{RND3}} \text{ sets } bad\,] \tag{10}$$

Game NON1. So far we have not changed the structure of the games at all: it has remained an

050 $\mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \emptyset$ // Multisets
051 **for** $s \leftarrow 1$ **to** q **do**
100   **if** $ty^s = \mathsf{Enc}$ **then**
110   Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{P}_1^s \cdots \mathsf{P}_{\mathsf{u}[s]}^s = \mathsf{P}_1^r \cdots \mathsf{P}_{\mathsf{u}[s]}^r$ for some $r < s$
111         $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow PPP_i^r$
112   **for** $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
113             $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$
114             $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$

120   **for** $i \leftarrow 1$ **to** m **do**
121             $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus 2\,(PPP_1^s \oplus PPP_m^s)$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$
122             $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC_i^s\}$

200   **else** $(ty^s = \mathsf{Dec})$
210   Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{C}_1^s \cdots \mathsf{C}_{\mathsf{u}[s]}^s = \mathsf{C}_1^r \cdots \mathsf{C}_{\mathsf{u}[s]}^r$ for some $r < s$
211         $CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow CCC_i^r$
212   **for** $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
213             $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC_i^s\}$
214             $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$

220   **for** $i \leftarrow 1$ **to** m **do**
221             $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus 2\,(CCC_1^s \oplus CCC_m^s)$; $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$
222             $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$; $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$

300 $bad \leftarrow$ (some value appears more than once in $\mathfrak{D}$)
        or (some value appears more than once in $\mathfrak{R}$)

Figure 8: Game NON1 is based on game RND3 but now $\boldsymbol{\tau} = (\mathbf{ty}, \mathbf{P}, \mathbf{C})$ is a fixed, allowed transcript.

adversary asking q questions to an oracle, our oracle answering those questions, and the internal variable *bad* either ending up true or false. The next step, however, actually gets rid of the adversary, as well as all interaction in the game.

We want to bound the probability that *bad* gets set to true in game RND3. We may assume that the adversary is deterministic, and so the probability is over the random choices to the P^s and and C^s values that are made while answering the queries (in lines 011 and 021), and the random choices $PPP_i^s \xleftarrow{\$} \{0,1\}^n$ and $CCC_i^s \xleftarrow{\$} \{0,1\}^{nm}$ that are made in finalization—phase one (lines 114 and 214). We will now eliminate the coins associated to lines 011 and 021. Recall that the adversary asks no pointless queries.

We would like to make the stronger statement that for *any* set of values that might be returned to the adversary at lines 011 and 021, and for any set of queries (none pointless) associated to them, the finalization step of game RND3 rarely sets *bad*. However, this statement isn't quite true. For example, assume that r-th and s-th queries ($r < s$) are both encipher queries, and that the random choices in line 011 specify that the first block in the two answers is the same, $C_1^r = C_1^s$. Then the flag *bad* is sure to be set, since we will have a "collision" between CC_1^r and CC_1^s. Formally, since $CCC_0^r \leftarrow 0^n$ and $CCC_0^s \leftarrow 0^n$ (both in line 111), then we would get in line 131 $CC_1^r = C_1^r \oplus 0^n = C_1^s \oplus 0^n = CC_1^s$ and since the two formal variables CC_1^r and CC_1^s belong to $\mathfrak{R}$ we would set *bad* when we examine them in line 302. A similar example can be given for decipher queries. We call such collisions *immediate* collisions. Formally, an immediate collision happens whenever we have $C_1^r = C_1^s$ ($r < s$) and query s is an encipher query, and whenever we have $P_1^r = P_1^s$ ($r < s$) and query s is a decipher query. Clearly the probability of an immediate collision in game RND3 is at most $\binom{q}{2}/2^n = q(q-1)/2^{n+1}$.

We make from the Finalization part of game RND3 a new game, game NON1 (for "noninter-

050 $\mathfrak{D} \leftarrow \emptyset$ // Multiset
051 **for** $s \leftarrow 1$ **to** q **do**
100   **if** $ty^s = \mathsf{Enc}$ **then**
110   Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{P}_1^s \cdots \mathsf{P}_{\mathsf{u}[s]}^s = \mathsf{P}_1^r \cdots \mathsf{P}_{\mathsf{u}[s]}^r$ for some $r < s$
111         $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 2$ **to** $\mathsf{u}[s]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s, \quad PPP_i^s \leftarrow PPP_i^r$
112   **for** $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
113             $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}}$
114             $PPP_i^s \xleftarrow{\$} \{0,1\}^n$
120   **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus 2\,(PPP_1^s \oplus PPP_m^s)$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}}$
200   **else** ($ty^s = \mathsf{Dec}$)
210   Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{C}_1^s \cdots \mathsf{C}_{\mathsf{u}[s]}^s = \mathsf{C}_1^r \cdots \mathsf{C}_{\mathsf{u}[s]}^r$ for some $r < s$
211         $CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s, \quad CCC_i^s \leftarrow CCC_i^r$
212   **for** $i \in [\mathsf{u}[s]+1 \mathinner{..} m]$ **do** $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}}$
220   **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus 2\,(CCC_1^s \oplus CCC_m^s)$
230   **for** $i \in [1 \mathinner{..} m]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}}$
300 $bad \leftarrow$ (some value appears more than once in $\mathfrak{D}$)

Figure 9: Game NON2. Twice the probability that bad gets set in this game bounds the probability that bad gets set in game NON1. We highlight by shading the random selections in this game, and we highlight by boxing statements that grow $\mathfrak{D}$.

active"). This game silently depends on a fixed transcript $\boldsymbol{\tau} = \langle \mathbf{ty}, \mathbf{P}, \mathbf{C} \rangle$ with $\mathbf{ty} = (\mathsf{ty}^1, \cdots, \mathsf{ty}^q)$, $\mathbf{P} = (\mathsf{P}^1, \cdots, \mathsf{P}^q)$, and $\mathbf{C} = (\mathsf{C}^1, \cdots, \mathsf{C}^q)$ where $\mathsf{ty}^s \in \{\mathsf{Enc}, \mathsf{Dec}\}$ and $\mathsf{P}^s = \mathsf{P}_1^s \cdots \mathsf{P}_m^s$ and $\mathsf{C}^s = \mathsf{C}_1^s \cdots \mathsf{C}_m^s$ for $|\mathsf{P}_i^r| = |\mathsf{C}_i^r| = n$. This fixed transcript may not specify any immediate collisions or pointless queries; we call such a transcript *allowed*. Thus saying that $\boldsymbol{\tau}$ is allowed means that for all $r < s$ we have the following: if $\mathsf{ty}^s = \mathsf{Enc}$ then (i) $\mathsf{P}^s \neq \mathsf{P}^r$ and (ii) $\mathsf{C}_1^s \neq \mathsf{C}_1^r$; while if $\mathsf{ty}^s = \mathsf{Dec}$ then (i) $\mathsf{C}^s \neq \mathsf{C}^r$ and (ii) $\mathsf{P}_1^s \neq \mathsf{P}_1^r$. Now fix an allowed transcript $\boldsymbol{\tau}$ that *maximizes the probability of the flag bad being set*. This one transcript $\boldsymbol{\tau}$ is hardwired into game NON1. We have that

$$\Pr[\, A^{\mathrm{RND3}} \text{ sets } bad \,] \quad \leq \quad \Pr[\, \mathrm{NON1} \text{ sets } bad \,] + q(q-1)/2^{n+1} \tag{11}$$

This step can be viewed as conditioning on the presence or absence of an immediate collision, followed by the usual argument that an average of a collection of real numbers is at most the maximum of those numbers. One can also view the transition from game RND3 to game NON1 as *augmenting* the adversary, letting it specify not only the queries to the game, but also the answers to these queries (as long as it does not specify immediate collisions or pointless queries). In terms of game RND3, instead of having the oracle choose the answers to the queries at random in lines 011 and 021, we let the adversary supply both the queries and the answers. The oracle just records these queries and answers. When the adversary is done, we execute the finalization step as before to determine the bad flag. Clearly such an augmented adversary does not interact with the oracle at all, it just determines the entire transcript, giving it as input to the oracle. Now maximizing the probability of setting bad over all such augmented adversaries is the same as maximizing this probability over all allowed transcripts.

