The EMD Mode of Operation
(A Tweaked, Wide-Blocksize, Strong PRP)

Phillip Rogaway *

26 September 2002

Abstract

We describe a block-cipher mode of operation, EMD, that builds a strong pseudorandom permutation (PRP) on nm bits ($m \geq 2$) out of a strong PRP on n bits (i.e., a block cipher). The constructed PRP is also tweaked (in the sense of [10]): to determine the nm-bit ciphertext block $C = \mathbb{E}_K^T(P)$ one provides, besides the key K and the nm-bit plaintext block P, an n-bit tweak T. The mode uses $2m$ block-cipher calls and no other complex or computationally expensive steps (such as universal hashing). Encryption and decryption are identical except that encryption uses the forward direction of the underlying block cipher and decryption uses the backwards direction. We suggest that EMD provides an attractive solution to the disk-sector encryption problem, where one wants to encipher the contents of an nm-bit disk sector in a way that depends on the sector index and is secure against chosen-plaintext/chosen-ciphertext attack.

Key words: block-cipher usage, cryptographic standards, disk encryption, EMD mode, modes of operation, provable security, symmetric encryption.

Note (added Feb 2003): the modes in this paper are wrong

The modes described in this note are wrong: there are simple attacks that distinguish EMD/EME oracles and their inverses from a random permutation and its inverse. The proof in this paper for EMD has a bug in the final case analysis in Appendix A.3. The attacks were found by Antoine Joux, “Cryptanalysis of the EMD Mode of Operation”, to appear at *Eurocrypt 2003*.

This note was not published anywhere, and my first inclination was to withdraw it from ePrint as well. But that would leave Joux with a *Eurocrypt* paper attacking a scheme that is described nowhere. This seemed undesirable. So I'm going to leave this buggy manuscript up on ePrint, unchanged apart from this note, at least for a while.

A new version of this paper, “A Tweakable Enciphering Mode”, has been written and will be distributed soon. The paper is joint with Shai Halevi. In the new paper Shai and I fix the buggy mode and its proof.

* Department of Computer Science, University of California, Davis, California, 95616, USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, 50200 Thailand. E-mail: `rogaway@cs.ucdavis.edu` WWW: `www.cs.ucdavis.edu/~rogaway/`

1 Introduction

Motivation. Suppose you want to encipher each 512-byte sector on a disk. A plaintext disk sector P having index T is to be replaced by a ciphertext disk sector $C = \mathbb{E}_K^T(P)$ where K is a secret key. It is necessary that C have the same length as P; a block cipher $\mathbb{E}$, and not a semantically-secure encryption scheme, is what we want. But block-cipher $\mathbb{E}$, with a blocksize of 512 bytes, should be built a standard block-cipher E that has, say, 16-byte blocks. How should this be done?

The attack-model envisages a chosen-plaintext/chosen-ciphertext attack: the adversary can learn the ciphertext C for any plaintext P and "tweak" T that it chooses, and it can learn the plaintext P for any ciphertext C and tweak T. Any change in a plaintext should give a completely unpredictable ciphertext, and any change in the ciphertext should give a completely unpredictable plaintext. Identical plaintexts with different tweaks should encrypt to unrelated ciphertexts, and identical ciphertexts with different tweaks should decrypt to unrelated plaintexts. Slightly more formally, we want a *strong*, *tweaked*, *pseudorandom permutation* (PRP): for a random key K, each permutation $\mathbb{E}_K^T$ and its inverse $\mathbb{D}_K^T$ should be indistinguishable from random permutation Π^T and its inverse $\coprod^T$.

Conventional modes, like CBC with an IV of T, don't solve this problem. They can't by their very structure: the first block of ciphertext doesn't even depend on all of the blocks of plaintext. The most reasonable known approach to construct the desired kind of object is to follow Naor and Reingold [15, 16], who give a general method for turning a block cipher into a long-blocksize block-cipher. Their papers were our starting point.

EMD Mode. In this paper we propose a new mode of operation, EMD mode. Given a block cipher $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a number $m \geq 2$, EMD mode provides a tweaked block-cipher $\mathbb{E} = \text{EMD}[E,m] = \text{EMD-}E$ where $\mathbb{E}: \mathcal{K} \times \{0,1\}^n \times \{0,1\}^{nm} \to \{0,1\}^{nm}$. We call n the *block length* (and it is also the *tweak length* in our construction) and nm is the *sector length* (in bits) and m is the *blocks per sector*. As a tweaked block-cipher, each $\mathbb{E}_K^T(\cdot) = \mathbb{E}(K,T,\cdot)$ is permutation on $\{0,1\}^{nm}$. We denote the inverse of this permutation by $\mathbb{D}_K^T$.

EMD has the following characteristics: (1) It uses exactly $2m$ block-cipher calls. (2) It uses no other costly operations (in particular, EMD uses no universal hashing). (3) It uses just the one key K for the underlying block cipher—no additional key material is needed. (4) Encryption by $\mathbb{E}$ uses only the forward direction of the block cipher E, while decryption by $\mathbb{D}$ uses only the backward direction of the block cipher, $D = E^{-1}$. (5) The mode is completely symmetric: encryption is identical to decryption except for using D in place of E. (6) The mode is as cache-efficient as one can hope for in a strong PRP. (7) The method is simple to understand and easy to implement. The above set of characteristics make EMD-E attractive in both hardware and software as long as E is. We emphasize that EMD does not change the internals of any cryptographic primitive; that the inclusion of a tweak T adds considerable versatility and ease of correct use; and that the mode is fully specified—there are no missing pieces (like a universal hash function) left to fill in.

The name "EMD" is meant to suggest Encrypt–Mask–Decrypt. Namely, to encipher with EMD one encrypts the plaintext P to form an intermediate value PPP; then one computes a mask $MASK$ from PPP and the tweak T and xors $MASK$ with PPP to give an intermediate value CCC; finally, one decrypts CCC to form the ciphertext C. For a preview, see Figures 1 and 2.

Provable security of EMD. We prove EMD-E secure, in the sense of a strong (tweaked) PRP, assuming that E itself is secure as a strong PRP. The actual results are quantitative, showing that an adversary that attacks $\mathbb{E} = \text{EMD-}E$ can be turned into one for attacking E with the usual quadratic degradation in security (namely, proven security falls off in $5m^2q^2/2^n$ where q is the number of queries to $\mathbb{E}$ or $\mathbb{D}$ and and m is the number of blocks per sector and n is the block size.) The proof uses the game-substitution approach found in works like [9], reducing the analysis of EMD to the computation that a flag *bad* is set in a particular probabilistic program.

Origin of this paper. Our work on this topic grew out of a request for algorithms from the IEEE Security in Storage Working Group (SISWG) [8]. The working group chair, Jim Hughes, described the problem directly to the author, leading to the current work.

Prior work. Naor and Reingold give an elegant approach for making a strong PRP on N bits from a block cipher on $n < N$ bits [15, 16]. Their method involves applying to the input a $K1$-keyed permutation from N bits to N bits, then enciphering the result (say in ECB mode) using a second key $K2$, and then applying to the result the inverse of a $K3$-keyed permutation from N bits to N bits. Their work stops short of fully specifying a mode of operation, but in [15] they come closer, showing how to make the keyed permutations out of xor-universal hash-functions. It is certainly possible to give a practical and fully specified realization of [15, 16], and to add in a tweak as well. Indeed our first approach was to do exactly that. Our work evolved in a different direction when we could not find a realization of NR mode that as simple and efficient as something resembling two passes of CBC.

Another construction for a long blocksize strong PRP appears in unpublished work of [5]. No proof of correctness was offered and the scheme uses about $3m$ block-cipher calls—the same as [15] with an xor-universal hash-function built from CBC. Yet another long block-size block-cipher is constructed by [3], but it does not yield a strong PRP.

The notion of a tweaked block-cipher is due to Liskov, Rivest and Wagner [10]. Earlier work by Schroeppel describes an innovative block cipher that was already designed to incorporate a tweak [19]. Pseudorandom permutations (PRPs) were first defined and constructed by Luby and Rackoff [11, 12], who also considered strong ("super") PRPs and offered the viewpoint of modeling blocks ciphers as PRPs. The concrete-security treatment of PRPs begins in [2].

An ad. hoc suggestion we have seen for disk-sector encryption [8] is forward-then-backwards PCBC mode [14]. The mode is easily broken in the sense of a strong PRP. A completely different approach for disk-sector encryption is to build a wide-blocksize block-cipher from scratch. Such attempts include block ciphers BEAR, LION, and Mercy [1, 6]. A non-block-cipher primitive designed for disk-sector encryption is the pseudorandom function SEAL [18].

Concurrent work. Shai Halevi has been working on the same problem and has invented modes of his own [7]. We haven't seen a writeup and don't yet know the details.

Publication note. This note is an early version of a paper that is still evolving. It is being released now as a service to the IEEE SISWG. Though the EMD algorithm will not change, additional material will be added to this paper prior to its publication.

2 Specification of EMD

Notation. Fix a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. This means that $n \geq 1$ and $\mathcal{K}$ is a finite nonempty set and $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$ for each $K \in \mathcal{K}$. We denote by $D = E^{-1}$ the inverse of block cipher E, namely, $X = D_K(Y)$ if $E_K(X) = Y$. In practice, a typical choice for E will be AES128, whence $n = 128$.

