A Parallelizable Enciphering Mode

Shai Halevi[*] Phillip Rogaway[†]

July 28, 2003

Abstract

We describe a block-cipher mode of operation, EME, that turns an n-bit block cipher into a tweakable enciphering scheme that acts on strings of mn bits, where $m \in [1..n]$. The mode is *parallelizable*, but as serial-efficient as the non-parallelizable mode CMC [6]. EME can be used to solve the disk-sector encryption problem. The algorithm entails two layers of ECB encryption and a "lightweight mixing" in between. We prove EME secure, in the reduction-based sense of modern cryptography. We motivate some of the design choices in EME by showing that a few simple modifications of this mode are insecure.

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 **1**

2 Preliminaries **2**

3 Specification of EME **3**

4 Security of EME **3**

5 Proof Ideas **5**

6 Some Insecure Modifications **6**

6.1 The "extra" block-cipher call is needed 6

6.2 Necessity of the Length Restriction 7

References **8**

A Extending EME to Longer Messages **9**

B Proof of Theorem 1 — Security of EME **11**

B.1 The game-substitution sequence 11

B.2 Analysis of the non-interactive game 17

1 Introduction

Tweakable enciphering schemes and their use. A *tweakable enciphering scheme* is a function $\mathbf{E}$ that maps a plaintext P into a ciphertext $C = \mathbf{E}_K^T(P)$ under the control of a key K and tweak T. The ciphertext must have the same length as the plaintext and there must be an inverse $\mathbf{D}_K^T$ to $\mathbf{E}_K^T$. We are interested in schemes that are secure in the sense of a tweakable, strong pseudorandom-permutation ($\pm\widetilde{\mathrm{prp}}$): an oracle that maps (T, P) into $\mathbf{E}_K^T(P)$ and maps (T, C) into $\mathbf{D}_K^T(C)$ must be indistinguishable (when the key K is random and secret) from an oracle that realizes a T-indexed family of random permutations and their inverses. A tweakable enciphering scheme that is secure in the $\pm\widetilde{\mathrm{prp}}$-sense makes a desirable tool for solving the disk-sector encryption problem: one stores at disk-sector location T the ciphertext $C = \mathbf{E}_K^T(P)$ for plaintext P. The IEEE Security in Storage Working Group [8] plans to standardize a $\pm\widetilde{\mathrm{prp}}$-secure enciphering scheme.

Our contribution. This paper specifies EME, which is a simple and parallelizable tweakable enciphering scheme. The scheme is built from a block cipher, such as AES. By making EME parallelizable we accommodate ultra-high-speed mass-storage devices to the maximal extent possible given our security goals. When based on a block cipher $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ our mode uses a k-bit key and $2m+1$ block-cipher calls to encipher an mn-bit plaintext in a way that depends on an n-bit tweak. We require that $m \in [1..n]$.

The name EME is meant to suggest ECB-Mix-ECB, as enciphering under EME involves ECB-encrypting the plaintext, a lightweight mixing step, and another ECB-encryption. For a description of EME look ahead to Figures 1 and 2.

We prove that EME is secure, assuming that the underling block cipher is secure. The proof is in the standard, provable-security tradition: an attack on EME (as a $\pm\widetilde{\mathrm{prp}}$ with domain $\mathcal{M} = \{0,1\}^n \cup \{0,1\}^{2n} \cup \cdots \cup \{0,1\}^{n^2}$) is shown to imply an attack on the underlying block cipher (as a strong PRP with domain $\{0,1\}^n$).

We go on to motivate some of the choices made in EME by showing that other choices would result in insecure schemes. Finally, we suggest an extension to EME that operates on sectors that are longer than mn bits.

CMC mode. The EME algorithm is follow-on work to the CMC method of Halevi and Rogaway [6]. Both modes are tweakable enciphering schemes built from a block cipher $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. But CMC is inherently sequential, as it is built around CBC encryption and decryption. EME was designed to overcome this limitation, which was seen as potentially problematic for high-speed encryption devices. The change does not increase the serial complexity; both modes use $2m + 1$ block-cipher calls (and little additional overhead) to act on an mn-bit messages.

Further history. Naor and Reingold gave an elegant approach for making a strong PRP on N bits from a block cipher on $n < N$ bits [14, 15]. Their approach involves a hashing step, a layer of ECB encryption (say), and another hashing step. They do not give a fully-specified mode, but they do show how to carry out the hashing step given an xor-universal hash-function that maps N bits to n bits [14]. In practice, instantiating this object is problematic: to compare well with CMC or EME one should find a construction that is simple and has a collision bound of about 2^{-128} and is more efficient, in both hardware and software, than AES. No such construction is known.

An early, unpublished version of the CMC paper contained buggy versions of the CMC and EME algorithms. Joux discovered the problem [9] and thereby played a key role in our arriving at a correct solution. CMC was easily fixed in response to Joux's attack, but EME did not admit a simple fix. (Indeed, Section 6.1 effectively proves that no "simple fix" is possible for the earlier buggy EME construction).

Efforts to construct a block cipher with a large blocksize from one with a smaller blocksize go back to Luby and Rackoff [13], who also put forward the notion of a PRP and a strong PRP. The concrete-security treatment of PRPs that we use begins with Bellare, Kilian, and Rogaway [2]. The notion of a tweakable block-cipher is due to Liskov, Rivest, and Wagner [12]. The first attempt to directly construct an mn-bit block cipher from an n-bit one is due to Zheng, Matsumoto, and Imai [17]. A different approach is to build a wide-blocksize block-cipher from scratch, as with BEAR, LION, and Mercy [1, 4].