Game NON2. Before we move to analyze the non-interactive game, we make one last change, aimed at reducing the number of cases that we need to handle in the analysis. We observe that due to the complete symmetry between $\mathfrak{D}$ and $\mathfrak{R}$, it is sufficient to analyze the collision probability in just one of them. Specifically, because of this symmetry we can assume w.l.o.g. that in game NON1

$$\Pr[\text{some value appears more than once in } \mathfrak{D}] \geq \Pr[\text{some value appears more than once in } \mathfrak{R}]$$

and therefore $\Pr[\,\text{NON1 sets } bad\,] \leq 2 \cdot \Pr[\text{some value appears more than once in } \mathfrak{D}]$.

We therefore replace the game NON1 by game NON2, in which we only set the flag *bad* if there is a collision in $\mathfrak{D}$. We now can drop the code that handles $\mathfrak{R}$, as well as anything else that doesn't affect the multiset $\mathfrak{D}$. The resulting game is described in Figure 9, and we have

$$\Pr[\,\text{NON1 sets } bad\,] \quad \leq \quad 2 \cdot \Pr[\,\text{NON2 sets } bad\,] \tag{12}$$

D.2 Analysis of the Non-Interactive Game NON2

It is helpful to view the multiset $\mathfrak{D}$ as a set of formal variables (rather than a multiset containing the values that these variables assume). Namely, whenever in game NON2 we set $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{X\}$ for some variable X, we would think of it as setting $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{\text{“}X\text{”}\}$ where “X” is the name of that formal variable. Viewed in this light, our goal now is to bound the probability that two formal variables in $\mathfrak{D}$ assume the same value in the execution of NON2. We observe that the formal variables in $\mathfrak{D}$ are uniquely determined by $\boldsymbol{\tau}$—they don't depend on the random choices made in the game NON2; specifically,

$$\begin{aligned}\mathfrak{D} \quad = \quad & \{PP_i^s \mid \ \mathsf{ty}^s = \mathsf{Dec}\} \ \cup \ \{PP_i^s \mid \ \mathsf{ty}^s = \mathsf{Enc} \text{ and } i > \mathsf{u}[s]\} \ \cup \\ & \{CCC_i^s \mid \ \mathsf{ty}^s = \mathsf{Enc}\} \ \cup \ \{CCC_i^s \mid \ \mathsf{ty}^s = \mathsf{Dec} \text{ and } i > \mathsf{u}[s]\}\end{aligned}$$

We view the formal variables in $\mathfrak{D}$ as *ordered* according to when they are assigned a value in the execution of game NON2. This ordering too is fixed, depending only on the fixed transcript $\boldsymbol{\tau}$.

Throughout the remainder of this section, in all probability claims, the implicit experiment is that of game NON2. We adopt the convention that in an arithmetic or probability expression, a formal variable implicitly refers to its value. For example, $\Pr[X = X']$ means the probability that the value assigned to X is the same as the value assigned to X'. (At times we may still write “X” to stress that we refer to the name of the formal variable X, or value(X) to stress that we refer to the value of X.) The rest of this section is devoted to case analysis, proving the following claim:

Claim 1 For any two distinct variables $X, X' \in \mathfrak{D}$ we have $\Pr[X = X'] \leq 2^{-n}$ □

Before proving Claim 1, we show how to use it to complete the proof of Theorem 1. As there are no more than 2σ variables in $\mathfrak{D}$, we use the union bound to conclude

$$\Pr[\,\text{NON2 sets } bad\,] \leq \binom{2\sigma}{2} / 2^n \tag{13}$$

Combining Lemma 6 with Equations 9, 10, 11, 12, and 13 we are done:

$$\begin{aligned}\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(A) \quad & \leq \quad \mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(A) + q(q-1)/2^{2n+1} \\ & \leq \quad 2 \cdot \Pr[\text{NON2 sets } bad] + q(q-1)/2^{n+1} + q(q-1)/2^{2n+1} \\ & \leq \quad 2 \cdot \binom{2\sigma}{2} / 2^n + q(q-1)/2^{n+1} + q(q-1)/2^{2n+1} \\ & \leq \quad 5\sigma^2/2^n\end{aligned}$$

$$\phi(CCC_i^s) \;=\; \begin{cases} CCC_i^s & \text{if } i > \mathsf{u}[s] \text{ and } \mathsf{ty}^s = \mathsf{Dec} & \text{CCC1} \\ PPP_m^s & \text{if } \mathsf{ty}^s = \mathsf{Enc} & \text{CCC2} \end{cases}$$

$$\phi(PP_i^s) \;=\; \begin{cases} \mathsf{none} & \text{if } i = 1 & \text{PP1} \\ CCC_m^s & \text{if } i > 1 \text{ and } \mathsf{ty}^s = \mathsf{Dec} & \text{PP2} \\ PPP_{i-1}^s & \text{if } i > 1 \text{ and } \mathsf{ty}^s = \mathsf{Enc} \text{ and } i > \mathsf{u}[s] + 1 & \text{PP3} \\ PPP_{i-1}^{\mathsf{r}[s,i-1]} & \text{if } i > 1 \text{ and } \mathsf{ty}^s = \mathsf{Enc} \text{ and } i = \mathsf{u}[s] + 1 \text{ and } \mathsf{ty}^{\mathsf{r}[s,i-1]} = \mathsf{Enc} & \text{PP4} \\ CCC_m^{\mathsf{r}[s,i-1]} & \text{if } i > 1 \text{ and } \mathsf{ty}^s = \mathsf{Enc} \text{ and } i = \mathsf{u}[s] + 1 \text{ and } \mathsf{ty}^{\mathsf{r}[s,i-1]} = \mathsf{Dec} & \text{PP5} \end{cases}$$

Figure 10: Defining the last free variable, $\phi(X)$, associated to formal variable $X \in \mathfrak{D}$. Transcript $\boldsymbol{\tau} = (\mathsf{ty}, \mathsf{P}, \mathsf{C})$ has been fixed and it determines $\mathsf{u}[\cdot]$ and $\mathsf{r}[\cdot,\cdot]$.

Since A was an arbitrary adversary that asks at most q queries, each consisting of m blocks, we conclude that $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{CMC}[\mathrm{Perm}(n)]}(n\sigma) \leq 5\sigma^2/2^n$, as required, which completes Equation (1). The proof for the remainder of Theorem 1 is explained in the discussion following the theorem.

The case analysis. We now need to prove Claim 1. We first prove a few claims (Claims 3 through 7 below), each covering some special cases of collisions, and then go through a systematic case analysis, showing that all possible cases are indeed covered by these claims.

Inspecting the code of game NON2 we see that the only random choices in the game are the selection $PPP_i^s \xleftarrow{\$} \{0,1\}^n$ on encipher (line 114) and the selection $CCC_i^s \xleftarrow{\$} \{0,1\}^n$ on decipher (line 212). Hereafter we refer to these variables as the *free variables* of the game, and we let $\mathfrak{F}$ denote the set of them:

$$\mathfrak{F} \;=\; \{PPP_i^s \mid \; \mathsf{ty}^s = \mathsf{Enc} \text{ and } i > \mathsf{u}[s]\} \;\cup\; \{CCC_i^s \mid \; \mathsf{ty}^s = \mathsf{Dec} \text{ and } i > \mathsf{u}[s]\}$$

The value of any variable in the game NON2 can be expressed as a function in the free variables. To help the analysis, we consider, for each variable $X \in \mathfrak{D}$, the *last free variable* that X depends on, denoted $\phi(X)$. We now explicitly define the function $\phi\colon \mathfrak{D} \to \mathfrak{F} \cup \{\mathsf{none}\}$. For that definition, it is important to keep in mind that since all the queries are of the same length and there are no pointless queries, we have that $\mathsf{u}[s] < m$ for all s. Therefore we have $PPP_m^s \in \mathfrak{F}$ whenever $\mathsf{ty}^s = \mathsf{Enc}$, and $CCC_m^s \in \mathfrak{F}$ whenever $\mathsf{ty}^s = \mathsf{Dec}$.