A tweaked block-cipher is a function $\mathbb{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite nonempty set and $\mathcal{T}$ is a nonempty set and $n \geq 1$ and $\mathbb{E}(K, T, \cdot) = E_K^T(\cdot)$ is a permutation on $\{0,1\}^n$. We denote by $\mathbb{D} = \mathbb{E}^{-1}$ the inverse of a tweaked block-cipher $\mathbb{E}$: $X = \mathbb{D}_K^T(Y)$ if $\mathbb{E}_K^T(X) = Y$.

Let $\mathrm{GF}(2^n)$ denote the field with 2^n points. We interchangeably think of a point S in $\mathrm{GF}(2^n)$ as an abstract point in the field, as an n-bit string $S_{n-1} \ldots S_1 S_0 \in \{0,1\}^n$, or as as the formal polynomial $S(\mathsf{x}) = S_{n-1}\mathsf{x}^{n-1} + \cdots + S_1\mathsf{x} + S_0$ with binary coefficients. To add two points, $S \oplus T$, 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 $p_{128}(\mathsf{x}) = \mathsf{x}^{128} + \mathsf{x}^7 + \mathsf{x}^2 + \mathsf{x} + 1$. It is easy to multiply $S = a_{n-1} \cdots a_1 a_0$ by x, which we denote $\mathsf{multx}\,(S)$ or $S \cdot \mathsf{x}$. We illustrate the process for for $n = 128$, in which case

$$\mathsf{multx}\,(S) \;=\; \begin{cases} S \ll 1 & \text{if } \mathsf{msb}(S) = 0 \\ (S \ll 1) \oplus \mathrm{const87} & \text{if } \mathsf{msb}(S) = 1 \end{cases}$$

where const87 is $0^{120}10000111$. Here $S \ll 1$ is the left shift of $S = S_{n-1} \ldots S_1 S_0$ by one bit, namely, $S \ll 1 = S_{n-2} S_{n-3} \cdots S_1 S_0\, 0$ and $\mathsf{msb}(S)$ is the first bit of S (that is, S_{n-1}).

Specification. Fix a constant $m \geq 2$. We construct from block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweaked block-cipher $\mathbb{E}\colon \mathcal{K} \times \{0,1\}^n \times \{0,1\}^{mn} \to \{0,1\}^{mn}$. We specify in Figure 1 both the forward direction of our construction, $\mathbb{E} = \mathrm{EMD}[E, m]$, and its inverse $\mathbb{D}$. In that figure, all capitalized variables except for K are n-bit strings. Key K is an element of $\mathcal{K}$. An illustration of EMD-mode is given in Figure 2.

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

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

$Mask \leftarrow \mathsf{multx}\,(PPP_1 \oplus PPP_m) \oplus T$
for $i \in [1 \,..\, m]$ **do** $CCC_i \leftarrow PPP_{m+1-i} \oplus Mask$

$CCC_0 \leftarrow 0^n$
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 $\mathbb{D}_K^T(C_1 \cdots C_m)$

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

$Mask \leftarrow \mathsf{multx}\,(CCC_1 \oplus CCC_m) \oplus T$
for $i \in [1 \,..\, m]$ **do** $PPP_i \leftarrow CCC_{m+1-i} \oplus Mask$

$PPP_0 \leftarrow 0^n$
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 1: EMD mode in the encipher direction (left) and in the decipher direction (right).

We refer to $X \in \{0,1\}^n$ a *block* and we call $X \in \{0,1\}^{nm}$ a *sector*. Thus we call m the *blocks per sector*. We refer to n as the *block size*, while nm the *sector size* measured in bits and $nm/8$ is the sector size measured in bytes and m is the sector size measured in blocks.

3 Definitions

In this section we recall definitions for the security of block ciphers and tweaked block-ciphers. The definitions are adapted from [2, 10, 12].

For $n \geq 1$ is a number, let $\mathrm{Perm}(n)$ denote the set of all permutations $\pi\colon \{0,1\}^n \to \{0,1\}^n$. For $n \geq 1$ a number and $\mathcal{T}$ a nonempty set, let $\mathrm{Perm}^{\mathcal{T}}(n)$ denote the set of all functions $\pi\colon \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\pi(T, \cdot) \in \mathrm{Perm}(n)$ for all $T \in \mathcal{T}$. Let $\mathrm{Perm}^t(n)$ denote $\mathrm{Perm}^{\mathcal{T}}(n)$ where $\mathcal{T} = \{0,1\}^t$. A *block*

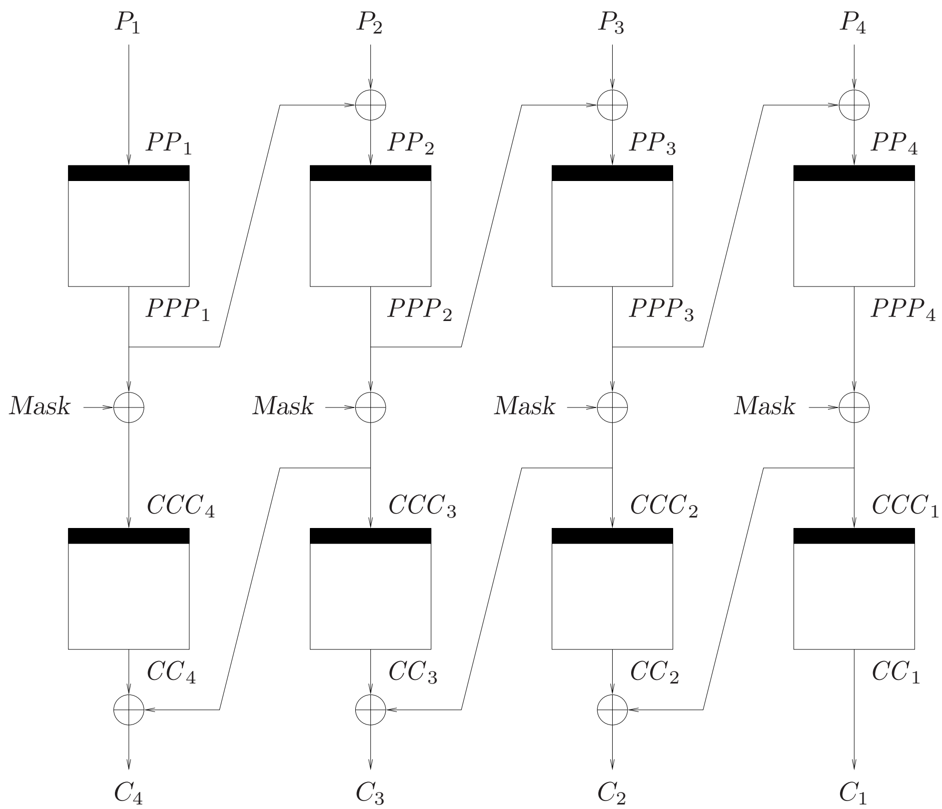

Figure 2: EMD mode for a message of $m = 4$ blocks. The boxes represent the block cipher E. We set $Mask = \mathsf{multx}(PPP_1 \oplus PPP_m) \oplus T$. This value can also be computed as $\mathsf{multx}(CCC_1 \oplus CCC_m) \oplus T$.

cipher is a map $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite, nonempty set, $n \geq 1$ is a number, and $E_K(\cdot) = E(K, \cdot) \in \mathrm{Perm}(n)$ for all $K \in \mathcal{K}$. A *tweaked block-cipher* is a map $\mathbb{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^N \to \{0,1\}^N$ where $\mathcal{K}$ is a finite, nonempty set, $\mathcal{T}$ is a nonempty set, $N \geq 1$ is a number, and $\mathbb{E}_K^T(\cdot) = \mathbb{E}(K, T, \cdot) \in \mathrm{Perm}(N)$ for all $K \in \mathcal{K}$ and $T \in \mathcal{T}$. Note that we can consider $\mathrm{Perm}(n)$ as a block cipher (one key K names each permutation $\pi \in \mathrm{Perm}(n)$) and we can consider $\mathrm{Perm}^{\mathcal{T}}(N)$ as a tweaked block-cipher (one key K names the permutation for each tweak T). The inverse of a block cipher E is the block cipher $D = E^{-1}$ defined by $D_K(Y) = X$ iff $E_K(X) = Y$. The inverse of a tweaked block-cipher $\mathbb{E}$ is the tweaked block-cipher $\mathbb{D} = \mathbb{E}^{-1}$ defined by $\mathbb{D}_K^T(Y) = X$ iff $\mathbb{E}_K^T(X) = Y$.

An *adversary* A is an algorithm with access to zero or more oracles, which we denote $A^{f\,g\,\cdots}$. When we write $A^{f\,g}$ it is only a matter of viewpoint if A has two oracles or one (the one oracle taking an extra, initial, argument to indicate if the query is directed to f or to g). Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let A be an adversary. We define security in the sense of a strong PRP using

$$\mathbf{Adv}_E^{\pm\mathrm{prp}}(A) \;=\; \Pr[K \xleftarrow{\$} \mathcal{K}:\; A^{E_K(\cdot)\,E_K^{-1}(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n):\; A^{\pi(\cdot)\,\pi^{-1}(\cdot)} \Rightarrow 1]$$

The notation shows an experiment to the left of the colon and an event to the right of the colon and we are looking at the probability of that event after performing the specified experiment. By $A^{\mathcal{O}} \Rightarrow 1$ we mean the event that A (with oracle $\mathcal{O}$) returns the bit 1. We will sometimes simplify the notation, when the context is sufficient, by omitting the experiment or the placeholder-arguments of the oracle.