Discussion. EME has some advantages over CMC beyond its parallelizability. First, it uses a single key for the underlying block cipher, instead of two keys. All block-cipher calls are keyed by this one key. Second, enciphering under EME uses only the forward direction of the block cipher, while deciphering now uses only the reverse direction. This is convenient when using a cipher such as AES, where the two directions are substantially different, as a piece of hardware or code might need only to encipher or only to decipher. Finally, we prove EME secure as a variable-input-length (VIL) cipher and not just as a fixed-input-length (FIL) one. This means that, in an attack, the adversary may intermix plaintexts and ciphertexts of various lengths.

We comment that the parallelizability goal is arguably of less utility for a $\pm\widetilde{\mathrm{prp}}$-secure enciphering scheme than for some other cryptographic goals. This is because, parallelizable or not, a $\pm\widetilde{\mathrm{prp}}$-secure encryption scheme cannot avoid having latency that grows with the length of the message being processed (to achieve the $\pm\widetilde{\mathrm{prp}}$ security notion one cannot output a single bit of ciphertext until the entire plaintext has been seen). Still, parallelizability is useful even here, and the user community wants it [7]. More broadly, EME continues a tradition of trying to make modes of operation (like CTR mode, IAPM/OCB [10, 16], and PMAC [3]) that achieve parallelizability at near-zero added computational cost compared to their intrinsically serial counterparts.

2 Preliminaries

Basics. We use the same notions and notation as in [6]. A *tweakable enciphering scheme* is a function $\mathbf{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} = \bigcup_{i \in I} \{0,1\}^i$ is the *message space* (for some nonempty index set $I \in \mathbb{N}$) and $\mathcal{K} \neq \emptyset$ is the *key space* and $\mathcal{T} \neq \emptyset$ is the *tweak space*. We require that for every $K \in \mathcal{K}$ and $T \in \mathcal{T}$ we have that $\mathbf{E}(K, T, \cdot) = \mathbf{E}_K^T(\cdot)$ is a length-preserving permutation on $\mathcal{M}$. The inverse of an enciphering scheme $\mathbf{E}$ is the enciphering scheme $\mathbf{D} = \mathbf{E}^{-1}$ where $X = \mathbf{D}_K^T(Y)$ if and only if $\mathbf{E}_K^T(X) = Y$. A *block cipher* is the special case of a tweakable enciphering scheme where the message space is $\mathcal{M} = \{0,1\}^n$ (for some $n \geq 1$) and the tweak space is $\mathcal{T} = \{\varepsilon\}$ (the empty string). The number n is called the *blocksize*.

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. The notation $A \Rightarrow 1$ describes the event that the adversary A outputs the bit one.

Security measure. For a tweakable enciphering scheme $\mathbf{E}\colon \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ we consider the advantage that the adversary A has in distinguishing $\mathbf{E}$ and its inverse from a random tweakable permutation and its inverse:

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

The notation show, 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 $X \xleftarrow{\$} \mathcal{X}$ we mean to choose X at random from the finite set $\mathcal{X}$. By $\mathrm{Perm}^{\mathcal{T}}(\mathcal{M})$ we mean the set of all functions $\pi\colon \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ where $\pi(T, \cdot)$ is a length-preserving permutation. By Perm(n) we mean all permutations on $\{0,1\}^n$. In writing $\pm\widetilde{\text{prp}}$ the tilde serves as a reminder that the PRP is tweakable and the $\pm$ symbol is a reminder that this is the "strong" (chosen plaintext/ciphertext attack) notion of security. For a block cipher, we omit the tilde.

Without loss of generality we assume that an adversary never repeats an encipher query, never repeats a decipher query, never queries its deciphering oracle with (T, C) if it got C in response to some (T, M) encipher query, and never queries its enciphering oracle with (T, M) if it earlier got M in response to some (T, C) decipher query. We call such queries *pointless* because the adversary "knows" the answer that it should receive.

When $\mathcal{R}$ is a list of resources and $\mathbf{Adv}_{\Pi}^{\text{xxx}}(A)$ has been defined, we write $\mathbf{Adv}_{\Pi}^{\text{xxx}}(\mathcal{R})$ for the maximal value of $\mathbf{Adv}_{\Pi}^{\text{xxx}}(A)$ over all adversaries A that use resources at most $\mathcal{R}$. Resources of interest are the running time t and the number of oracle queries q and the query complexity σ_n (where $n \geq 1$ is a number). The query complexity σ_n is measured as follows. A string X contributes $\max\{|X|/n,\ 1\}$ to the query complexity; a tuple of strings $(X_1, X_2, \ldots)$ contributes the sum of the contributions of each string; and the query complexity of an adversary is the sum of the contributions from all oracle queries plus the contribution from the adversary's output. So, for example, an adversary that asks oracle queries $(T_1, P_1) = (0^n, 0^{2n})$ and then $(T_2, P_2) = (0^n, \varepsilon)$ and then outputs a bit b has query complexity $3 + 2 + 1 = 6$. The name of an argument (e.g., t or σ_n) will be enough to make clear what resource it refers to.

FINITE FIELDS. We interchangeably view an n-bit string as: a string; a nonnegative integer less than 2^n (msb first); a formal polynomial over GF(2) (with the coefficient of x^{n-1} first and the free term last); and an abstract point in the finite filed GF(2^n). To do addition on field points, one xors their string representations. To do multiplication on field points, one must fix a degree-n irreducible polynomial. We choose to use the lexicographically first *primitive* polynomial of minimum weight. For $n = 128$ this is the polynomial $\mathsf{x}^{128}+\mathsf{x}^7+\mathsf{x}^2+\mathsf{x}+1$. See [5] for a list of the indicated polynomials. We note that with this choice of field-point representations, the point $\mathsf{x} = 0^{n-2}10 = 2$ will always have order $2^n - 1$ in the multiplicative subgroup of GF(2^n), meaning that $2, 2^2, 2^3, \ldots, 2^{2^n-1}$ are all distinct. Finally, we note that given $L = L_{n-1} \cdots L_1 L_0 \in \{0,1\}^n$ it is easy to compute $2L$. We illustrate the procedure for $n = 128$, in which case $2L = L{\ll}1$ if $\mathsf{firstbit}(L) = 0$, and $2L = (L{\ll}1) \oplus \text{Const87}$ if $\mathsf{firstbit}(L) = 1$. (Here $\text{Const87} = 0^{120}10^4 1^3$ and $\mathsf{firstbit}(L)$ means L_{n-1} and $L{\ll}1$ means $L_{n-2}L_{n-3} \cdots L_1 L_0 0$.)