A formal variable $CCC_i^s \in \mathfrak{D}$ may be assigned a value $PPP_{m+1-i}^s \oplus 2(PPP_1^s \oplus PPP_m^s)$ in line 120 (on encipher), in which case the variable PPP_m^s (which has to be a free variable) is the "last free variable that CCC_i^s depends on", $\phi(CCC_i^s) = PPP_m^s$. Alternatively, CCC_i^s may be chosen at random in line 212, in which case it is itself a free variable, $\phi(CCC_i^s) = CCC_i^s$. The rules for $PP_i^s \in \mathfrak{D}$ are a bit more involved. On either encipher (line 113) or decipher (line 230), the variable PP_i^s is assigned the value $\mathsf{P}_i^s \oplus PPP_{i-1}^s$, so $\phi(PP_i^s)$ must be the same as "the last free variable that PPP_{i-1}^s depends on". Here we must consider several cases:

- If $i = 1$ then PPP_{i-1}^s is the constant zero, and so we denote $\phi(PP_i^s) = \mathsf{none}$.
- If $i > 1$ and this is a decipher query, then PPP_{i-1}^s is assigned a value $CCC_{m+1-i}^s \oplus 2(CCC_1^s \oplus CCC_m^s)$ in line 220, and as CCC_m^s must be a free variable, we have $\phi(PP_i^s) = CCC_m^s$.
- If $i > 1$ and this is an encipher query, and if $i > \mathsf{u}[s] + 1$, then PPP_{i-1}^s is chosen at random in line 114 (at the iteration just before the assignment of PP_i^s). Thus, PPP_{i-1}^s is itself a free variable, so we have $\phi(PP_i^s) = PPP_{i-1}^s$.

- If $i > 1$ and this is an encipher query, and if $i = \mathsf{u}[s]+1$, then PPP^s_{i-1} is assigned the value PPP^r_{i-1} in line 111, where r is some prior query such that $\mathsf{P}^r_1 \cdots \mathsf{P}^r_{i-1} = \mathsf{P}^s_1 \cdots \mathsf{P}^s_{i-1}$. For the purpose of determining the "last free variable that PPP^s_{i-1} depends on" we need to look at the first such query r. Thus, for an encryption query, $\mathsf{ty}^s = \mathsf{Enc}$, we define for any index j,

  $$\mathsf{r}[s,j] \stackrel{\text{def}}{=} \min\{\ r \le s\ :\ \mathsf{P}^r_1 \cdots \mathsf{P}^r_j = \mathsf{P}^s_1 \cdots \mathsf{P}^s_j\}.$$

  Setting $r = \mathsf{r}[s, i-1]$, we know that $i-1 \ge \mathsf{u}[r]+1$ (since the prefix $\mathsf{P}^r_1 \cdots \mathsf{P}^r_{i-1}$ did not appear anywhere before query r). Hence, if query r is an encryption query then PPP^r_{i-1} is itself a free variable, and therefore $\phi(PP^s_i) = PPP^r_{i-1}$. Lastly, if query r is a decryption query then (similarly to the second bullet above), we have $\phi(PP^s_i) = CCC^r_m$.

- The case where this is an encipher query and $i < \mathsf{u}[s]+1$ is not interesting, since in this case $PP^s_i \notin \mathfrak{D}$.

A summary of all these cases appears in Figure 10. We stress that just like the sets $\mathfrak{D}$ and $\mathfrak{F}$, the function ϕ too depends only on the fixed transcript $\boldsymbol{\tau}$ and not on the random choices in the game NON2. Justifying the name "last free variable" we observe the following, which follows from the preceding discussion:

Claim 2 Let $X \in \mathfrak{D}$ be a formal variable, and let $XXX = \phi(X)$. If $XXX \ne \mathsf{none}$ then the value that X assumes in game NON2 can be expressed as value$(X) = a \cdot$ value$(XXX) \oplus \beta$ where $a \in \{1, 2, 3\}$ is a constant (that depends the name of the formal variable X and on the fixed transcript $\boldsymbol{\tau}$) and β is an expression involving only constants and free variables that occur before XXX in the game NON2. □

As an immediate corollary of Claim 2, we get the following.

Claim 3 Let $X, X' \in \mathfrak{D}$ be formal variables such that $\phi(X) \ne \phi(X')$. Then for any fixed $\Delta \in \{0,1\}^n$ we have that $\Pr[X \oplus X' = \Delta] = 2^{-n}$.

Proof: let $XXX = \phi(X)$ and let $XXX' = \phi(X')$, and assume that XXX' occurs before XXX in NON2. By Claim 2, we have $X \oplus X' = a \cdot XXX \oplus b \quad \oplus \quad a' \cdot XXX' \oplus b'$, where a is a constant, and $b \oplus a' \cdot XXX' \oplus b'$ is an expression involving only constants and free variables that occur before XXX. As the value of XXX is chosen at random from $GF(2^n)$, independently of the other free variables, it follows that $\Pr[X \oplus X' = \Delta] = 2^{-n}$. ▌

Claim 3 leaves us with the task of analyzing collisions between two variables that are assigned the same free variable by ϕ. These cases are covered by the following four claims.

Claim 4 For any i and any $r < s$, if $PP^r_i, PP^s_i \in \mathfrak{D}$ with $\phi(PP^r_i) = \phi(PP^s_i)$ there is a fixed $\Delta \in \{0,1\}^n$, $\Delta \ne 0^n$, such that the values assigned to PP^r_i, PP^s_i satisfy $PP^r_i \oplus PP^s_i = \Delta$ with probability one.

Proof: If $i = 1$ then we have $PP^r_1 \oplus PP^s_1 = \mathsf{P}^r_1 \oplus \mathsf{P}^s_1$, which is clearly a constant. If query s is decipher then this constant must be non-zero since the transcript $\boldsymbol{\tau}$ cannot specify immediate collisions. It query s is encipher then the fact that $PP^s_1 \in \mathfrak{D}$ tells us that $1 > u[s]$, which means that P^s_1 is a "new block", different than P^r_1. Thus, here too this constant must be non-zero. From now on we assume that $i \ge 2$.

The conditions $r < s$ and $\phi(PP_i^r) = \phi(PP_i^s)$ imply that $\phi(PP_i^s)$ was assigned by rules PP4 or PP5 from Figure 10: If we were using rules PP2 or PP3, we would get $\phi(PP_i^r) = \phi(PP_i^s) \in \{PPP_{i-1}^s, CCC_m^s\}$, which is impossible since PP_i^r cannot depend on a free variable from a future query $s > r$.

It therefore follows that $\mathsf{ty}^s = \mathsf{Enc}$, $i = u[s]+1$, and we either have $\mathsf{r}[s, i-1] = r$ (if $\phi(PP_i^r)$ was assigned by rules PP2 or PP3) or at least $\mathsf{r}[s, i-1] = \mathsf{r}[r, i-1]$ (if we used rules PP4 or PP5). Either way, from the definition of $\mathsf{r}[\cdot,\cdot]$ we conclude that $\mathsf{P}_1^r \ldots \mathsf{P}_{i-1}^r = \mathsf{P}_1^s \ldots \mathsf{P}_{i-1}^s$, and therefore $PPP_{i-1}^r = PPP_{i-1}^s$. This, in turn, implies that $PP_i^r \oplus PP_i^s = \mathsf{P}_i^r \oplus \mathsf{P}_i^s$, which is a constant. To show that this constant must be non-zero, we note that since $PP_i^s \in \mathfrak{D}$, then $\mathsf{P}_1^s \ldots \mathsf{P}_i^s$ cannot be a prefix of P^r. But since we know that $\mathsf{P}_1^s \ldots \mathsf{P}_{i-1}^s$ is a prefix of P^r, it must be that $\mathsf{P}_i^r \neq \mathsf{P}_i^s$. ∎

Claim 5 For any encipher query s, any two indexes $i \neq j$, and any fixed $\Delta \in \{0,1\}^n$ we have that $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = 2^{-n}$.