Similarly, if $\mathbb{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a tweaked block-cipher and A is an adversary we define security in the sense of a strong, tweaked PRP using

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

There is no loss of generality in the definitions above to assume that regardless of responses that 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, 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 *purposeless* because the adversary "knows" the answer that it should receive. A query is called *valid* if it is well-formed and not purposeless. A sequence of queries and their responses is valid if every query is the sequence is valid. We henceforth assume that adversaries ask only valid queries.

For E a block cipher we let $\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(q)$ be the maximal value of $\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(A)$ over adversaries that ask at most q oracle queries. Define $\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t, q)$ in the same way except that the adversary is also limited to running time of at most t. By convention, running time includes description size. We similarly define $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{E}}(q)$ and $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbb{E}}(t, q)$ for a tweaked block cipher $\mathbb{E}$.

4 Security Theorem

The information-theoretic statement of security is as follows. The proof is given in Appendix A.

Theorem 1 Fix numbers $n \geq 1$ and $m \geq 2$. Then

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EMD}[\mathrm{Perm}(n),m]}(q) \leq \frac{5m^2q^2}{2^n} \qquad \Box$$

As usual, one can easily pass to the corresponding, complexity-theoretic assertion. The assumption needed of the underlying block cipher is that it be secure in the sense of a strong PRP.

5 Extensions

In this section we sketch some forthcoming extensions.

Variable input lengths. A tweaked, variable-input-length (VIL) cipher is a map $\mathbb{E}: \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} \subseteq \{0,1\}^*$ may have strings of various lengths and $\mathbb{E}^T_K(\cdot)$ is a permutation and $|M| = |\mathbb{E}_K(M)|$ for all $M \in \mathcal{M}$ [3]. It is straightforward to adapt the notion of a strong, tweaked PRP to give a notion of security for VIL ciphers. Interestingly, EMD, with no changes at all, is already secure as a VIL cipher. The domain of messages $\mathcal{M} = \{0,1\}^{2n}(\{0,1\}^n)^*$ is all strings having two or more blocks.

Dealing with message of arbitrary lengths. It is possible to extend the domain of EMD not only to $\{0,1\}^{2n}(\{0,1\}^n)^*$ but to all of $\{0,1\}^{\geq 2n}$. Our approach for dealing with short final blocks is a general one. First the plaintext P is partitioned into $P'P''$ where $|P'|$ is a multiple of n and $|P''| < n$. Next one enciphers P' to C' using the VIL cipher but augmenting the given tweak by P''. Then C' is partitioned into C^*C'' where $|C''| + |P''| = n$. Next one enciphers $C'' \parallel P''$ using some bits from C^* as a tweak, thus getting C^{**}. The ciphertext is C^*C^{**}.

A parallelizable, pinelineable realization. We describe a different way of instantiating the Encrypt–Mask–Decrypt approach, motivated by an exchange with Shai Halevi [7]. Let $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. For $L \in \{0,1\}^n$ and $i \in [0 \mathinner{..} 2^{n-1}]$ let iL be the n-bit string which is the product, in $\mathrm{GF}(2^n)$, of L and the binary string that represents i. (Multiplication of strings is defined using a customary representation of field points, say the one used in [4, 17].) Let $P = P_1 \cdots P_m$ be the message we wish to encrypt, and assume that m is even. Then realize *Encrypt*() by way of $PPP = \mathit{Encrypt}_K(P) = E_K(P_1 \oplus L) \parallel E_K(P_2 \oplus 2L) \parallel \cdots \parallel E_K(P_m \oplus mL)$ where $L = E_K(0^n)$; realize *Mask*() by way of xoring each block PPP_i with $\mathit{Mask} = \mathsf{multx}\,(PPP_1 \oplus \cdots \oplus PPP_m) \oplus T$; and realize *Decrypt*() as the inverse of *Encrypt*() except, as with EMD, we prefer to *Decrypt*() under the reverse orientation of the block cipher. The pseudocode for the resulting mode, which we call EME, is given in Figure 3. Note that because m is even we have that $PPP_1 \oplus \cdots \oplus PPP_m = CCC_1 \oplus \cdots \oplus CCC_m$ which is what allows *Mask* to be computed for both $\mathbb{E}$ and $\mathbb{D}$. Standard tricks make computing the sequence of offsets L, $2L$, $3L$, ... an easy task.

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

$L \leftarrow E_K(0^n)$

for $i \in [1 .. m]$ **do**
    $PP_i \leftarrow P_i \oplus iL$
    $PPP_i \leftarrow E_K(PP_i)$

$\mathit{Mask} \leftarrow \mathsf{multx}\,(PPP_1 \oplus \cdots \oplus PPP_m) \oplus T$
for $i \in [1 .. m]$ **do** $CCC_i \leftarrow PPP_i \oplus \mathit{Mask}$

for $i \in [1 .. m]$ **do**
    $CC_i \leftarrow E_K(CCC_i)$
    $C_i \leftarrow CC_i \oplus iL$

return $C_1 \cdots C_m$

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

$L \leftarrow E_K(0^n)$

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

$\mathit{Mask} \leftarrow \mathsf{multx}\,(CCC_1 \oplus \cdots \oplus CCC_m) \oplus T$
for $i \in [1 .. m]$ **do** $PPP_i \leftarrow CCC_i \oplus \mathit{Mask}$

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

return $P_1 \cdots P_m$

Figure 3: EME mode in the encipher direction (left) and in the decipher direction (right).

Though we do not include a proof with the current writeup, we believe that it is straightforward to adapt the proof of EMD in order to prove security of EME, and with essentially the same bounds as those of Theorem 1.

6 Acknowledgments

Many thanks to Mihir Bellare, who began work on this problem with me and promptly broke my first double-CBC-like attempts at a solution. Thanks to Jim Hughes, who, at Eurocrypt 2001, explained this problem to me and invited me to work on it. I had early and useful conversation with John Black at the same conference.

Phil Rogaway received support from NSF grant CCR-0085961 and a gift from CISCO Systems. Many thanks for their kind support. This work was carried out while Phil was physically resident 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] 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.

[3] 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.

[4] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT '01*, volume 2332 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

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

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

[7] S. Halevi. Personal communication. September 2002.

[8] J. Hughes. IEEE Security in Storage Working Group. Homepage at www.siswg.org. Call for algorithms at www.mail-archive.com/cryptography@wasabisystems.com/msg02102.html, May 2002.

[9] 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.

[10] 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/.

[11] M. Luby and C. Rackoff. A study of password security. In *Advances in Cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

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

[13] U. Maurer. Indistinguishability of random sources. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002.

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

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

[16] 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/.

[17] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *Eighth ACM Conference on Computer and Communications Security (CCS-8)*. ACM Press, 2001.

[18] P. Rogaway and D. Coppersmith. A software-optimized encryption algorithm. *Journal of Cryptology*, 11(4):273–287, Fall 1998. www.cs.ucdavis.edu/~rogaway.

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

A Proof of Theorem 1

We break the proof into three parts: (1) specifying the $\mathbf{Adv}_{\mathbb{E}}^{\$\,\$}$-measure for security of a $\mathcal{T}$-tweaked PRP $\mathbb{E}$; (2) doing a game-playing analysis in order to reduce the analysis of EMD to the analysis of a simpler probabilistic game; and (3) analyzing that game.

A.1 Distinguishability from Random Bits as a Measure of a Tweaked Strong PRP

Let $\mathbb{E}\colon \mathcal{K}\times\mathcal{T}\times\{0,1\}^N \to \{0,1\}^N$ be a tweaked block-cipher. Define the advantage of distinguishing $\mathbb{E}$ from random bits, $\mathbf{Adv}_{\mathbb{E}}^{\$\,\$}$, by

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

where $\$(\cdot,\cdot)$ is the oracle that returns a random N-bit string in response to each query. We insist that A makes no purposeless queries (defined at the end of Section 3) regardless of oracle responses. We extend the definition in the usual way to its resource-bounded versions. We have the following:

Proposition 2 Let $\mathbb{E}\colon \mathcal{K}\times\mathcal{T}\times\{0,1\}^N \to \{0,1\}^N$ be a tweaked block-cipher and let A be an adversary that makes at most q total oracle queries. Then

$$|\mathbf{Adv}_{\mathbb{E}}^{\pm\widetilde{\mathrm{prp}}}(A) - \mathbf{Adv}_{\mathbb{E}}^{\$\,\$}(A)| \;\le\; q(q-1)/2^{N+1} \qquad \square$$

The proof follows the well-known argument relating PRP-security to PRF-security [2]. 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 a collision occurs between the ith item added to sets $\mathcal{X}$ and $\mathcal{Y}$ and items already in those sets. Then $|\mathbf{Adv}_{\mathbb{E}}^{\pm\widetilde{\mathrm{prp}}}(A) - \mathbf{Adv}_{\mathbb{E}}^{\$\,\$}(A)| = |\Pr[\,A^{\mathbb{E}_K\,\mathbb{D}_K} \Rightarrow 1\,] - \Pr[\,A^{\pi\,\pi^{-1}} \Rightarrow 1\,] - \Pr[\,A^{\mathbb{E}_K\,\mathbb{D}_K} \Rightarrow 1\,] + \Pr[\,A^{\$\,\$} \Rightarrow 1\,]| = |\Pr[\,A^{\$\,\$} \Rightarrow 1\,] - \Pr[\,A^{\pi\,\pi^{-1}} \Rightarrow 1\,]| = |\Pr[\,A^{\$\,\$} \Rightarrow 1\,|\,\mathsf{C}]\Pr[\mathsf{C}] + \Pr[\,A^{\$\,\$} \Rightarrow 1\,|\neg\mathsf{C}](1 - \Pr[\mathsf{C}]) - \Pr[\,A^{\pi\,\pi^{-1}} \Rightarrow 1\,]| \le |zy + x(1-y) - x| = |y(z-x)| \le y$ 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 ith query will cause C_i with probability at most $(i-1)/2^N$.