3 Specification of EME

We construct from block cipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable enciphering scheme that we denote by EME[E] or EME-E. The enciphering scheme has key space $\mathcal{K}$, the same as the underlying cipher, and tweak space is $\mathcal{T} = \{0,1\}^n$. The message space $\mathcal{M} = \{0,1\}^n \cup \{0,1\}^{2n} \cup \cdots \{0,1\}^{n^2}$ contains any string having any number m of n-bit blocks, where $m \in [1..n]$. An illustration of EME mode is given in Figure 2. In the figures, all capitalized variables except for K are n-bit strings (key K is an element of the key-space $\mathcal{K}$). Variable names P and C are meant to suggest *plaintext* and *ciphertext*. When we write $\mathbf{E}_K^T(P_1 \cdots P_m)$ 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 when we write $\mathbf{D}_K^T(C_1 \cdots C_m)$ we mean that the incoming ciphertext $C = C_1 \cdots C_m$ is partitioned into n-bit strings $C_1, \ldots, C_m$. It is an error to provide $\mathbf{E}$ with a plaintext that is not mn bits for some $m \in [1 \,..\, n]$, or to supply $\mathbf{D}$ with a ciphertext that is not mn bits for some $m \in [1..n]$.

4 Security of EME

The following theorem relates the advantage an adversary can get in attacking EME[E] to the advantage that an adversary can get in attacking the block cipher E.

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

$L \leftarrow 2E_K(0^n)$
for $i \in [1 \,..\, m]$ **do**
  $PP_i \leftarrow 2^{i-1}\, L \oplus P_i$
  $PPP_i \leftarrow E_K(PP_i)$

$SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
$MP \leftarrow PPP_1 \oplus SP \oplus T$
$MC \leftarrow E_K(MP)$
$M \leftarrow MP \oplus MC$
for $i \in [2 \,..\, m]$ **do** $CCC_i \leftarrow PPP_i \oplus 2^{i-1} M$
$SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
$CCC_1 \leftarrow MC \oplus SC \oplus T$

for $i \in [1 \,..\, m]$ **do**
  $CC_i \leftarrow E_K(CCC_i)$
  $C_i \leftarrow CC_i \oplus 2^{i-1}\, L$

return $C_1 \cdots C_m$

Algorithm $\mathbf{D}_{Kt}^T(C_1 \cdots C_m)$

$L \leftarrow 2E_K(0^n)$
for $i \in [1 \,..\, m]$ **do**
  $CC_i \leftarrow 2^{i-1}\, L \oplus C_i$
  $CCC_i \leftarrow E_K^{-1}(CC_i)$

$SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
$MC \leftarrow CCC_1 \oplus SC \oplus T$
$MP \leftarrow E_K^{-1}(MC)$
$M \leftarrow MC \oplus MP$
for $i \in [2 \,..\, m]$ **do** $PPP_i \leftarrow CCC_i \oplus 2^{i-1} M$
$SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
$PPP_1 \leftarrow MP \oplus SP \oplus T$

for $i \in [1 \,..\, m]$ **do**
  $PP_i \leftarrow E_K^{-1}(PPP_i)$
  $P_i \leftarrow PP_i \oplus 2^{i-1}\, L$

return $P_1 \cdots P_m$

Figure 1: Enciphering (left) and deciphering (right) under $\mathbf{E} = \mathrm{EME}[E]$, where $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ 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 $C = C_1 \cdots C_m$.