Proof: We prove this proposition by induction over s. In fact, it is easier to prove a stronger claim, asserting the same collision probability also for PPP_i^s's. Specifically, our inductive claim is that for any $s \leq q$, any $i \neq j$, and any fixed $\Delta \in \{0,1\}^n$, we have $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = 2^{-n}$ if s is an encipher query, and $\Pr[PPP_i^s \oplus PPP_j^s = \Delta] = 2^{-n}$ if s is a decipher query.

The proposition holds vacuously for $s = 0$ (since there are no such formal variables). Assume now that it holds for any $r < s$, and we prove for s. We prove here just the case of the CCC_i^s's, as the case for PPP_i^s's is completely analogous. (But note that we assume that both cases hold for any $r < s$).

Assume that $i > j$. If $j = 0$, then $CCC_j^s = 0$, while $CCC_i^s = PPP_{m-i+1}^s \oplus 2(PPP_1^s \oplus PPP_m^s)$ (line 120 of game NON2). As PPP_m^s is a free variable, it follows that $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = 2^{-n}$. From now on we assume that $i > j > 0$.

Recall that in line 120 of game NON2 we set $CCC_i^s = PPP_{m-i+1}^s \oplus 2(PPP_1^s \oplus PPP_m^s)$, and $CCC_j^s = PPP_{m-j+1}^s \oplus 2(PPP_1^s \oplus PPP_m^s)$. Therefore we have $CCC_i^s \oplus CCC_j^s = PPP_{i'}^s \oplus PPP_{j'}^s$, where we denote $i' = m - i + 1$ and $j' = m - j + 1$.

Depending on whether i', j' are more than $\mathsf{u}[s]$, the variables $PPP_{i'}^s, PPP_{j'}^s$ may or may not be set equal to previous variables $PPP_{i'}^r, PPP_{j'}^{r'}$, respectively. Either way, we denote by $PPP_{i'}^r$ "the variable that $PPP_{i'}^s$ is set equal to" (which can be $PPP_{i'}^s$ itself), and likewise $PPP_{i'}^{r'}$ is "the variable that $PPP_{j'}^s$ is set equal to". Namely, we have $r \stackrel{\text{def}}{=} \mathsf{r}[s, i']$ and $r' \stackrel{\text{def}}{=} \mathsf{r}[s, j']$. (Observe that $r, r' \leq s$, with $r = s$ iff $i' > \mathsf{u}[s]$ and similarly $r' = s$ iff $j' > \mathsf{u}[s]$.) Since we have $PPP_{i'}^s = PPP_{i'}^r$ and $PPP_{j'}^s = PPP_{j'}^{r'}$, we get

$$CCC_i^s \oplus CCC_j^s = PPP_{i'}^s \oplus PPP_{j'}^s = PPP_{i'}^r \oplus PPP_{j'}^{r'}$$

Recall that $i > j$, and therefore $i' < j'$, which in turn implies that $r \leq r'$ (because if $\mathsf{P}_1^{r'} \ldots \mathsf{P}_{j'}^{r'} = \mathsf{P}_1^s \ldots \mathsf{P}_{j'}^s$ then in particular $\mathsf{P}_1^{r'} \ldots \mathsf{P}_{i'}^{r'} = \mathsf{P}_1^s \ldots \mathsf{P}_{i'}^s$). Recall also that (by definition of $\mathsf{r}[\cdot,\cdot]$), the string $\mathsf{P}_1^{r'} \ldots \mathsf{P}_{j'}^{r'}$ is not a prefix of any prior plaintext. If query r' is an encipher query, this means that the variable $PPP_{j'}^{r'}$ is a free variable, hence $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = \Pr[PPP_{i'}^r \oplus PPP_{j'}^{r'} = \Delta] = 2^{-n}$.

If query r' is decipher and $r < r'$, then $PPP_{j'}^{r'}$ depends on the free variable $CCC_m^{r'}$ (via line 220), but $PPP_{i'}^r$ cannot depend on it, so again $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = \Pr[PPP_{i'}^r \oplus PPP_{j'}^{r'} = \Delta] = 2^{-n}$.

The last remaining case is when query r' is decipher and $r = r'$. Note that this means that $r' \neq s$ (since query s is encipher). That is, we have two variable $PPP_{i'}^{r'}$ and $PPP_{j'}^{r'}$ for some decipher query $r' < s$. We now use the induction hypothesis, thus concluding that also in this case $\Pr[CCC_i^s \oplus CCC_j^s = \Delta] = \Pr[PPP_{i'}^{rr'} \oplus PPP_{j'}^{r'} = \Delta] = 2^{-n}$. ▌

Claim 6 For any two distinct variables $PP_i^r, PP_j^{r'} \in \mathfrak{D}$ such that $\phi(PP_i^r) = \phi(PP_j^{r'}) = CCC_m^s$ for some s, we have that $\Pr[PP_i^r = PP_j^{r'}] \leq 2^{-n}$.

Proof: The case where $i = j$ is covered in Claim 4, where it is shown that $\Pr[CC_i^r = CC_j^{r'}] = 0$. So we now assume that $i \neq j$.

We note again that since CCC_m^s is a free variable, then query s is decipher, and therefore all the variables PP_k^s ($k = 1 .. m$) are in $\mathfrak{D}$, and they are all assigned $\phi(PP_k^s) = CCC_m^s$. In particular, we have $\phi(PP_i^s) = \phi(PP_i^r) = CCC_m^s$, and $\phi(PP_j^s) = \phi(PP_j^{r'}) = CCC_m^s$. Using Claim 4 again, we get with probability one $PP_i^r = PP_i^s \oplus \Delta_i$ and $PPr'_j = PP_j^s \oplus \Delta_j$ for some fixed $\Delta_i, \Delta_j \in \{0,1\}^n$. (Note that here we allow $r = s$ or $r' = s$, so these Δ's may equal to zero.) Hence $PP_i^r \oplus PP_j^{r'} = PP_i^s \oplus PP_j^s \oplus \Delta_i \oplus \Delta_j$.

Recalling that in line 113 of game NON2 we set $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$ and $PP_j^s \leftarrow \mathsf{P}_j^s \oplus PPP_{j-1}^s$, we conclude that

$$PP_i^r \oplus PP_j^{r'} = PP_i^s \oplus PP_j^s \oplus \Delta_i \oplus \Delta_j = PPP_{i-1}^s \oplus PPP_{j-1}^s \oplus \mathsf{P}_i^s \oplus \mathsf{P}_j^s \oplus \Delta_i \oplus \Delta_j$$

Let $\Delta = \mathsf{P}_i^s \oplus \mathsf{P}_j^s \oplus \Delta_i \oplus \Delta_j$. By Claim 5 we get $\Pr[PP_i^r = PP_j^{r'}] = \Pr[PPP_{i-1}^s \oplus PPP_{j-1}^s = \Delta] = 2^{-n}$. ▌

Claim 7 For any free variable CCC_m^s and any $PP_i^r \in \mathfrak{D}$ such that $\phi(PP_i^r) = CCC_m^s$ it is the case that $\Pr[PP_i^r = CCC_m^s] = 2^{-n}$.