A.2 The Game-Substitution Sequence

Let n, m, q all be fixed. Let A be an adversary that asks q oracle queries (none purposeless), each of nm bits. Our goal in this subsection is to show that $\mathbf{Adv}_{\mathrm{EMD}[\mathrm{Perm}(n)]}^{\$\,\$}(A) \le \Pr[\,\mathrm{NON3}\text{ sets } \mathit{bad}\,] + q^2m^2/2^n$ where NON3 is some probability space and "NON3 sets *bad*" is an event defined there. Later, in Section A.3, we bound $\Pr[\,\mathrm{NON3}\text{ sets } \mathit{bad}\,]$, and, putting that together with Proposition 2, we will get Theorem 1.

Game NON3 is obtained by a game-substitution argument, as carried out in works like [9]. The goal is to simplify the rather complicated setting of A adaptively querying oracle $\mathbb{E}_\pi\,\mathbb{D}_\pi$ where $\mathbb{E} = \mathrm{EMD}[\mathrm{Perm}(n), m]$ and $\mathbb{D} = \mathbb{E}^{-1}$ and $\pi \xleftarrow{\$} \mathrm{Perm}(n)$. We want 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.

Initialization:

$\pi \xleftarrow{\$} \mathrm{Perm}(n)$

To respond to an oracle query $\mathsf{Enc}(T,\ P_1 \cdots P_m)$:

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

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

for $i \leftarrow 1$ **to** m **do**
  $CC_i \leftarrow \pi(CCC_i); \quad C_i \leftarrow CC_i \oplus CCC_{i-1}$

return $C_1 \cdots C_m$

To respond to an oracle query $\mathsf{Dec}(T,\ C_1 \cdots C_m)$:

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

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

for $i \leftarrow 1$ **to** m **do**
  $PP_i \leftarrow \pi^{-1}(PPP_i); \quad P_i \leftarrow PP_i \oplus PPP_{i-1}$

return $C_1 \cdots C_m$

Figure 4: Game EMD1, above, mimics the definition of EMD-mode to provide a perfect realization of an $\mathbb{E}_\pi\,\mathbb{D}_\pi$ oracle where $\mathbb{E} = \mathrm{EMD}[\mathrm{Perm}(n), m]$ and $\mathbb{D} = \mathbb{E}^{-1}$ and $\pi \xleftarrow{\$} \mathrm{Perm}(n)$.

Game EMD1 We begin by defining a game—specifically, the behavior of a probabilistic and stateful oracle—that exactly captures what A sees when interacting with an oracle $\mathbb{E}_\pi\,\mathbb{D}_\pi$ where $\mathbb{E} = \mathrm{EMD}[\mathrm{Perm}(n), m]$ and $\mathbb{D} = \mathbb{E}^{-1}$ and $\pi \xleftarrow{\$} \mathrm{Perm}(n)$. When describing games we will denotes A's oracle by $\mathsf{Enc}\,\mathsf{Dec}$. That means that the adversary's queries are tagged with a type, Enc or Dec, and the queries get answered using a mechanism associated that type. Game EMD1, described in Figure 4, specifies how to answer Enc and Dec queries in a way that exactly mimics the definition of EMD mode. Because it so closely follows the definition of EMD an inspection of that game makes clear that A receives an identical view if interacting with $\mathbb{E}_\pi\,\mathbb{D}_\pi$ (for a random permutation π) or with the oracle of game EMD1. Thus, in particular, we have that

$$\Pr[\, A^{\mathbb{E}_\pi\,\mathbb{D}_\pi} \Rightarrow 1\,] \quad = \quad \Pr[\, A^{\mathrm{EMD1}} \Rightarrow 1\,] \tag{1}$$

Game RND1 For completeness and to help further establish our notation we also specify as a game the oracle $\mathsf{Enc}\,\mathsf{Dec}$ which coincides with the oracle $\$\,\$$ used to define $\mathbf{Adv}^{\$\,\$}$. We write out the oracle in Figure 5. We have immediately that

$$\Pr[\, A^{\$\,\$} \Rightarrow 1\,] \quad = \quad \Pr[\, A^{\mathrm{RND1}} \Rightarrow 1\,] \tag{2}$$

Combining Equations 1 and 2 we know that

$$\begin{aligned} \mathbf{Adv}^{\$\,\$}_{\mathrm{EMD}[\mathrm{Perm}(n),m]} &= \Pr[\, A^{\mathbb{E}_\pi\,\mathbb{D}_\pi} \Rightarrow 1\,] - \Pr[\, A^{\$\,\$} \Rightarrow 1\,] \\ &= \Pr[\, A^{\mathrm{EMD1}} \Rightarrow 1\,] - \Pr[\, A^{\mathrm{RND1}} \Rightarrow 1\,] \end{aligned} \tag{3}$$

```
To respond to oracle query Enc(T, P_1 ··· P_m):
100   for i ← 1 to m do C_i ←$ {0,1}^n
101   return C_1 ··· C_m

To respond to oracle query Dec(T, C_1 ··· C_m):
200   for i ← 1 to m do P_i ←$ {0,1}^n
201   return P_1 ··· P_m
```

Figure 5: Game RND1 realizes the defining experiment for a $ $ oracle for an nm-bit block cipher.

Game EMD2 We now make some changes in the way that game EMD1 is played. These changes don't effect anything an adversary can see. Such changes are said to give an *adversarially indistinguishable* game—the new game and the old one provide to any adversary an identical distribution on views. The new game, denoted EMD2, is shown in Figure 6. Rather than choosing the random permutation $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ up front, we fill in its values as needed. 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 unassigned range values. When we need $\pi^{-1}(y)$ and there is no x for which $\pi(x)$ has been assigned y we likewise choose x at random from the unassigned domain values. As we fill in π its domain and its range thus grows. At a given point in time we let $\mathrm{Domain}(\pi) = \{x \in \{0,1\}^n : \pi(i) \neq \mathsf{undef}\}$ denote the current domain and we let $\mathrm{Range}(\pi) = \{y \in \{0,1\}^n : \pi(x) = y \text{ for some } x \in \{0,1\}^n\}$ denote the current range. We let $\overline{\mathrm{Domain}}(\pi)$ and $\overline{\mathrm{Range}}(\pi)$ be the complement of these sets relative to $\{0,1\}^n$.

Actually, instead of directly sampling from $\overline{\mathrm{Range}}(\pi)$ we sample y from $\{0,1\}^n$ and then *re-sample*, this time from $\overline{\mathrm{Range}}(\pi)$, if the initially chosen sample y was already in the range of π. We behave analogously when we sample from $\overline{\mathrm{Domain}}(\pi)$. Whenever we are forced to re-sample we set a flag *bad*. The flag *bad* is never seen by the adversary A that interacts with the EMD2 oracle; it is only present to facilitate the subsequent analysis. We also set *bad* under some additional circumstances (lines 114, 133, 214, and 233), as shown in the game.

As we run Game EMD2 we maintain sets $\mathcal{P}$ and $\mathcal{C}$ for the plaintext sectors and ciphertext sectors that have already been asked of Enc and Dec, respectively. Using these sets we know that certain values of π have already been filled in.

Game EMD2 is adversarially indistinguishable from game EMD1. We therefore have that

$$\Pr[\,A^{\mathrm{EMD1}} \Rightarrow 1\,] \quad = \quad \Pr[\,A^{\mathrm{EMD2}} \Rightarrow 1\,] \tag{4}$$

Game EMD3 We now make a small, adversarially-invisible, change to game EMD2. Looking at line 131 of game EMD2, note that we first choose CC_i at random and then define C_i from it, according to $C_i \leftarrow CC_i \oplus CCC_{i-1}$. It is equivalent to choose C_i at random and then define CC_i from it, according to $CC_i \leftarrow C_i \oplus CCC_{i-1}$. The analogous comments apply to line 231 of game EMD2; we could just as well have chosen P_i at random and defined PP_i using it. Thus, in game EMD3, we make this and only this change, modifying only lines 131 and 231. An adversary given EMD2 or EMD3 is provided an identical view and so, in particular,

$$\Pr[\,A^{\mathrm{EMD2}} \Rightarrow 1\,] \quad = \quad \Pr[\,A^{\mathrm{EMD3}} \Rightarrow 1\,] \tag{5}$$

Initialization:

$bad \leftarrow \mathsf{false}$; $\mathcal{P} \leftarrow \mathcal{C} \leftarrow \emptyset$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$

To respond to an oracle query $\mathsf{Enc}(T,\ P_1 \cdots P_m)$:

Let u be the largest value in $[0 \mathinner{..} m]$ s.t. $P_1 \cdots P_u$ is a prefix of a string in $\mathcal{P}$
$PPP_0 \leftarrow CCC_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $PP_i \leftarrow P_i \oplus PPP_{i-1}$, $PPP_i \leftarrow \pi(PP_i)$
for $i \leftarrow u+1$ **to** m **do**
    $PPP_i \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$
    $PP_i \leftarrow P_i \oplus PPP_{i-1}$; **if** $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \leftarrow \pi(PP_i)$
    $\pi(PP_i) \leftarrow PPP_i$
$Mask \leftarrow \mathsf{multx}\,(PPP_1 \oplus CCC_1) \oplus T$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i \leftarrow PPP_{m+1-i} \oplus Mask$
for $i \leftarrow 1$ **to** m **do**
    $CC_i \xleftarrow{\$} \{0,1\}^n$; $C_i \leftarrow CC_i \oplus CCC_{i-1}$
    if $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CC_i \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$, $C_i \leftarrow CC_i \oplus CCC_{i-1}$
    if $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CC_i \leftarrow \pi(CCC_i)$, $C_i \leftarrow CC_i \oplus CCC_{i-1}$
    $\pi(CCC_i) \leftarrow CC_i$
$\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$
return $C_1 \cdots C_m$