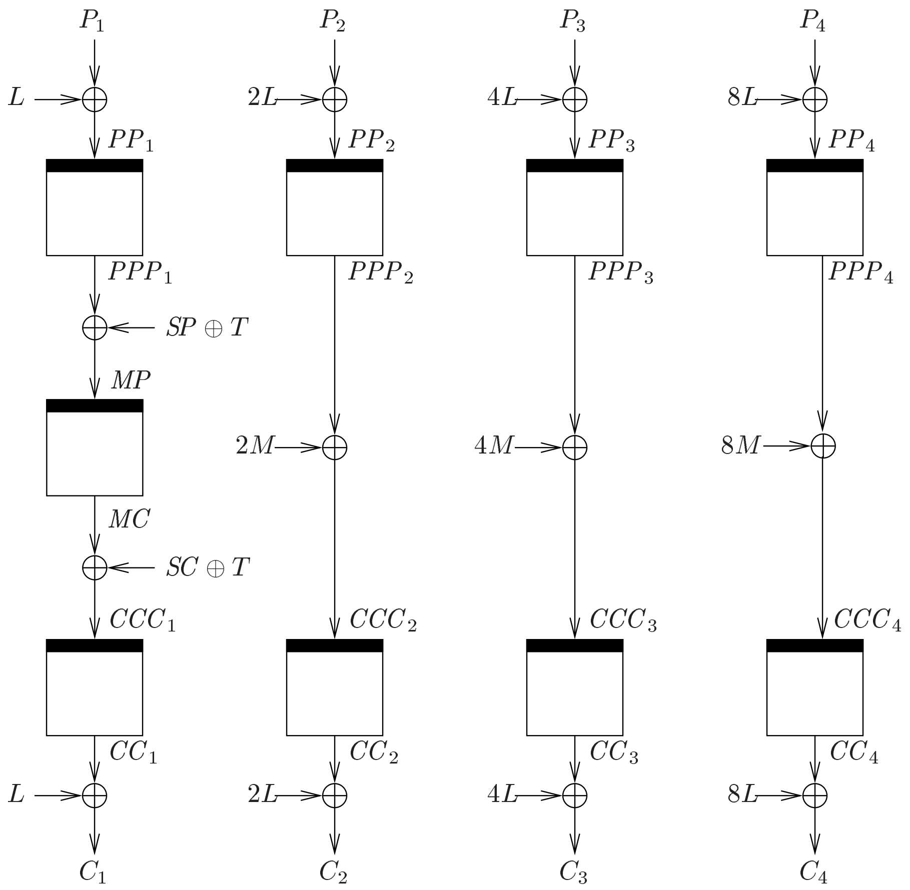

Figure 2: Enciphering a four-block message $P_1P_2P_3P_4$ under EME. The boxes represent E_K and $L = 2E_K(0^n)$. We set $SP = PPP_2 \oplus PPP_3 \oplus PPP_4$ and $M = MP \oplus MC$ and $SC = CCC_2 \oplus CCC_3 \oplus CCC_4$.

Theorem 1 [EME security] Fix $n, t, \sigma_n \in \mathbb{N}$ and a block cipher $E: \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(\sigma_n) \leq \frac{7\,\sigma_n^2}{2^n} \quad \text{and} \tag{1}$$

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

where $t' = t + O(\sigma_n)$. □

The heart of Theorem 1 is Equation (1), which is given in Appendix B. Equation (2) embodies the standard way to pass from the information-theoretic setting to the complexity-theoretic one.

5 Proof Ideas

Since the proof in Appendix B is quite long we give a brief sketch here of some of its ideas. We consider an attack against EME as a game between the attacker and the mode itself, where the cipher is replaced by a truly random permutation and this permutation is chosen "on the fly" during this game. We give names to all of the internal blocks that occur in the game, where an internal block is any of the n-bit values PP_i, PPP_i, MP, MC, CCC_i, CC_i that arise as the game is played. For example, PPP_i^s is the PPP_i-block of the s^{th} query of the attacker.

As usual with such modes, the core of the proof is to show that "accidental collisions" are unlikely. An accidental collision is an equality between two internal blocks which is not obviously guaranteed due to the structure of the mode. Specifically, an equality between the i^{th} blocks in two different encipher queries $P_i^s = P_i^t$ implies that we also have the equalities $PP_i^s = PP_i^t$ and $PPP_i^s = PPP_i^t$ and so these do not count as collisions. (And likewise for decipher queries.) Most other collisions are considered accidental collisions and we show that those rarely happen.[1] Showing that accidental collisions are rare is ultimately done by case analysis (but, as usual, it takes a non-trivial argument to get there). For example, in one case we show that with high probability $PP_i^s \neq PP_j^t$; in another case we show that with high probability $PPP_i^s \neq MC^t$.

The analysis of most of the cases is standard. Below we illustrate one of the more interesting cases. We show that for an encipher query P^s the block MP^s does not collide with any of the previous MP^r blocks (cf. Claim 7 in Appendix B). This is easily seen if any of the plaintext blocks P_i^s is a "new blocks" (i.e., different than P_i^r for all $r < s$). But we need to show it also for the case where the plaintext P^s was obtained by "mix-and-matching" blocks from previous plaintext vectors. So let $r < s$ be the last plaintext that share some blocks with P^s, i.e., $P_i^r = P_i^s$ for some index i. This means that all the blocks P_i^s appeared in queries no later than r If queries s and r sport the same plaintext vectors, $P^r = P^s$, and differ only in the tweak values, $T^r \neq T^s$, then we clearly have $MP^r \oplus MP^s = T^r \oplus T^s \neq 0$. So assume that $P^r \neq P^s$, let Eq be the set of indexes where they are equal, and denote $D_r = \{1..m^r\} - Eq$ and $D_s = \{1..m^s\} - Eq$, where m^s and m^r are the lengths (in blocks) of queries r and s. That is, $P_i^r = P_i^s$ exactly for all $i \in Eq$, which means that all the blocks P_i^s for $i \in D_s$ appeared in queries before query r. This, in turn, implies that the value of PPP_i^s for any $i \in D_s$ depends only on things that were determined before query r.

Assume that query r was decipher (and that MC^r did not already accidentally collide with anything), so MP^r was chosen "almost at random" during the processing of query r. We show that the sum $MP^s \oplus MP^r$ can be expressed as $aMP^r + \beta$, where $a \neq 0$ is a constant and β is some

[1] Actually, we only care about collisions between two values in the domain of π or between two values in its range; collisions between a domain value and a range value, such as $PP_i^s = CC_i^r$, are inconsequential and we ignore those.

expression that only depends on things that were determined before the choice of MP^r. Thus, the sum $MP^s \oplus MP^r$ is rarely zero. We can write this sum as

$$\begin{aligned} MP^s \oplus MP^r &= T^s \oplus \sum_{i=1}^{m^s} PPP_i^s \;\oplus T^r \oplus \sum_{i=1}^{m^r} PPP_i^r \;=\; T^r \oplus T^s \oplus \sum_{i\in D_s} PPP_i^s \;\oplus \sum_{i\in D_r} PPP_i^r \\ &= \text{things-that-were-determined-before-query-}r \;\oplus \sum_{i\in D_r} PPP_i^r \end{aligned}$$

Assuming that D_r is non-empty, it is sufficient to show that we can express $\sum_{i\in D_r} PPP_i^r = aMP^r + \beta$ where a is non-zero and β only depends on things that were determined before the choice of MP^r (cf. Claim 2 in Appendix B). There are two cases in this proof, depending on whether $1 \in D_r$ or not, but they both boil down to the same point: since we use the value $2^{i-1}(MC^s \oplus MP^s)$ to mask the CCC_i block, the sum of PPP_i^r's can be written as

$$\sum_{i\in D_r} PPP_i^r \;=\; \text{some-expression-in-the-}CCC_i^r\text{'s-and-}MC^r \;\oplus\; \left(\sum_{i\in D'} 2^{i-1}\right) MP^r$$

where D' is also a non-empty set, $D' \subseteq [1..m^r]$, and so the coefficient of MP^r in this expression is non-zero. The case where query r is encipher is a bit longer, but it uses similar observations.