Proof: Since $\phi(PP_i^r) = CCC_m^s$, Claim 4 tells us that $PP_i^r = PP_i^s \oplus \Delta$ for some fixed Δ (where $\Delta = 0$ if $r = s$ and $\Delta \neq 0$ otherwise). We therefore get

$$\begin{aligned}
PP_i^r \oplus CCC_m^s &= PP_i^s \oplus \Delta \oplus CCC_m^s \\
&= PPP_{i-1}^s \oplus \mathsf{P}_i^s \oplus \Delta \oplus CCC_m^s \\
&= \Big(CCC_{m-(i-1)+1}^s \oplus 2(CCC_1^s \oplus CCC_m^s)\Big) \oplus \mathsf{P}_i^s \oplus \Delta \oplus CCC_m^s \\
&= \mathsf{P}_j^s \oplus \Delta \oplus 2CCC_1^s \oplus CCC_{m-i+2}^s \oplus 3CCC_m^s
\end{aligned}$$

The last expression is of the form $a \cdot CCC_m^s \oplus b$ where a is a constant ($a = 2$ if $j = 2$ and $a = 3$ otherwise), and b is an expression involving only constants and free variables that occur before CCC_m^s. Hence we have $\Pr[PP_i^r = CCC_m^s] = \Pr[CCC_m^s = b \cdot a^{-1}] = 2^{-n}$. ▌

Proof of Claim 1. All that is left now is to verify that Claims 3 through 7 above indeed cover all the possible types of collisions between $X, X' \in \mathfrak{D}$. It will be convenient to refer by name to the four parts of $\mathfrak{D}$, so we denote

$$\begin{array}{ll}
\mathfrak{D}1 \stackrel{\text{def}}{=} \{PP_i^s \mid \ \mathsf{ty}^s = \mathsf{Dec}\} & \mathfrak{D}2 \stackrel{\text{def}}{=} \{PP_i^s \mid \ \mathsf{ty}^s = \mathsf{Enc} \text{ and } i > \mathsf{u}[s]\} \\
\mathfrak{D}3 \stackrel{\text{def}}{=} \{CCC_i^s \mid \ \mathsf{ty}^s = \mathsf{Enc}\} & \mathfrak{D}4 \stackrel{\text{def}}{=} \{CCC_i^s \mid \ \mathsf{ty}^s = \mathsf{Dec} \text{ and } i > \mathsf{u}[s]\}
\end{array}$$

Now let $X, X' \in \mathfrak{D}$ be two distinct variables, and assume that X occurs *after* X' in game NON2. We partition the analysis to four cases, depending on the part of $\mathfrak{D}$ that contains X.

$X \in \mathfrak{D}4$. Here $X =$ "CCC_i^s" where s is a decryption query and $i > \mathsf{u}[s]$. In this case X is a free variable $\phi(X) = X$, but since X comes after X', it must be that $\phi(X') \neq X$. By Claim 3 we have $\Pr[X = X'] = 2^{-n}$.

$X \in \mathfrak{D}3$. Here $X =$ "CCC_i^s" where s is an encryption query, hence $\phi(X) = PPP_m^s$. If X' is also of this form $X' =$ "CCC_j^s", belonging to the same query s (but having a different index, $i \neq j$) then Claim 5 tells us that $\Pr[X = X'] = 2^{-n}$. In any other case $\phi(X') \neq PPP_m^s$, and again from Claim 3 we get $\Pr[X = X'] = 2^{-n}$.

$X \in \mathfrak{D}1$. Here $X =$ "PP_i^s" where s is a decryption query, hence $\phi(X) = CCC_m^s$. We have two sub-cases, depending on whether X' is of the form "CCC_j^r" or "PP_j^r". Consider first the case that $X' =$ "CCC_j^r" for some r, j. If $X' =$ "CCC_m^s" (i.e., the last CCC variable in the same query s), then by Claim 7 we have $\Pr[X = X'] = 2^{-n}$. In any other case, $\phi(X') \neq CCC_m^s$ so we get the same bound from Claim 3.

As for the other case, $X' =$ "PP_j^r" for some r, j, if $\phi(X') = \phi(X) = CCC_m^s$, then by Claim 6 we get $\Pr[X = X'] \leq 2^{-n}$. Otherwise, we get the usual $\Pr[X = X'] = 2^{-n}$ from Claim 3.

$X \in \mathfrak{D}2$. Here $X =$ "PP_i^s" where s is an encryption query and $i > \mathsf{u}[s]$. Hence $\phi(X)$ is assigned according to one of the rules PP3, PP4 or PP5.

The case where it was assigned according to rule PP5 is handled just as the case $X \in \mathfrak{D}1$ above. In this case we have $\phi(X) = CCC_m^r$ where $r = \mathsf{r}[s, i-1]$. If $X' =$ "CCC_m^r" then by Claim 7 we have $\Pr[X = X'] = 2^{-n}$. If $X' =$ "CCC_j^t" for any other pair $(t, j) \neq (r, m)$, then $\phi(X') \neq CCC_m^t$ and we get the bound from Claim 3. If $X' =$ "PP_j^r" with $\phi(X') = \phi(X) = CCC_m^r$ then we get the bound from Claim 6, and if $X' =$ "PP_j^r" with $\phi(X') \neq \phi(X)$ then we get the bound again from Claim 3.

Assume, then, that $\phi(X)$ is assigned according to one of the rules PP3, or PP4. That is, we have $\phi(X) = PPP_{i-1}^r$, where $r = \mathsf{r}[s, i-1]$ ($r = s$ if $i > \mathsf{u}[s]+1$, and $r < s$ otherwise). Observe that the index of this free variable PPP_{i-1}^r is strictly smaller than m. If $X' =$ "CCC_j^t" for some t, j then $\phi(X') \neq PPP_{i-1}^r$ (since $\phi(X')$ is either some CCC_j^t or PPP_m^t.) Similarly, if $X' =$ "PP_j^t" for $j \neq i$ and some t, then $\phi(X') \neq PPP_{i-1}^r$ (since $\phi(X')$ can be either none, some CCC_j^t or PPP_m^t). In these cases, we get the usual $\Pr[X = X'] = 2^{-n}$ from Claim 3.

The only case left is when $X' =$ "PP_i^t" for some t. Here, either $\phi(X') \neq \phi(X)$, in which case $\Pr[X = X'] = 2^{-n}$ by Claim 3, or $\phi(X') = \phi(X)$, in which case $\Pr[X = X'] = 0$ by Claim 4.

This completes the proof of Claim 1 and with that the proof of Theorem 1 as well. ∎

E Proof of Theorem 2 — Security of the $\mathbb{E} \triangleleft E$ construction

Let A be an adversary with a pair of oracles. Assume that A runs in time t, asks q queries, and these queries total μ bits. Recall that A asks no pointless queries (as previously defined). We consider A's oracles being instantiated with a random instance of any of the following pairs $(\mathcal{E}, \mathcal{D})$:

(1) $\mathcal{E} = \mathbb{E} \triangleleft E$ and $\mathcal{D} = \mathcal{E}^{-1}$

(2) $\mathcal{E} = \mathbb{E} \triangleleft \mathrm{Perm}(n)$ and $\mathcal{D} = \mathcal{E}^{-1}$

(3) $\mathcal{E} = \mathrm{Perm}(\mathcal{M}) \triangleleft \mathrm{Perm}(n)$ and $\mathcal{D} = \mathcal{E}^{-1}$

(4) $\mathcal{E} = \$$ and $\mathcal{D} = \$$, each returning a random string of $|M|$ bits to any query (T, M)

(5) $\mathcal{E} = \mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ where $\mathcal{T} = \{0,1\}^n$ and $\mathcal{D} = \mathcal{E}^{-1}$

Let p_i be the probability that A outputs 1 when its oracles are instantiated as a random pair of oracles selected according to case (i). Let $p_{ij} = p_i - p_j$. So establishing Equation (4) is the same as bounding p_{15}. We use the triangle inequality, noting that $p_{15} \le p_{12} + p_{23} + p_{34} + p_{45}$ and $p_{35} \le p_{34} + p_{35}$. Let t' be as specified in the theorem statement. The following are either easy or standard:

- $p_{12} \le \mathbf{Adv}^{\mathrm{prp}}_{\mathbb{E}}(t', q)$. To show this, construct a distinguisher A_{12} for E and $\mathrm{Perm}(n)$ by simulating $\mathbb{E}$ and its inverse, as needed.
- $p_{23} \le \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t', q, \mu)$. To show this, construct a distinguisher A_{23} for $\mathbb{E}$ (and its inverse) and $\mathrm{Perm}(\mathcal{M})$ (and its inverse), simulating the random permutation $\pi \in \mathrm{Perm}(n)$.
- $p_{45} \le 0.5q^2/2^N$. This fact is easy and general. It follows by Lemma 6.