To respond to an oracle query $\mathsf{Dec}(T,\ C_1 \cdots C_m)$:

Let u be the largest value in $[0 \mathinner{..} m]$ s.t. $C_1 \cdots C_u$ is a prefix of a string in $\mathcal{C}$
$CCC_0 \leftarrow PPP_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $CC_i \leftarrow C_i \oplus CCC_{i-1}$, $CCC_i \leftarrow \pi^{-1}(CC_i)$
for $i \leftarrow u+1$ **to** m **do**
    $CCC_i \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CCC_i \xleftarrow{\$} \overline{\mathrm{Domain}}(\pi)$
    $CC_i \leftarrow C_i \oplus CCC_{i-1}$; **if** $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \leftarrow \pi^{-1}(CC_i)$
    $\pi(CCC_i) \leftarrow CC_i$
$Mask \leftarrow \mathsf{multx}\,(CCC_1 \oplus CCC_m) \oplus T$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i \leftarrow CCC_{m+1-i} \oplus Mask$
for $i \leftarrow 1$ **to** m **do**
    $PP_i \xleftarrow{\$} \{0,1\}^n$; $P_i \leftarrow PP_i \oplus PPP_{i-1}$
    if $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PP_i \xleftarrow{\$} \overline{\mathrm{Domain}}(\pi)$, $P_i \leftarrow PP_i \oplus PPP_{i-1}$
    if $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PP_i \leftarrow \pi(PPP_i)$, $P_i \leftarrow PP_i \oplus PPP_{i-1}$
    $\pi(PP_i) \leftarrow PPP_i$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$; $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$
return $P_1 \cdots P_m$

Figure 6: Game EMD2 is adversarially indistinguishable from Game EMD1 but works a little differently, filling in π as needed.

Initialization:

000 $bad \leftarrow \mathsf{false}$; $\mathcal{P} \leftarrow \mathcal{C} \leftarrow \emptyset$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$

To respond to an oracle query $\mathsf{Enc}(T,\ P_1 \cdots P_m)$:

110 Let u be the largest value in $[0 \mathinner{..} m]$ s.t. $P_1 \cdots P_u$ is a prefix of a string in $\mathcal{P}$
111 $PPP_0 \leftarrow CCC_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $PP_i \leftarrow P_i \oplus PPP_{i-1}$, $PPP_i \leftarrow \pi(PP_i)$
112 **for** $i \leftarrow u+1$ **to** m **do**
113 $\quad PPP_i \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$
114 $\quad PP_i \leftarrow P_i \oplus PPP_{i-1}$; **if** $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \leftarrow \pi(PP_i)$
115 $\quad \pi(PP_i) \leftarrow PPP_i$
120 $Mask \leftarrow \mathsf{multx}\,(PPP_1 \oplus PPP_m) \oplus T$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i \leftarrow PPP_{m+1-i} \oplus Mask$
130 **for** $i \leftarrow 1$ **to** m **do**
131 $\quad C_i \xleftarrow{\$} \{0,1\}^n$; $CC_i \leftarrow C_i \oplus CCC_{i-1}$
132 $\quad$ **if** $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CC_i \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$, $C_i \leftarrow CC_i \oplus CCC_{i-1}$
133 $\quad$ **if** $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CC_i \leftarrow \pi(CCC_i)$, $C_i \leftarrow CC_i \oplus CCC_{i-1}$
134 $\quad \pi(CCC_i) \leftarrow CC_i$
140 $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$
141 **return** $C_1 \cdots C_m$

To respond to an oracle query $\mathsf{Dec}(T,\ C_1 \cdots C_m)$:

210 Let u be the largest value in $[0 \mathinner{..} m]$ s.t. $C_1 \cdots C_u$ is a prefix of a string in $\mathcal{C}$
211 $CCC_0 \leftarrow PPP_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $CC_i \leftarrow C_i \oplus CCC_{i-1}$, $CCC_i \leftarrow \pi^{-1}(CC_i)$
212 **for** $i \leftarrow u+1$ **to** m **do**
213 $\quad CCC_i \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $CCC_i \xleftarrow{\$} \overline{\mathrm{Domain}}(\pi)$
214 $\quad CC_i \leftarrow C_i \oplus CCC_{i-1}$; **if** $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PPP_i \leftarrow \pi^{-1}(CC_i)$
215 $\quad \pi(CCC_i) \leftarrow CC_i$
220 $Mask \leftarrow \mathsf{multx}\,(CCC_1 \oplus CCC_m) \oplus T$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i \leftarrow CCC_{m+1-i} \oplus Mask$
230 **for** $i \leftarrow 1$ **to** m **do**
231 $\quad P_i \xleftarrow{\$} \{0,1\}^n$; $PP_i \leftarrow P_i \oplus PPP_{i-1}$
232 $\quad$ **if** $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PP_i \xleftarrow{\$} \overline{\mathrm{Domain}}(\pi)$, $P_i \leftarrow PP_i \oplus PPP_{i-1}$
233 $\quad$ **if** $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$, $PP_i \leftarrow \pi(PPP_i)$, $P_i \leftarrow PP_i \oplus PPP_{i-1}$
234 $\quad \pi(PP_i) \leftarrow PPP_i$
240 $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$; $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$
241 **return** $P_1 \cdots P_m$

Figure 7: Game EMD3 is adversarially indistinguishable from Game EMD2 but works a little differently, choosing random return values in lines 131 and 231 instead of choosing random values to assign to π.

Initialization:

$bad \leftarrow \mathsf{false}$; $\mathcal{P} \leftarrow \mathcal{C} \leftarrow \emptyset$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$

To respond to an oracle query $\mathsf{Enc}(T,\ P_1 \cdots P_m)$:

Let u be the largest value in $[0\,..\,m]$ s.t. $P_1 \cdots P_u$ is a prefix of a string in $\mathcal{P}$
$PPP_0 \leftarrow CCC_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $PP_i \leftarrow P_i \oplus PPP_{i-1}$, $PPP_i \leftarrow \pi(PP_i)$
for $i \leftarrow u+1$ **to** m **do**
  $PPP_i \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $PP_i \leftarrow P_i \oplus PPP_{i-1}$; **if** $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(PP_i) \leftarrow PPP_i$

$Mask \leftarrow \mathsf{multx}\,(PPP_1 \oplus PPP_m) \oplus T$; **for** $i \in [1\,..\,m]$ **do** $CCC_i \leftarrow PPP_{m+1-i} \oplus Mask$

for $i \leftarrow 1$ **to** m **do**
  $C_i \xleftarrow{\$} \{0,1\}^n$; $CC_i \leftarrow C_i \oplus CCC_{i-1}$
  if $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(CCC_i) \leftarrow CC_i$

$\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$

return $C_1 \cdots C_m$

To respond to an oracle query $\mathsf{Dec}(T,\ C_1 \cdots C_m)$:

Let u be the largest value in $[0\,..\,m]$ s.t. $C_1 \cdots C_u$ is a prefix of a string in $\mathcal{C}$
$CCC_0 \leftarrow PPP_0 \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** u **do** $CC_i \leftarrow C_i \oplus CCC_{i-1}$, $CCC_i \leftarrow \pi^{-1}(CC_i)$
for $i \leftarrow u+1$ **to** m **do**
  $CCC_i \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $CC_i \leftarrow C_i \oplus CCC_{i-1}$; **if** $CC_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(CCC_i) \leftarrow CC_i$

$Mask \leftarrow \mathsf{multx}\,(CCC_1 \oplus CCC_m) \oplus T$; **for** $i \in [1\,..\,m]$ **do** $PPP_i \leftarrow CCC_{m+1-i} \oplus Mask$

for $i \leftarrow 1$ **to** m **do**
  $P_i \xleftarrow{\$} \{0,1\}^n$; $PP_i \leftarrow P_i \oplus PPP_{i-1}$
  if $PP_i \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  if $PPP_i \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
  $\pi(PP_i) \leftarrow PPP_i$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{C_1 \cdots C_m\}$; $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_1 \cdots P_m\}$
return $P_1 \cdots P_m$

Figure 8: Game RND2 is obtained from game EMD3 by dropping statements that immediately follow the setting of bad. This makes the game adversarially indistinguishable from game RND1.

Game RND2 We next modify game EMD3 by omitting the statement which immediately follow the setting of *bad* to true. (This is the usual trick under the game-substitution approach.) See Figure 8 for the definition of this new game, which we call RND2.