One last "trick" that is worth mentioning is the way we handle an adaptive adversary. To bound the probability of accidental collisions we analyze this probability in the presence of an augmented adversary, that can specify both the queries and their answers. That is, we let the adversary specify the entire transcript (with some minor restrictions) then choose some "permutation" π that maps the given queries to the given answers, and then consider the probability of accidental collisions. Clearly, this augmented adversary is no longer adaptive, hence the analysis becomes more tractable.

6 Some Insecure Modifications

In this section we justify two of our design choices by showing that changing them would result in insecure schemes. Specifically, we show that the block-cipher call that sits in between the two ECB layers is effectively unavoidable, and we show that that the length restriction $m < n$ also is needed.

6.1 The "extra" block-cipher call is needed

The EME construction has three block-cipher invocations in its "critical path" (that is, the construction is depth-3 in block-cipher gates). We now show that, in some sense, this is the best that you can do for a constructions of this type. Specifically, we show that for a construction of the type ECB-Mix-ECB, implementing the intermediate mixing layer by any linear transformation always results in an insecure scheme. This remains true even for an *un*tweakable scheme, even when one considers only fixed-input-length inputs, even when each block-cipher call in each ECB encryption layer uses an independent key, and even if the linear transformation in the middle is key-dependent. This result implies that, as opposed to the Hash-Encrypt-Hash approach that was proven secure by Naor and Reingold [15], the "dual" approach of Encrypt-Hash-Encrypt will not be secure under typical assumptions.[2]

Formally, fix $m, n \in \mathbb{N}$ with $m \geq 2$, and let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. The scheme $\mathbf{E} = \text{BrokenEME}$ is defined on message space $\{0,1\}^{mn}$ and key space $\mathcal{K}^{2m} \times \mathcal{K}'$ where $\mathcal{K}'$ is a set of invertible linear transformations on $\{0,1\}^{mn}$. BrokenEME is keyed with $2m$ independent keys $K_1, \ldots, K_m, K'_1, \ldots, K'_m \in \mathcal{K}$, and with an invertible (possibly secret) linear transformation[3]

[2] This may seem somewhat surprising, as one may think that Encrypt-Hash-Encrypt should be at least as secure since it uses "more cryptography".

[3] In fact, it is easy to see that the attack described below works also when T is an affine transformation.

$T\colon \{0,1\}^{mn} \to \{0,1\}^{mn}$. To encipher a plaintext $P = P_1 \cdots P_m \in \{0,1\}^{mn}$ we do the following:

- Set $PPP_i = E_{k_i}(P_i)$ for $i = 1 \ldots m$. Let $PPP = PPP_1 \cdots PPP_m$ be the concatenation of the PPP_i blocks ($PPP \in \{0,1\}^{mn}$).
- Apply the linear transformation T to obtain $CCC = CCC_1 \cdots CCC_m = T(PPP)$.
- Set $C_i = E_{k'_i}(CCC_i)$ for $i = 1 \ldots m$. The ciphertext is the concatenation of all the C_i blocks, $C = C_1 \cdots C_m \in \{0,1\}^{mn}$.

Deciphering is done in the obvious way.

We now give an adversary A that attacks the mode, distinguishing it from a truly random permutation and its inverse using only four queries. Denote the adversary with it's oracles as $A^{\mathcal{E}\,\mathcal{D}}$. The adversary A picks two mn-bit plaintexts that differ only in their first block, namely $P^1 = P_1 P_2 \cdots P_m$ and $P^2 = P'_1 P_2 \cdots P_m$ (with $P_1 \neq P'_1$). Then A queries its oracle as follows:

(1) Let $C^1 = C^1_1 \cdots C^1_m \leftarrow \mathcal{E}(P^1)$ and let $C^2 = C^2_1 \cdots C^2_m \leftarrow \mathcal{E}(P^2)$.
(2) Create two "complementing mixes" of the two ciphertexts, for example $C^3 = C^2_1 C^1_2 \cdots C^1_m$ and $C^4 = C^1_1 C^2_2 \cdots C^2_m$.
(3) Let $P^3 = P^3_1 \cdots P^3_m \leftarrow \mathcal{D}(C^3)$ and let $P^4 = P^4_1 \cdots P^4_m \leftarrow \mathcal{D}(C^4)$.

If the plaintext vectors P^3 and P^4 agree in all but their first block then A outputs 1 ("real") while otherwise it outputs 0 ("random"). To see that this works, we denote the intermediate variables in the four queries by PPP^i_j and CCC^i_j ($i = 1..4$ and $j = 1..m$) and denote the "vector of differences" between PPP^1 and PPP^2 by $DP = DP_1 \cdots DP_m \stackrel{\text{def}}{=} PPP^1 \oplus PPP^2$. Since P^1 and P^2 agree everywhere except in their first block, it follows that also the "vector of differences" DP is zero everywhere except in the first block. Similarly, we denote the "vector of differences" between CCC^1 and CCC^2 by $DC = DC_1 \cdots DC_m \stackrel{\text{def}}{=} CCC^1 \oplus CCC^2$ and since we computed $CCC^i = T(PPP^i)$ and T is a linear transformation, it follows that $DC = T(PPP^1) \oplus T(PPP^2) = T(PPP^1 \oplus PPP^2) = T(DP)$. Recall now that for any $j \in [1..m]$ we have either $C^3_j = C^1_j$ and $C^4_j = C^2_j$, or $C^3_j = C^2_j$ and $C^4_j = C^1_j$. It follows that for all j, $CCC^3_j \oplus CCC^4_j = CCC^1_j \oplus CCC^2_j = DC_j$, namely $CCC^3 \oplus CCC^4 = DC$. Putting this together we now compute $PPP^3 \oplus PPP^4$ as:

$$\begin{aligned} PPP^3 \oplus PPP^4 &= T^{-1}(CCC^3) \oplus T^{-1}(CCC^4) \\ &= T^{-1}(CCC^3 \oplus CCC^4) \;=\; T^{-1}(DC) \;=\; T^{-1}(T(DP)) \;=\; DP \end{aligned}$$