The main thing, then is to show that $p_{34} \le q^2/2^n + 0.5q^2/2^N$. This is done as follows. First consider the perfect simulation of the p_3-defining environment given in Figure 11. Here the flag *bad* is initialized to false and π and Π are initialized to partial functions that are defined nowhere. We let $\mathrm{Domain}(\pi)$, $\mathrm{Range}(\pi)$, $\overline{\mathrm{Domain}}(\pi)$ and $\overline{\mathrm{Range}}(\pi)$ have their usual meaning, with complements taken relative to $\{0,1\}^n$. We similarly refer to $\mathrm{Domain}(\Pi)$, $\mathrm{Range}(\Pi)$, $\overline{\mathrm{Domain}}_i(\Pi)$ and $\overline{\mathrm{Range}}_i(\Pi)$, where the complements are restricted to strings of length i. We call this "game Real".

Following the usual game-playing approach, we can modify or drop from game Real any statements whose execution only occurs following the setting of *bad* to true and this change will not effect A's ability to set *bad* to true. Thus we modify game Real by dropping the shaded statements, obtaining game Rand. We note that the modified game returns uniform random bits, and so p_{34} can be bounded by probability that A manages to set the flag *bad* in game Rand.

Observe that game Rand provides *no* information to the adversary A about the value of π; all values related to π are xored with a random string that is independent of π. As a consequence, it is easy to compute the probability the A can set the flag *bad*. On the i^{th} enciphering query, the chance that *bad* will get set at lines 12 is at most $(i-1)/2^n$ and the the chance that *bad* will get set at lines 16 is at most $(i-1)/2^N$ (because we have assumed that the shortest message in the message space has at least N bits). The chance that *bad* gets set to true at line 14 uses the fact that all of the $\mathbb{T}$-values are random, and the adversary is given no information about any $\mathbb{T}$-value. Thus this probability is also at most $(i-1)/2^n$. Deciphering queries work identically. Summing, we have that the chance that *bad* gets set to true during the game is at most $q^2/2^n + 0.5q^2/2^N$. This completes the proof.

F Proof of Theorem 3 — $\pm\widetilde{\mathrm{prp}}$-security $\Rightarrow$ $\pm\widetilde{\mathrm{ind}}$-security

Our goal is to show that for any enciphering scheme $\mathcal{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ whose message space $\mathcal{M}$ consists of strings of length at least n bits, and for any t, q, μ, we have that $\mathbf{Adv}^{\pm\widetilde{\mathrm{ind}}}_{\mathcal{E}}(t, q, 2\mu) \le 2\,\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(t', q, \mu) + 2q^2/(2^n - q)$. We show how to construct a PRP attacking adversary B from a left-or-right distinguisher A such that the advantage and resource bounds of the former is related to the advantage and resource bound of the latter according to the formula from above.

The crux of the proof is to show that $\mathbf{Adv}^{\pm\widetilde{\mathrm{ind}}}_{\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})}(q) \le 2\binom{q}{2}/(2^n - q)$. Namely, when we replace $\mathcal{E}$ with a (tweakable) truly random permutation, then any left-or-right distinguisher that asks at most q queries can have advantage of at most $2\binom{q}{2}/(2^n - q)$. The intuition here is that when we have a truly random permutation, the adversary's only information about b comes from "accidental collisions" that happen with probability at most $\binom{q}{2}2^{-n}$. (For example, if the adversary sees the two queries $\mathcal{E}^b_K(T_0, P_0, T_1, P_1) = C$ and $\mathcal{D}^b_K(T'_0, C_0, T_1, C_1) = P_1$, with $C_1 \ne C$, it can conclude that $b = 0$ for sure, since these two queries are not consistent with $b = 1$.)

When A asks $\mathcal{E}\ (T, P)$

if $T \in \text{Domain}(\pi)$ **then** $\mathbb{T} \leftarrow \pi(T)$
    else $\mathbb{T} \xleftarrow{\$} \{0,1\}^n$
        if $\mathbb{T} \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$ $, \mathbb{T} \xleftarrow{\$} \overline{\text{Range}}(\pi)$
$\pi(T) \leftarrow \mathbb{T}$; $\quad X \leftarrow \mathbb{T} \mathbin{\leftarrow\!\!\!\oplus} P$
if $X \in \text{Domain}(\Pi)$ **then** $bad \leftarrow \mathsf{true}$ $, Y \leftarrow \Pi(X)$ **else**
    $Y \xleftarrow{\$} \{0,1\}^{|P|}$
    if $Y \in \text{Range}(\Pi)$ **then** $bad \leftarrow \mathsf{true}$ $, Y \xleftarrow{\$} \overline{\text{Range}}_{|P|}(\Pi)$
$\Pi(X) \leftarrow Y$
return $\mathbb{T} \mathbin{\leftarrow\!\!\!\oplus} Y$

When A asks $\mathcal{D}\ (T, C)$

if $T \in \text{Domain}(\pi)$ **then** $\mathbb{T} \leftarrow \pi(T)$
    else $\mathbb{T} \xleftarrow{\$} \{0,1\}^n$
        if $\mathbb{T} \in \text{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$ $, \mathbb{T} \xleftarrow{\$} \overline{\text{Range}}(\pi)$
$\pi(T) \leftarrow \mathbb{T}$; $\quad Y \leftarrow \mathbb{T} \mathbin{\leftarrow\!\!\!\oplus} C$
if $Y \in \text{Range}(\Pi)$ **then** $bad \leftarrow \mathsf{true}$ $, X \leftarrow \Pi^{-1}(Y)$ **else**
    $X \xleftarrow{\$} \{0,1\}^{|C|}$
    if $X \in \text{Domain}(\Pi)$ **then** $bad \leftarrow \mathsf{true}$ $, X \xleftarrow{\$} \overline{\text{Domain}}_{|C|}(\Pi)$
$\Pi(X) \leftarrow Y$
return $\mathbb{T} \mathbin{\leftarrow\!\!\!\oplus} X$

Figure 11: Game Real perfectly simulates $\text{Perm}(\mathcal{M}) \triangleleft \text{Perm}(n)$. Game Rand drops the shaded statements.

To formalize this intuition, fix an adversary A, and assume without loss of generality that it is deterministic. Also assume that $q < 2^{n-1}$ (otherwise the theorem is vacuously true). We consider two games, denoted Real and Ideal. In both games we run A and build two permutations π_0, π_1 that are intended to be consistent with $b = 0$, $b = 1$ respectively. In the game Real the distribution over the pairs (b, A's view) will be identical to the real left-or-right game, and in the game Ideal the view of A will be independent of the bit b. We will prove the bound on $\mathbf{Adv}^{\pm\widetilde{\text{ind}}}_{\text{Perm}^{\mathcal{T}}(\mathcal{M})}$ by arguing that these two games are very close.