First note that in game RND2 we return, in response to any Enc-query, nm random bits, $C_1 \cdots C_m$. Similarly, we return, in response to any Dec-query, nm random bits, $P_1 \cdots P_m$. Thus RND2 provides an adversary with an identical view to RND1 and we know that

$$\Pr[\, A^{\text{RND1}} \Rightarrow 1\,] \;=\; \Pr[\, A^{\text{RND2}} \Rightarrow 1\,] \tag{6}$$

From a different angle, EMD3 and RND2 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

$$\Pr[\, A^{\text{EMD3}} \Rightarrow 1\,] - \Pr[\, A^{\text{RND2}} \Rightarrow 1\,] \;\leq\; \Pr[\, A^{\text{RND2}} \text{ sets } bad\,] \tag{7}$$

Combining Equations 3, 4, 5, 6, and 7, we thus have that

$$\begin{aligned}
\mathbf{Adv}^{\$\,\$}_{\text{EMD}[\text{Perm}(n),m]}(A) &= \Pr[\, A^{\text{EMD1}} \Rightarrow 1\,] - \Pr[\, A^{\text{RND1}} \Rightarrow 1\,] \\
&= \Pr[\, A^{\text{EMD3}} \Rightarrow 1\,] - \Pr[\, A^{\text{RND2}} \Rightarrow 1\,] \\
&\leq \Pr[\, A^{\text{RND2}} \text{ sets } bad\,] \qquad (8)
\end{aligned}$$

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

Game RND3 We now make a "cosmetic" change in game RND2 which will help set the notation for future accounting. The change is simply to tag each variable by the query number s associated to that variable. Clearly

$$\Pr[\, A^{\text{RND2}} \text{ sets } bad\,] \;=\; \Pr[\, A^{\text{RND3}} \text{ sets } bad\,] \tag{9}$$

Game RND4 Next we reorganize game RND3 so as to separate out the random values that are returned to the adversary. We already remarked, when showing that games RND2 and RND1 were adversarially indistinguishable, that game RND2 returned a nm-bit string in response to each adversary query. Of course this remains true in game RND3. Now, in game RND4, shown in Figure 10, we make that even more clear by choosing the necessary $C^s = C_1^s \cdots C_m^s$ or $P^s = P^1 \cdots P^m$ response just as soon as the s-th Enc or Dec query is made, respectively. Nothing else is done at that point except for recording if the adversary made an Enc query or a Dec query. Only when the adversary finishes all of its oracle queries and halts do we execute the "finalization" step of game RND4. That part of the game determines the value of flag *bad*. The procedure is designed to set *bad* under exactly the same conditions as in game RND4—indeed the game is identical to game RND3 except for the reordering of statements for which there is no dependency. The following is thus clear:

$$\Pr[\, A^{\text{RND3}} \text{ sets } bad\,] \;=\; \Pr[\, A^{\text{RND4}} \text{ sets } bad\,] \tag{10}$$

Initialization:

$bad \leftarrow \mathsf{false}$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$

To respond to the s-th oracle query $\mathsf{Enc}(T^s,\ P_1^s \cdots P_m^s)$:

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 $r \in [1 \mathinner{..} s{-}1]$
$PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $u[s]$ **do** $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow \pi(PP_i^s)$
for $i \leftarrow u[s]+1$ **to** m **do**
    $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$; **if** $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $\pi(PP_i^s) \leftarrow PPP_i^s$

$Mask^s \leftarrow \mathsf{multx}\,(PPP_1^s \oplus PPP_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus Mask^s$

for $i \leftarrow 1$ **to** m **do**
    $C_i^s \xleftarrow{\$} \{0,1\}^n$; $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$
    if $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    if $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $\pi(CCC_i^s) \leftarrow CC_i^s$

return $C_1^s \cdots C_m^s$

To respond to the s-th oracle query $\mathsf{Dec}(T^s,\ C_1^s \cdots C_m^s)$:

Let $u[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $C_1^r \cdots C_{u[s]}^r = C_1^s \cdots C_{u[s]}^s$ for some $r \in [1 \mathinner{..} s{-}1]$
$CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $u[s]$ **do** $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow \pi^{-1}(CC_i^s)$
for $i \leftarrow u[s]+1$ **to** m **do**
    $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$; **if** $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $\pi(CCC_i^s) \leftarrow CC_i^s$

$Mask^s \leftarrow \mathsf{multx}\,(CCC_1^s \oplus CCC_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus Mask^s$

for $i \leftarrow 1$ **to** m **do**
    $P_i^s \xleftarrow{\$} \{0,1\}^n$; $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$
    if $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    if $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
    $\pi(PP_i^s) \leftarrow PPP_i^s$

return $P_1^s \cdots P_m^s$

Figure 9: Game RND3, a notational change from game RND2, adds a superscript s to variables associated to the s-th query.

To respond to the s-th oracle query $\mathsf{Enc}(T^s,\ P_1^s \cdots P_m^s)$:

$ty^s \leftarrow \mathsf{Enc}$
$C_1^s \cdots C_m^s \xleftarrow{\$} \{0,1\}^{nm}$
return $C_1^s \cdots C_m^s$

To respond to the s-th oracle query $\mathsf{Dec}(T^s,\ C_1^s \cdots C_m^s)$:

$ty^s \leftarrow \mathsf{Dec}$
$P_1^s \cdots P_m^s \xleftarrow{\$} \{0,1\}^{nm}$
return $P_1^s \cdots P_m^s$

Finalization:

$bad \leftarrow \mathsf{false}$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$
for $s \leftarrow 1$ **to** q **do**

  if $ty^s = \mathsf{Enc}$ **then**
    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 $r \in [1 \mathinner{..} s-1]$
    $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $u[s]$ **do** $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow \pi(PP_i^s)$
    for $i \leftarrow u[s]+1$ **to** m **do**
      $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$; **if** $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(PP_i^s) \leftarrow PPP_i^s$
    $Mask^s \leftarrow \mathsf{multx}\,(PPP_1^s \oplus PPP_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$
      if $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      if $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(CCC_i^s) \leftarrow CC_i^s$
  else $(ty^s = \mathsf{Dec})$
    Let $u[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $C_1^r \cdots C_m^r = C_1^s \cdots C_{u[s]}^s$ for some $r \in [1 \mathinner{..} s-1]$
    $CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $u[s]$ **do** $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow \pi^{-1}(CC_i^s)$
    for $i \leftarrow u[s]+1$ **to** m **do**
      $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $CC_i^s \leftarrow C_i^s \oplus CCC_{i-1}^s$; **if** $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(CCC_i^s) \leftarrow CC_i^s$
    $Mask^s \leftarrow \mathsf{multx}\,(CCC_1^s \oplus CCC_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $PP_i^s \leftarrow P_i^s \oplus PPP_{i-1}^s$
      if $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      if $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(PP_i^s) \leftarrow PPP_i^s$

Figure 10: Game RND4 is adversarially indistinguishable from game RND3 but defers the setting of bad until all queries have been asked and answered.

$bad \leftarrow \mathsf{false}$; **for** $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow \mathsf{undef}$
for $s \leftarrow 1$ **to** q **do**

  if $\mathsf{ty}_s = \mathsf{Enc}$ **then**
    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 $r \in [1 \mathinner{..} s{-}1]$
    $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow \pi(PP_i^s)$
    for $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
      $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$; **if** $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(PP_i^s) \leftarrow PPP_i^s$
    $Mask^s \leftarrow \mathsf{multx}\,(PPP_1^s \oplus PPP_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i^m \leftarrow PPP_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$
      if $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      if $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(CCC_i^s) \leftarrow CC_i^s$

  else ($\mathsf{ty}_s = \mathsf{Dec}$)
    Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{C}_1^r \cdots \mathsf{C}_m^r = \mathsf{C}_1^s \cdots \mathsf{C}_{\mathsf{u}[s]}^s$ for some $r \in [1 \mathinner{..} s{-}1]$
    $CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow \pi^{-1}(CC_i^s)$
    for $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
      $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $CCC_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$; **if** $CC_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(CCC_i^s) \leftarrow CC_i^s$
    $Mask^s \leftarrow \mathsf{multx}\,(CCC_1^s \oplus CCC_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$
      if $PP_i^s \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      if $PPP_i^s \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow \mathsf{true}$
      $\pi(PP_i^s) \leftarrow PPP_i^s$

Figure 11: Game NON1 is based on game RND4 but now $(\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$ are fixed, valid constants, where $\mathbf{ty} = (\mathsf{ty}^1, \ldots, \mathsf{ty}^q)$ and $\mathbf{T} = (\mathsf{T}^1, \ldots, \mathsf{T}^q)$ and $\mathbf{C} = (\mathsf{C}^1, \ldots, \mathsf{C}^q)$ and $\mathbf{P} = (\mathsf{P}^1, \ldots, \mathsf{P}^q)$ and $\mathsf{C}^s = \mathsf{C}_1^s \cdots \mathsf{C}_m^s$ and $\mathsf{P}^s = \mathsf{P}_1^s \cdots \mathsf{P}_m^s$. There is no longer any adversary—it has been absorbed by universal quantification over $(\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$.

Game NON1 So far we have not changed the structure of the games at all: it has remained an adversary asking q questions to an oracle, our 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 RND4. We may assume that the adversary is deterministic, and so the probability is over the random choices made at lines 011, 021, 113, and 213. We will now eliminate the coins associated to lines 011 and 021. Recall that the adversary asks no purposeless queries.

We would like to make the stronger statement that for *any* set of values that might be returned by the adversary at lines 011 and 021, and for *any* set of queries (none purposeless) associated to them, the Finalization step of game RND4 rarely sets *bad*. However, this statement isn't quite true. Notice, in particular, that if the r-th and s-th queries ($r < s$) are Enc queries that return strings beginning C_1^r and C_1^s where $C_1^r = C_1^s$, then the flag *bad* will get set at line 132 when we process the C_1^s. Similarly, if the r-th and s-th queries ($r < s$) are Dec queries that return strings beginning P_1^r and P_1^s where $P_1^r = P_1^s$, then the flag *bad* will get set at line 232 when we process the P_1^s. We call such collisions *immediate* collisions. Clearly the probability of an immediate collision is at most $(1+2+\cdots+(q-1))/2^{n+1} = q(q-1)/2^{n+1}$.