This means that $PPP^4_j = PPP^3_j$ for $j = 2..m$, and therefore also $P^4_j = P^3_j$ for all but the first block.

6.2 Necessity of the Length Restriction

Recall that EME is defined on message space $\mathcal{M} = \bigcup_{m \in [1..n]} \{0,1\}^{mn}$. Here we show that the restriction $m \le n$ is justified. In fact, we do not know whether allowing $m = n+1$ breaks the security of EME, but we can show that allowing $m = n+2$ permits easy distinguishing attacks. The details of the attack depend somewhat on the representation of the field $\mathrm{GF}(2^n)$. Below we demonstrate it for $n = 128$, where the field $\mathrm{GF}(2^{128})$ is represented using the polynomial $P_{128}(\mathsf{x}) = \mathsf{x}^{128} + \mathsf{x}^7 + \mathsf{x}^2 + \mathsf{x} + 1$.

Assume that $m \ge n+2$ and let J be a nonempty *proper subset* of the indexes from 2 to m, $J \subset \{2, 3, \ldots, m\}$, $J \neq \emptyset$, such that in the field $\mathrm{GF}(2^n)$ we have $\sum_{j \in J} 2^{j-1} = 0$. For example, when $\mathrm{GF}(2^{128})$ is represented using P_{128}, we have

$$2^{129} + 2^8 + 2^3 + 2^2 + 2^1 \;=\; 2(2^{128} + 2^7 + 2^2 + 2^1 + 2^0) \;=\; 0$$

so we can set $J = \{130, 9, 4, 3, 2\}$. The attack proceeds as follows:

(1) Pick an arbitrary tweak T. All the queries in the attack will use the same tweak T. (In other words, the attack works also when EME is used as an *un*tweakable scheme.) Pick two plaintext vectors that differ only in their first block, $P^1 = P_1P_2 \ldots P_m$ and $P^2 = P_1'P_2 \cdots P_m$ (with $P_1 \neq P_1'$).

(2) Encipher both plaintext vectors to get $C^1 = \mathcal{E}(T, P^1)$ and $C^2 = \mathcal{E}(T, P^2)$.

(3) Create a ciphertext vector C^3 such that $C_j^3 = \begin{cases} C_j^1 & \text{if } j \in J \\ C_j^2 & \text{if } j \notin J \end{cases}$.

(4) Decipher C^3 to get $P^3 = \mathcal{D}(T, C^3)$.

Output 1 ("real") if P^3 and P^2 agree in all the blocks $j \in ([2..m] \setminus J)$ and output 0 ("random") otherwise. To see that this works we denote the intermediate variables in the three queries by PPP_j^i and CCC_j^i and MP^i and MC^i and M^i ($i = 1..3$ and $j = 1..m$).

We note that $PPP_j^1 = PPP_j^2$ for all $j \in [2..m]$, and in particular for all $j \in J$. Also, from the construction of C^3 we get that $CCC_j^3 = CCC_j^1$ for $j \in J$ and $CCC_j^3 = CCC_j^2$ for $j \notin J$. Thus

$$\begin{aligned} MC^2 \oplus MC^3 &= \left(T \oplus \sum_{j=1}^{m} CCC_j^2\right) \oplus \left(T \oplus \sum_{j=1}^{m} CCC_j^3\right) \\ &= \sum_{j\in J} \left(CCC_j^2 \oplus CCC_j^3\right) \;=\; \sum_{j\in J} \left(CCC_j^2 \oplus CCC_j^1\right) \\ &= \sum_{j\in J} \left((PPP_j^2 \oplus 2^{j-1}M^2) \oplus (PPP_j^1 \oplus 2^{j-1}M^1)\right) \\ &= \sum_{j\in J} \left(2^{j-1}M^2 + 2^{j-1}M^1\right) \;=\; (M^2 + M^1)\sum_{j\in J} 2^{j-1} \;=\; 0 \end{aligned}$$

So we have $MC^3 = MC^2$ and therefore also $MP^3 = MP^2$ and $M^3 = M^2$. Thus for any $j \notin J$, $j > 1$ we have $PPP_j^3 = CCC_j^3 \oplus 2^{j-1}M^3 \;=\; CCC_j^2 + 2^{j-1}M^2 = PPP_j^2$ and therefore also $P_j^3 = P_j^2$.

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] 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.

[4] 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.

[5] S. Duplichan. A primitive polynomial search program. Web document. Available at http://users2.ev1.net/∼sduplichan/primitivepolynomials/primivitePolynomials.htm, 2003.

[6] 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. Full version available on the ePrint archive, http://eprint.iacr.org, 2003.

[7] J. Hughes. Personal communication, 2002.

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

[9] 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.

[10] C. Jutla. Encryption modes with almost free message integrity. In *Advances in Cryptography – EUROCRYPT'01*, volume 2999 of *Lecture Notes in Computer Science*, pages 529–544. Springer-Verlag, 2001.

[11] 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.

[12] 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/.

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

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

[15] 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/.