Throughout these games, we maintain for each tweak value $T \in \mathcal{T}$ and each position $b \in \{0,1\}$ a "pool of unused plaintexts", denoted $\mathcal{P}^b_T$ and a "pool of unused ciphertexts", denoted $\mathcal{C}^b_T$. We also keep a flag, *bad*, which is initialized to *false*. We begin both games by choosing a random bit b, setting $\mathcal{P}^i_T = \mathcal{C}^b_T \leftarrow \mathcal{M}$ for all b, T, and then running the adversary A. We respond to A's queries during this run as follows:

AN ENCIPHER QUERY, $\mathcal{E}(T_0, P_0, T_1, P_1)$, $|P_0| = |P_1| = \ell$:

  Pick $C_0 \leftarrow C_1 \leftarrow \{0,1\}^\ell \cap \mathcal{C}^0_{T_0}$
  if $C_1 \notin \cap \mathcal{C}^1_{T_1}$ **then** $bad \leftarrow \mathsf{true}$ $, C_1 \leftarrow \{0,1\}^\ell \cap \mathcal{C}^1_{T_1}$
  return C_b

A DECIPHER QUERY, $\mathcal{D}(T_0, C_0, T_1, C_1)$, $|C_0| = |C_1| = \ell$:

  Pick $P_0 \leftarrow P_1 \leftarrow \{0,1\}^\ell \cap \mathcal{P}^0_{T_0}$
  if $P_1 \notin \cap \mathcal{P}^1_{T_1}$ **then** $bad \leftarrow \mathsf{true}$ $, P_1 \leftarrow \{0,1\}^\ell \cap \mathcal{P}^1_{T_1}$
  return P_b

where the only difference between the two games is that in game Ideal we do not execute the shaded statements, following the setting $bad \leftarrow \mathsf{true}$. After each query, we assign $\pi_0(T_0; P_0) \leftarrow C_0$ and $\pi_1(T_1; P_1) \leftarrow C_1$, and update the sets $\mathcal{P}, \mathcal{C}$ by setting

$$\mathcal{P}^0_{T_0} \leftarrow \mathcal{P}^0_{T_0} - \{P_0\}, \quad \mathcal{P}^1_{T_1} \leftarrow \mathcal{P}^1_{T_1} - \{P_1\}, \quad \mathcal{C}^0_{T_0} \leftarrow \mathcal{C}^0_{T_0} - \{C_0\}, \;\text{ and } \mathcal{C}^1_{T_1} \leftarrow \mathcal{C}^1_{T_1} - \{C_1\}$$

It is important to note that because of the restrictions that are listed in Table 1, as long as the flag *bad* is not set we cannot run into a conflict when defining π_0 and π_1 in this manner.

It is clear that in the game Ideal, the resulting view of A is independent of the bit b, and that as long as *bad* is not set, the distributions of both games are identical. Therefore, we have

$$\Pr_{\text{Real}}[A^{\mathcal{E}^b, \mathcal{D}^b} \Rightarrow b] \;\; \leq \;\; \Pr[\,\text{Real sets } bad\,] + \Pr_{\text{Ideal}}[A^{\mathcal{E}^b, \mathcal{D}^b} \Rightarrow b] \;\; = \;\; \frac{1}{2} + \Pr[\,\text{Real sets } bad\,]$$

It remains to bound the probability $\Pr[\,\text{Real sets } bad\,]$. Assume that the i'th query of the adversary is an encipher query. The size of $\mathcal{C}^0_{T_0} \cap \{0,1\}^\ell$ when this query is asked is at least $2^\ell - i + 1 \geq 2^n - i + 1$. On the other hand, the size of $(\mathcal{C}^0_{T_0} \cap \{0,1\}^\ell) - \mathcal{C}^1_{T_1}$ is at most $i - 1$. Thus, the probability of $C_1 \notin \mathcal{C}^1_{T_1}$ in this step is at most $\frac{i-1}{2^N - i + 1}$. The same bound holds also if this is a decipher query. As there are at most q queries, the total probability of the bad event is at most

$$\sum_{i=1}^{q} \frac{i-1}{2^n - i + 1} < \sum_{i=1}^{q} \frac{i-1}{2^n - q} = \frac{\binom{q}{2}}{2^n - q}$$

We conclude that $\Pr_{\text{Real}}[A^{\mathcal{E}^b, \mathcal{D}^b} \Rightarrow b] \leq \frac{1}{2} + \binom{q}{2}/(2^n - q)$.

We are now ready to show how to construct a PRP attacking adversary B from a left-or-right distinguisher A. Recall that B has two oracles $\mathcal{E}(\cdot,\cdot)$, $\mathcal{D}(\cdot,\cdot)$, and that A has two oracles $\mathcal{E}^*(\cdot,\cdot,\cdot,\cdot)$, $\mathcal{D}^*(\cdot,\cdot,\cdot,\cdot)$. The PRP attacker B begins by choosing a random bit b, and then it runs the left-or-right distinguisher A. When A makes an encryption query $\mathcal{E}^*(T_0, P_0, T_1, P_1)$, B makes an encryption query of its own, $\mathcal{E}(T_b, P_b)$, and returns the answer to A. Similarly, on decryption query $\mathcal{D}^*(T_0, C_0, T_1, C_1)$, B makes a decryption query of its own, $\mathcal{D}(T_b, C_b)$, and returns the answer to A. When A halts, B checks its output. If A outputs a single bit b', and if $b = b'$, then B outputs 1. In any other case, B outputs 0. It is clear that the running time of B is that of A plus an amount linear in the query lengths and B asks its oracle exactly half the number of bits that A does.

By definition, when the oracles of B are implemented by an enciphering scheme $\mathcal{E}$, the probability that A outputs $b' = b$ is exactly $\frac{1}{2}(1 + \mathbf{Adv}^{\pm\widetilde{\text{ind}}}_{\mathcal{E}}(A))$. On the other hand, from the discussion above we know that when the oracles of B are implemented by a tweakable truly random permutation, the probability that A outputs $b' = b$ is at most $\frac{1}{2} + \binom{q}{2}/(2^n - q)$. It follows that

$$\mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{\mathcal{E}}(B) \;\geq\; \frac{1}{2}\left(1 + \mathbf{Adv}^{\pm\widetilde{\text{ind}}}_{\mathcal{E}}(A)\right) - \left(\frac{1}{2} + \frac{\binom{q}{2}}{2^n - q}\right) \;=\; \frac{1}{2}\mathbf{Adv}^{\pm\widetilde{\text{ind}}}_{\mathcal{E}}(A) - \frac{\binom{q}{2}}{2^n - q}$$

G Proof of Theorem 4 — $\pm\widetilde{\text{ind}}$-security $\Rightarrow$ $\pm\widetilde{\text{prp}}$-security

We show how to construct a left-or-right distinguisher A from a PRP attacker B. Recall that A has two oracles $\mathcal{E}^*(\cdot,\cdot,\cdot,\cdot)$, $\mathcal{D}^*(\cdot,\cdot,\cdot,\cdot)$, and that B has two oracles $\mathcal{E}(\cdot,\cdot)$, $\mathcal{D}(\cdot,\cdot)$. Also recall that we can assume that B never makes pointless queries to its oracles.

The left-or-right distinguisher A runs the PRP attacker B, and uses its own oracles to answer the oracle queries of B. Roughly speaking, when B asks a query (T, X), A prepares a query $(T, \$, T, X)$ to its oracle, where \$ is a randomly chosen string of the appropriate length. When the left-or-right bit is 1, the left-or-right oracle always returns the right encryption or the decryption of (T, X), so the oracles of B are implemented by the enciphering scheme. On the other hand, when the left-or-right bit is 0, the oracle returns "random answers", so the oracles of B are implemented by a random permutation. Details follow.