We make from the Finalization part of game RND4 a new game, game NON1 (for "noninteractive"). This game silently depends on constants $\mathbf{ty} = (\mathsf{ty}^1, \cdots, \mathsf{ty}^q)$, $\mathbf{T} = (\mathsf{T}^1, \cdots, \mathsf{T}^q)$, and $\mathbf{P} = (\mathsf{P}^1, \cdots, \mathsf{P}^q)$, and $\mathbf{C} = (\mathsf{C}^1, \cdots, \mathsf{C}^q)$ where $\mathsf{ty}^s \in \{\mathsf{Enc}, \mathsf{Dec}\}$, $\mathsf{T}^s \in \{0,1\}^n$, $\mathsf{P}^s = \mathsf{P}_1^s \cdots \mathsf{P}_m^s$, and $\mathsf{C}^s = \mathsf{C}_1^s \cdots \mathsf{C}_m^s$, for $|\mathsf{P}_i^r| = |\mathsf{C}_i^r| = n$. Constants $(\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$ may not specify any immediate collisions or purposeless query; we call such a set of constants *valid*. Saying that $(\mathbf{ty}, \mathbf{P}, \mathbf{C})$ is valid means, with the notation above, that if $\mathsf{ty}^s = \mathsf{Enc}$ then $\mathsf{P}^s \neq \mathsf{P}^r$ for any $r < s$, and if $\mathsf{ty}^s = \mathsf{Dec}$ then $\mathsf{C}^s \neq \mathsf{C}^r$ for any $r < s$. Now fix a *worst* set of valid constants $(\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$, meaning one for which $\Pr[\,\text{NON1 sets } bad\,]$ is maximized. Then we have that

$$\Pr[\,A^{\text{RND4}} \text{ sets } bad\,] \;\leq\; \Pr[\,\text{NON1 sets } bad\,] + 0.5\, q(q-1)/2^n \tag{11}$$

Game NON2 We now re-write game NON1 so as to eliminate the variable π. Notice that in game NON1 the variable π was used in a quite restricted way: apart from lines 111 and 211, which are easily re-coded without use of π, we didn't actually use π to keep track of the association between domain points and range points; all we were really using π for was to keep track of which points are in its domain and which points were in its range. We could just as well have kept that information as two multisets, $\mathcal{X}$ and $\mathcal{Y}$. In game NON2, shown in Figure 12, that is exactly what is done. We keep track of what was the growing domain and range of π in multisets $\mathcal{X}$ and $\mathcal{Y}$. Instead of setting $\pi(X) \leftarrow Y$ we put X into $\mathcal{X}$ and Y into $\mathcal{Y}$. Before we were always checking, just before the assignment to π, if it would cause a collision in its domain or range. Now we do the exact same check using $\mathcal{X}$ and $\mathcal{Y}$, but we defer it to game's end. Games NON1 and NON2 set *bad* under exactly the same condition, so

$$\Pr[\,A^{\text{NON1}} \text{ sets } bad\,] \;\leq\; \Pr[\,\text{NON2 sets } bad\,] \tag{12}$$

Game NON3 We now re-write game NON2 so as to eliminate the intermediate variable *Mask*. See Figure 13. Since games NON2 and NON3 are adversarially indistinguishable, We have that

$$\Pr[\,A^{\text{NON2}} \text{ sets } bad\,] \;\leq\; \Pr[\,\text{NON3 sets } bad\,] \tag{13}$$

$\mathcal{X} \leftarrow \mathcal{Y} \leftarrow \emptyset$ (multisets)
for $s \leftarrow 1$ **to** q **do**

  if $\mathsf{ty}_s = \mathsf{Enc}$ **then**
    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 $r \in [1 \mathinner{..} s{-}1]$
    $PPP_0^s \leftarrow CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow PPP_i^r$
    for $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
      $PPP_i^s \xleftarrow{\$} \{0,1\}^n$
      $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{PP_i^s\}$; $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{PPP_i^s\}$
    $Mask^s \leftarrow \mathsf{multx}\,(PPP_1^s \oplus PPP_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{CCC_i^s\}$; $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{CC_i^s\}$

  else ($ty_s = \mathsf{Dec}$)
    Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{C}_1^r \cdots \mathsf{C}_m^r = \mathsf{C}_1^s \cdots \mathsf{C}_{u[s]}^s$ for some $r \in [1 \mathinner{..} s{-}1]$
    $CCC_0^s \leftarrow PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow CCC_i^r$
    for $i \leftarrow \mathsf{u}[s]+1$ **to** m **do**
      $CCC_i^s \xleftarrow{\$} \{0,1\}^n$
      $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{CCC_i^s\}$; $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{CC_i^s\}$
    $Mask^s \leftarrow \mathsf{multx}\,(CCC_1^s \oplus CCC_m^s) \oplus T^s$; **for** $i \in [1 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_{m+1-i}^s \oplus Mask^s$
    for $i \leftarrow 1$ **to** m **do**
      $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{PP_i^s\}$; $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{PPP_i^s\}$

$bad \leftarrow$ (there is a collision in $\mathcal{X}$) **or** (there is a collision in $\mathcal{Y}$)

Figure 12: Game NON2 is like game NON1 but eliminates the function π, doing equivalent bookkeeping with the multisets $\mathcal{X}$ and $\mathcal{Y}$. This is the game that, finally, we can analyze.

$\mathcal{X} \leftarrow \mathcal{Y} \leftarrow \emptyset$ (multisets)
for $s \leftarrow 1$ **to** q **do**

  if $\mathsf{ty}_s = \mathsf{Enc}$ **then**
    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 $r \in [1 \mathinner{..} s{-}1]$
    $PPP_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus PPP_{i-1}^s$, $PPP_i^s \leftarrow PPP_i^r$

    $PPP_i^s \xleftarrow{\$} \{0,1\}^n$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
    $PP_i^s \leftarrow PPP_{i-1}^s \oplus \mathsf{P}_i^s$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{PP_i^s\}$, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{PPP_i^s\}$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$

    $CCC_i^s \leftarrow (PPP_1^s \oplus PPP_m^s) \cdot \mathsf{x} \oplus PPP_{m+1-i}^s \oplus \mathsf{T}^s$ for each $i \in [1 \mathinner{..} m]$
    $CC_1^s \leftarrow \mathsf{C}_1^s$
    $CC_i^s \leftarrow (PPP_1^s \oplus PPP_m^s) \cdot \mathsf{x} \oplus PPP_{m+2-i}^s \oplus \mathsf{T}^s \oplus \mathsf{C}_i^s$ for each $i \in [2 \mathinner{..} m]$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{CCC_i^s\}$, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{CC_i^s\}$ for each $i \in [1 \mathinner{..} m]$

  else ($ty_s = \mathsf{Dec}$)
    Let $\mathsf{u}[s]$ be the largest value in $[0 \mathinner{..} m]$ s.t. $\mathsf{C}_1^r \cdots \mathsf{C}_m^r = \mathsf{C}_1^s \cdots \mathsf{C}_{u[s]}^s$ for some $r \in [1 \mathinner{..} s{-}1]$
    $CCC_0^s \leftarrow 0^n$; **for** $i \leftarrow 1$ **to** $\mathsf{u}[s]$ **do** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus CCC_{i-1}^s$, $CCC_i^s \leftarrow CCC_i^r$

    $CCC_i^s \xleftarrow{\$} \{0,1\}^n$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
    $CC_i^s \leftarrow CCC_{i-1}^s \oplus \mathsf{C}_i^s$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{CCC_i^s\}$, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{CC_i^s\}$ for each $i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$

    $PPP_i^s \leftarrow (CCC_1^s \oplus CCC_m^s) \cdot \mathsf{x} \oplus CCC_{m+1-i}^s \oplus \mathsf{T}^s$ for each $i \in [1 \mathinner{..} m]$
    $PP_1^s \leftarrow \mathsf{P}_1^s$
    $PP_i^s \leftarrow (CCC_1^s \oplus CCC_m^s) \cdot \mathsf{x} \oplus CCC_{m+2-i}^s \oplus \mathsf{T}^s \oplus \mathsf{P}_i^s$ for each $i \in [2 \mathinner{..} m]$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{PP_i^s\}$; $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{PPP_i^s\}$ for each $i \in [1 \mathinner{..} m]$

$bad \leftarrow$ (there is a collision in $\mathcal{X}$) **or** (there is a collision in $\mathcal{Y}$)

Figure 13: Game NON3 is adversarially indistinguishable from game NON2 but, among other cosmetic changes, writes out the assignments to $\mathcal{X}$ and $\mathcal{Y}$ without the use of auxiliary variables $Mask$, $MaskP$, and $MaskC$.

A.3 Analysis of the Final Game

We now turn to the analysis of game NON3. We upper bound the probability of a collision in $\mathcal{X}$, and the probability of a collision in $\mathcal{Y}$.