[16] 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.

[17] 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 Extending EME to Longer Messages

The restriction on the message size of EME, $m \leq n$, means, for example, that when using AES as the underlying cipher one cannot encrypt messages longer than 2KB. In some applications this restriction could be problematic. We now describe EME$^+$, an extension of EME that can be used to handle message of practically any length (as long as it is an integral number of blocks).

The idea is to divide the m-block input into "chunks" of at most n blocks each such that in each "chunk" we use construction similar to EME. Specifically, in the first "chunk" we use exactly the same construction as in EME. In all the other "chunks" we use a similar construction, except that we replace the addition of $SP \oplus T$ and $SC \oplus T$ (before and after the "extra block cipher call" in between the two ECB layers) by additions of the mask M_1 from the first "chunk".

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

$L \leftarrow 2E_K(0^n)$
for $i \in [1 \mathinner{..} m]$ **do**
  $PP_i \leftarrow 2^{i-1}\, L \oplus P_i,\;\; PPP_i \leftarrow E_K(PP_i)$

$MP_1 \leftarrow PPP_1 \oplus PPP_2 \oplus \cdots \oplus PPP_m \oplus T$
$MC_1 \leftarrow E_K(MP_1),\;\; M_1 \leftarrow MP_1 \oplus MC_1$
for $i \in [2..n]$ **do** $CCC_i \leftarrow PPP_i \oplus 2^{i-1} M_1$
for $j \in [2 \mathinner{..} \lceil m/n \rceil]$ **do**
  $MP_j \leftarrow PPP_{(j-1)n+1} \oplus M_1$
  $MC_j \leftarrow E_K(MP_j),\;\; M_j \leftarrow MP_j \oplus MC_j$
  $CCC_{1+(j-1)n} \leftarrow MC_j \oplus M_1$
  for $i \in [2+(j-1)n \mathinner{..} jn]$ **do**
    $CCC_i \leftarrow PPP_i \oplus 2^{i-1 \bmod n} M_j$
$CCC_1 \leftarrow MC_1 \oplus CCC_2 \oplus \cdots \oplus CCC_m \oplus T$

for $i \in [1 \mathinner{..} m]$ **do**
  $CC_i \leftarrow E_K(CCC_i),\;\; C_i \leftarrow CC_i \oplus 2^{i-1}\, L$

return $C_1 \cdots C_m$

Algorithm $\mathbf{D}_{Kt}^T(C_1 \cdots C_m)$

$L \leftarrow 2E_K(0^n)$
for $i \in [1 \mathinner{..} m]$ **do**
  $CC_i \leftarrow 2^{i-1}\, L \oplus C_i,\;\; CCC_i \leftarrow E_K^{-1}(CC_i)$

$MC_1 \leftarrow CCC_1 \oplus CCC_2 \oplus \cdots \oplus CCC_m \oplus T$
$MP_1 \leftarrow E_K^{-1}(MC_1),\;\; M_1 \leftarrow MC_1 \oplus MP_1$
for $i \in [2..n]$ **do** $PPP_i \leftarrow CCC_i \oplus 2^{i-1} M_1$
for $j \in [2 \mathinner{..} \lceil m/n \rceil]$ **do**
  $MC_j \leftarrow CCC_{(j-1)n+1} \oplus M_1$
  $MP_j \leftarrow E_K^{-1}(MC_j),\;\; M_j \leftarrow MC_j \oplus MP_j$
  $PPP_{1+(j-1)n} \leftarrow MP_j \oplus M_1$
  for $i \in [2+(j-1)n \mathinner{..} jn]$ **do**
    $PPP_i \leftarrow CCC_i \oplus 2^{i-1 \bmod n} M_j$
$PPP_1 \leftarrow MP_1 \oplus PPP_2 \oplus \cdots \oplus PPP_m \oplus T$

for $i \in [1 \mathinner{..} m]$ **do**
  $PP_i \leftarrow E_K^{-1}(PPP_i),\;\; P_i \leftarrow PP_i \oplus 2^{i-1}\, L$

return $P_1 \cdots P_m$

Figure 3: Enciphering (left) and deciphering (right) under $\mathbf{E} = \mathrm{EME}^+[E]$, 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 $C = C_1 \cdots C_m$.