Throughout the execution, A keeps track of the sets of "unused plaintexts" and "unused plaintexts" for each tweak value $T \in \mathcal{T}$. As in the proof of Theorem 3, we denote these sets by $\mathcal{P}_T, \mathcal{C}_T$, respectively. Initially, we have $\mathcal{P}_T = \mathcal{C}_T = \mathcal{M}$ for all T. When B asks an encryption query $\mathcal{E}(T, P)$, A picks at random $P' \xleftarrow{\$} \mathcal{P}_T$, asks its own encryption query $C \leftarrow \mathcal{E}^*(T, P', T, P)$, and return the answer C to B. It then updates $\mathcal{P}_T \leftarrow \mathcal{P}_T - \{P'\}$ and $\mathcal{C}_T \leftarrow \mathcal{C}_T - \{C\}$. Similarly, on decryption query $\mathcal{E}(T, C)$, A picks at random $C' \xleftarrow{\$} \mathcal{C}_T$, asks its own decryption query $P \leftarrow \mathcal{D}^*(T, C', T, C)$, returns the answer P to B, and updates $\mathcal{P}_T \leftarrow \mathcal{P}_T - \{P\}$ and $\mathcal{C}_T \leftarrow \mathcal{C}_T - \{C'\}$. If B halts with output 1, then A does the same. If B halts with any other output, then A outputs 0.

One can see that the running time of A is only slightly more than that of B, the difference being the time that it takes to store and update the sets $\mathcal{P}, \mathcal{C}$ and to pick random element from them. Using standard data structures, this extra time can be made $O(\mu \log \mu)$, where μ is the total number of query bits that B uses. Also, the number of bits that A queries its oracles is exactly twice that of B.

It should be obvious that when the left-or-right bit is set to 1, the oracles of B are indeed implemented by the enciphering scheme $\mathcal{E}$. On the other hand, when the bits is set to zero, then each encryption query is answered by a random "unused ciphertext", and each decryption query is answered by a random "unused plaintext", so the oracle of B are implemented by a tweakable truly random permutation. Hence, we have $\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\text{ind}}}(A) = \mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\text{prp}}}(B)$.

H Proof of Theorem 5 — $\pm\widetilde{\text{ind}}$-security $\Rightarrow$ $\pm\widetilde{\text{nm}}$-security

Let A be an adversary that attacks $\mathcal{E}$ in the $\pm\widetilde{\text{nm}}$-sense and suppose that A runs in time t and asks its oracles at most μ total bits and then A outputs a triple (T, C, P). We construct an adversary B that attacks $\mathbb{E}$ in the $\pm\widetilde{\text{prp}}$ sense.

Adversary B works as follows. It runs A. When A asks its first oracle, $\mathcal{E}$, to encipher some (T, P), adversary B asks its own first oracle, $\mathcal{E}$, to encipher (T, P) and B returns to A the answer C that it receives. When A asks its second oracle, $\mathcal{D}$, to decipher some (T, C), adversary B asks its own second oracle, $\mathcal{D}$, to decipher (T, C) and B returns to A the answer P that it receives. When A halts, outputting a triple (T, C, f), adversary B calls its second oracle, $\mathcal{D}$, on (T, C), getting a result P. Then B computes $f(P)$. If $f(P) = 1$ then adversary B outputs 1 (it is guessing that $(\mathcal{E}, \mathcal{D}) = (\mathcal{E}_K, \mathcal{D}_K)$ is a real enciphering oracle and its inverse) and if $f(P) \neq 1$ then adversary B outputs 0 (it is guessing $(\mathcal{E}, \mathcal{D}) = (\pi, \pi^{-1})$ is a random permutation oracle and its inverse).

Note that B asks its oracles $q + 1$ queries and at most $\sigma + \varsigma$ bits and B runs in time $t + c(\sigma + \varsigma)$ for some absolute constant c. We now proceed to analyze B's advantage in relation to A's. Recall that the advantage of B is

$$\mathbf{Adv}_{\mathcal{E}}^{\pm\widetilde{\text{prp}}}(B) \;=\; \Pr[\, B^{\mathcal{E}_K\ \mathcal{D}_K} \Rightarrow 1\,] - \Pr[\, B^{\pi\ \pi^{-1}} \Rightarrow 1\,] \tag{14}$$

and the advantage of A is

$$\begin{aligned}\mathbf{Adv}^{\pm\widetilde{\mathrm{nm}}}_{\mathcal{E}}(A) \;&=\; \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \leftarrow \mathcal{D}^T_K(C):\; f(P)=1] - \\ &\quad\; \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \xleftarrow{\$} \mathcal{P}^T(C):\; f(P)=1] \end{aligned} \tag{15}$$

By definition of the behavior of B we know that

$$\Pr[\,B^{\mathcal{E}_K\,\mathcal{D}_K} \Rightarrow 1\,] \;=\; \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \leftarrow \mathcal{D}^T_K(C):\; f(P)=1] \tag{16}$$

and so combining Equations 14, 15 and 16 we have that

$$\begin{aligned}
&\mathbf{Adv}^{\pm\widetilde{\mathrm{nm}}}_{\mathcal{E}}(A) \\
&= \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(B) + \Pr[\,B^{\pi\,\pi^{-1}} \Rightarrow 1\,] - \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \xleftarrow{\$} \mathcal{P}^T(C):\; f(P)=1] \\
&= \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(B) + \Pr[(T,C,f) \xleftarrow{\$} A^{\pi\,\pi^{-1}};\, P \leftarrow \pi^{-1}_T(C):\; f(P)=1] - \\
&\qquad\qquad \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \xleftarrow{\$} \mathcal{P}^T(C):\; f(P)=1] \\
&= \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(B) + \\
&\quad \Big(\Pr[(T,C,f) \xleftarrow{\$} A^{\pi\,\pi^{-1}};\, P \leftarrow \pi^{-1}_T(C):\; f(P)=1] - \Pr[(T,C,f) \xleftarrow{\$} A^{\pi\,\pi^{-1}};\, P \xleftarrow{\$} \mathcal{P}^T(C):\; f(P)=1]\Big) + \\
&\quad \Big(\Pr[(T,C,f) \xleftarrow{\$} A^{\pi\,\pi^{-1}};\, P \leftarrow \mathcal{P}^T(C):\; f(P)=1] - \Pr[(T,C,f) \xleftarrow{\$} A^{\mathcal{E}_K\,\mathcal{D}_K};\, P \xleftarrow{\$} \mathcal{P}^T(C):\; f(P)=1]\Big)
\end{aligned}$$

We claim that the first difference is zero and the second is at most $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(t', q+1, \mu+\varsigma)$. To see that the first difference is zero imagine filling into an initially empty table values for $\pi(\cdot,\cdot)$ as needed, in the natural way. When the adversary outputs a triple (T,C,f) either the value for $\pi^{-1}_T(C)$ has already been determined, in which case $P \xleftarrow{\$} \mathcal{P}^T(C)$ is equivalent to the assignment statement $P \leftarrow \pi^{-1}_T(C)$, or else the value for $\pi^{-1}_T(C)$ has not yet been determined, in which case filling in a random but consistent value for $\pi^{-1}_T(C)$ is the same as filling in random but consistent values for all the preimages of C under tweak T and then choosing a random one of these as P. To see that the second difference is at most $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(t', q+1, \mu+\varsigma)$ note that time t' (using standard data-structure methods) and $\mu+\varsigma$ bits of oracle queries is enough to make a distinguisher for $[\mathcal{E}_K\,\mathcal{D}_K]$ and $[\pi, \pi^{-1}]$ that behaves according to A to compute (T,C,f), samples P from $\mathcal{P}^T(C)$, and returns $1 - f(P)$. Thus

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{nm}}}_{\mathcal{E}}(A) \;\leq\; \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(B) + \mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(t', q+1, \mu+\varsigma)$$

and so

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{nm}}}_{\mathcal{E}}(t, \mu, \varsigma) \;\leq\; 2\,\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathcal{E}}(t', q+1, \mu+\varsigma)$$

This completes the proof.