Points are added to the multiset $\mathcal{X}$ at lines 112, 123, 212, and 223. Points augment the multiset $\mathcal{Y}$ at the same lines. We will show that for every pair of points X, X' that are added to $\mathcal{X}$, the probability that they collide is at most 2^{-n}. Since there are a total of $2mq$ points placed in $\mathcal{X}$ we get that the probability that there is some collision in $\mathcal{X}$ is at most $\binom{2mq}{2} \cdot 2^{-n} = 2mq(mq-1)/2^n$. The same statement holds for $\mathcal{Y}$. So from the sum bound the probability that *bad* gets set to true in game INT3 is at most $4mq(mq-1)/2^n$. Combining with the results of the previous two subsections we know that

$$\mathbf{Adv}^{\$\,\$}_{\mathrm{EMD}[\mathrm{Perm}(n),m]}(q) \quad \leq \quad 4mq(mq-1)/2^n + 0.5q(q-1)/2^n \tag{14}$$

and so

$$\begin{aligned} \mathbf{Adv}^{\mathrm{prp}}_{\mathrm{EMD}[\mathrm{Perm}(n),m]}(q) \quad &\leq \quad 4mq(mq-1)/2^n + 0.5q(q-1)/2^n + 2^{-mn} && (15) \\ &\leq \quad 5m^2q^2/2^n && (16) \end{aligned}$$

It remains to verify that collisions among points in the multiset $\mathcal{X}$ and $\mathcal{Y}$ occur with probability at most 2^{-n}.

In game NON3 the random strings chosen during the game's execution are all of the form PPP_i^s or CCC_i^s. However, not every PPP_i^s is random and not every CCC_i^s is random. The random strings are exactly the PPP_i^s where $i \in [u[s]+1 \mathinner{..} m]$ along with the CCC_i^s where $i \in [u[s]+1 \mathinner{..} m]$. The other PPP_i^s values are simply copies of PPP_i^r values for $r < s$ while the the other CCC_i^s values are copies of CCC_i^r values for $r < s$. For $i \in [1 \mathinner{..} m]$ let us write PPP_i^s to mean $(\mathsf{P\text{-}value}, i, r)$ where r is the smallest value in $[1 \mathinner{..} s]$ such that $\mathsf{P}_1^r \cdots \mathsf{P}_i^r = \mathsf{P}_1^s \cdots \mathsf{P}_i^s$. One can think of PPP_i^s as the formal symbol PPP_i^r whose value is copied into PPP_i^s. Define CCC_i^s similarly. We emphasize that PPP_i^r and CCC_i^r are constants associated to game NON3; they are not random variables.

By the structure of game NON3 every PPP_i^s and every CCC_i^s is a uniform random value. Furthermore, every PPP_i^s is independent of CCC_j^t and every PPP_i^s is independent of PPP_j^s when $i \neq j$ and PPP_i^s is independent of PPP_i^t when $\mathsf{PPP}_i^s \neq \mathsf{PPP}_i^t$ and CCC_i^s is independent of CCC_i^t when $\mathsf{CCC}_i^s \neq \mathsf{CCC}_i^t$.

Following game NON3, there are five kinds of points added to $\mathcal{X}$, as given in the following table.

$X1$	$PPP_{i-1}^s \oplus \mathsf{P}_i^s$	$i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
$X2$	$(PPP_1^s \oplus PPP_m^s) \cdot \mathsf{x} \oplus PPP_{m+1-i}^s \oplus \mathsf{T}^s$	$i \in [1 \mathinner{..} m]$
$X3$	CCC_i^s	$i \in [\mathsf{u}[s]{+}1 \mathinner{..} m]$
$X4$	P_1^s	
$X5$	$(CCC_1^s \oplus CCC_m^s) \cdot \mathsf{x} \oplus CCC_{m+2-i}^s \oplus \mathsf{T}^s \oplus \mathsf{P}_i^s$	$i \in [2 \mathinner{..} m]$

We must consider the possibility of a collision among two distinct points of any kind. This requires a case analysis, as follows.

Consider 1–1 collisions, meaning that $PPP_{i-1}^s \oplus \mathsf{P}_i^s = PPP_{j-1}^t \oplus \mathsf{P}_i^t$. Since these are two distinct points in the multiset, either $i \neq j$ or $s \neq t$. If $i \neq j$ then $\Pr[PPP_{i-1}^s \oplus \mathsf{P}_i^s = PPP_{j-1}^t \oplus \mathsf{P}_j^t] = 2^{-n}$ because $i-1 \neq j-1$ and differently subscripted PPP values are independent. Otherwise we are considering $\Pr[PPP_{i-1}^s \oplus \mathsf{P}_i^s = PPP_{i-1}^t \oplus \mathsf{P}_i^t]$. If $\mathsf{PPP}_{i-1}^s \neq \mathsf{PPP}_{i-1}^t$ then the probability in question is

2^{-n} because PPP^s_{i-1} and PPP^t_{i-1} are uniform and independent. Otherwise $PPP^s_{i-1} = PPP^t_{i-1}$ and the probability in question becomes $\Pr[\mathsf{P}^s_i = \mathsf{P}^t_i]$. But this probability is zero because $i \geq u[s]+1$. Namely, $\mathsf{PPP}^s_i = \mathsf{PPP}^t_i$ implies that $\mathsf{P}^s_1 \cdots \mathsf{P}^s_{i-1} = \mathsf{P}^t_1 \cdots \mathsf{P}^t_{i-1}$ and if, in addition, $\mathsf{P}^s_i = \mathsf{P}^t_i$ then $\mathsf{P}^s_1 \cdots \mathsf{P}^s_i = \mathsf{P}^t_1 \cdots \mathsf{P}^t_1$ and so $u[t] \geq i$, a contradiction.

Consider 1–2 collisions. This kind of collision can only occur with probability 2^{-n} because the random variable PPP^s_m in X2 is independent of the expression in X1.

Consider 1–3, 1–4, and 1–5 collisions. All of these occur with probability 2^{-n} because none of X3, X4, or X5 depend on PPP^s_{i-1}.

Consider 2–2 collisions, namely, a collision between $(PPP^s_1 \oplus PPP^s_m) \cdot \mathsf{x} \oplus PPP^s_{m+1-i} \oplus \mathsf{T}^s$ and $(PPP^t_1 \oplus PPP^t_m) \cdot \mathsf{x} \oplus PPP^t_{m+1-j} \oplus \mathsf{T}^t$. Here we have a number of cases to consider. If $\mathsf{PPP}^s_m \neq \mathsf{PPP}^t_m$ then the independence of PPP^s_m and $(PPP^t_1 \oplus PPP^t_m) \cdot \mathsf{x} \oplus PPP^t_{m+1-j} \oplus \mathsf{T}^t$. results in a collision probability to be 2^{-n}. If $\mathsf{PPP}^s_m = \mathsf{PPP}^t_m$ then the probability in question reduces to $\Pr[PPP^s_{m+1-i} \oplus \mathsf{T}^s = \Pr[PPP^t_{m+1-j} \oplus \mathsf{T}^t]$. Now if $i \neq j$ then this value is 2^{-n} by the independence of PPP^s_{m+1-i} and $\Pr[PPP^t_{m+1-j}$. And if $i = j$ then we must have that $\mathsf{T}^s \neq \mathsf{T}^t$ (by the validity of the constants associated to game NON3) and so the probability is 0.

Consider 2–3, 2–4, and 2–5 collisions. All of these occur with probability 2^{-n} by the presence of PPP^s_1, say, in the expression for X2.

Cases 3–3 and 3–4 are again obvious. For 3–5 collisions, if $\mathsf{CCC}^s_i \neq \mathsf{CCC}^t_1$ then use the randomness of the latter to get a collision bound of 2^{-n}. Otherwise, when $\mathsf{CCC}^s_i = \mathsf{CCC}^t_1$, use he randomness of CCC^s_m to get a collision bound of 2^{-n}.

Collisions of type 4–4 can not occur, by our assumption of validity, and collisions of type 4–5 clearly occur with probability 2^{-n}.

For collisions of type 5–5, argue exactly as with collisions of type 2–2.

This completes the argument that points from the multiset $\mathcal{X}$ collide with probability at most 2^{-n}. To prove that points in the multiset $\mathcal{Y}$ collide with probability at most 2^{-n}, first inspect the types of points which are placed in $\mathcal{Y}$.

$Y1$	$CCC^s_{i-1} \oplus \mathsf{C}^s_i$	$i \in [\mathsf{u}[s]{+}1 \,..\, m]$
$Y2$	$(CCC^s_1 \oplus CCC^s_m) \cdot \mathsf{x} \oplus CCC^s_{m+1-i} \oplus \mathsf{T}^s$	$i \in [1 \,..\, m]$
$Y3$	PPP^s_i	$i \in [\mathsf{u}[s]{+}1 \,..\, m]$
$Y4$	C^s_1	
$Y5$	$(PPP^s_1 \oplus PPP^s_m) \cdot \mathsf{x} \oplus PPP^s_{m+2-i} \oplus \mathsf{T}^s \oplus \mathsf{C}^s_i$	$i \in [2 \,..\, m]$

The points are identical to those that are in $\mathcal{X}$ except for the renaming of variables: one simply swaps the characters C and P, C and P. So the bound shown for collisions in $\mathcal{X}$ applies equally well for collisions in $\mathcal{Y}$. And this completes the proof of the theorem.

A.4 Comments

We have effectively double-counted type 4–4 collisions. By counting just slightly more carefully the bound is reduced to $4m^2q^2/2^n$.

The "core" of the proof, Section A.3, is case analysis that depends strongly on the details of the scheme. The lengthy game-substitution portion, Section A.2, seems comparatively independent of the details of the algorithm. Still, we know of know way to eliminate the argument and retain rigor. Perhaps the approach of Maurer [13] might be useful.