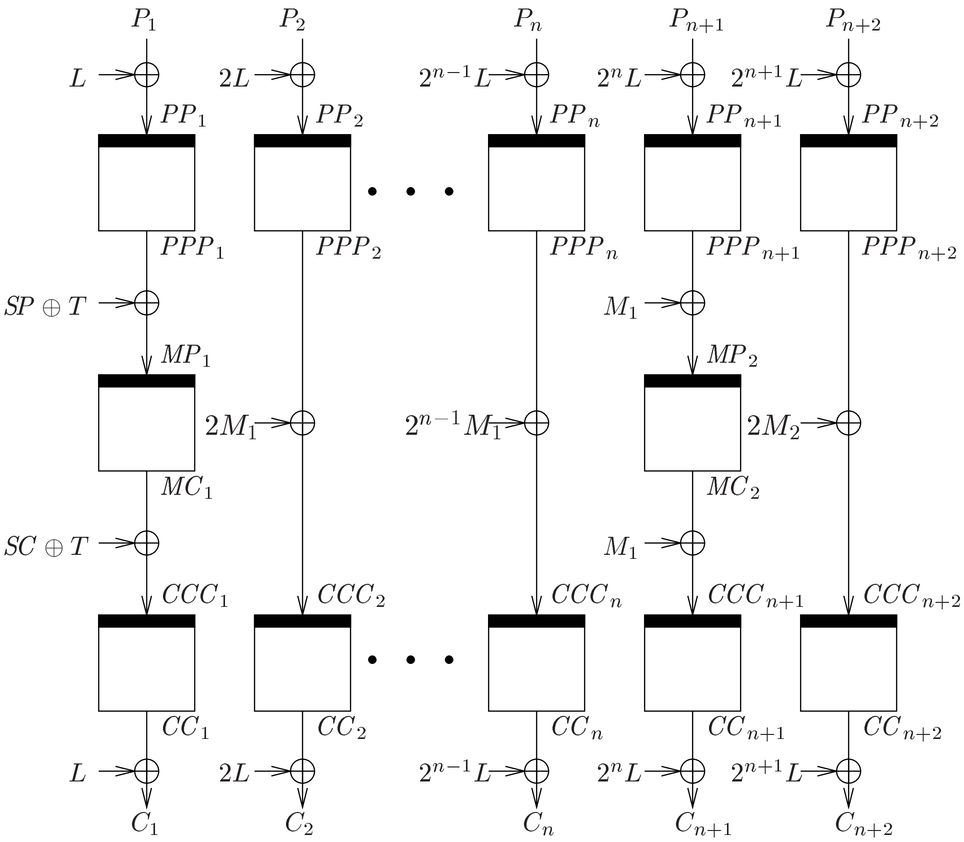

Figure 4: Enciphering an $(n+2)$-block message under EME^+. The boxes represent E_K and $L = 2E_K(0^n)$. We set $SP = PPP_2 \oplus \cdots PPP_m$, $M_i = MP_i \oplus MC_i$, and $SC = CCC_2 \oplus \cdots \oplus CCC_m$.

We specify in Figure 3 both the forward direction of our construction, $\mathbf{E} = \mathrm{EME}^+[E]$, and its inverse $\mathbf{D}$. An illustration of EME^+ mode is given in Figure 4. One observes that EME^+ is a "proper extension" of EME in that when we use it on a message of length $m \le n$ blocks, we get back the original EME mode.

Although we have not written a proof of security for EME^+ we expect that such proof can be written. One would follow the arguments in the proof for the basic EME in Appendix B, except that one needs to analyze a few more cases in the case analysis (specifically in the proof of Claim 7).

B Proof of Theorem 1 — Security of EME

Our proof of security for EME is divided into two parts: in Section B.1 we carry out a game-substitution argument, reducing the analysis of EME to the analysis of a simpler probabilistic game. In Section B.2 we analyze that simpler game. Before we begin we first recall a little lemma, saying that a (tweakable) truly random permutation looks very much like an oracle that just returns random bits (as long as you never ask pointless queries). So instead of analyzing indistinguishability from a random permutation we can analyze indistinguishability from random bits.

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

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

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, whose (standard) proof can be found, for example, in the full version of [6].

Lemma 2 [$\pm\widetilde{\mathrm{prp}}$**-security** $\approx \pm\widetilde{\mathrm{rnd}}$**-security**] Let $\mathbf{E}\colon \mathcal{K}\times\mathcal{T}\times\mathcal{M} \to \mathcal{M}$ be a tweaked block-cipher and let $q \ge 1$ be a number. Then $|\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathbf{E}}(q) - \mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathbf{E}}(q)| \le q(q-1)/2^{N+1}$ where N is the length of a shortest string in $\mathcal{M}$. $\square$

B.1 The game-substitution sequence

Fix n, σ_n, and q. Let A be an adversary that asks q oracle (none pointless) and has query complexity σ_n. Our goal is to show that $\mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) \le 2\Pr[\,\mathrm{NON2} \text{ sets } \mathit{bad}\,] + \frac{q\sigma_n}{2^n}$ where NON2 is some probability space and "NON2 sets *bad*" is an event defined there. Later we bound $\Pr[\,\mathrm{NON2} \text{ sets } \mathit{bad}\,]$, and, putting that together with Lemma 2, we get Equation (1) of Theorem 1.

Game NON2 is obtained by a game-substitution argument, as carried out in works like [11]. 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 EME1. We begin by describing the attack scenario of A against $\mathrm{EME}[\mathrm{Perm}(n)]$ 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 grows. 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 $\overline{\mathrm{Domain}}$ and $\overline{\mathrm{Range}}$ be

Subroutine Choose-$\pi(X)$:

$Y \xleftarrow{\$} \{0,1\}^n$; **if** $Y \in$ Range **then** $bad \leftarrow$ true , $Y \xleftarrow{\$} \overline{\text{Range}}$
if $X \in$ Domain **then** $bad \leftarrow$ true , $Y \leftarrow \pi(X)$
$\pi(X) \leftarrow Y$, Domain $\leftarrow$ Domain $\cup \{X\}$, Range $\leftarrow$ Range $\cup \{Y\}$; **return** Y

Subroutine Choose-$\pi^{-1}(Y)$:

$X \xleftarrow{\$} \{0,1\}^n$; **if** $X \in$ Domain **then** $bad \leftarrow$ true , $X \xleftarrow{\$} \overline{\text{Domain}}$
if $Y \in$ Range **then** $bad \leftarrow$ true , $X \leftarrow \pi^{-1}(Y)$
$\pi(X) \leftarrow Y$, Domain $\leftarrow$ Domain $\cup \{X\}$, Range $\leftarrow$ Range $\cup \{Y\}$; **return** X

Initialization:

$bad \leftarrow$ false; **for** all $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow$ undef
$EZ \xleftarrow{\$} \{0,1\}^n$; $L \leftarrow 2EZ$
$\pi(0^n) \leftarrow EZ$; Domain $\leftarrow \{0^n\}$; Range $\leftarrow \{EZ\}$

Respond to the s-th adversary query as follows:

AN ENCIPHER QUERY, $\mathsf{Enc}(T^s; P_1^s \cdots P_{m^s}^s)$:

for $i \leftarrow 1$ **to** m^s **do**
 Let $r = r[s,i]$ be the smallest index s.t. $P_i^s = P_i^r$
 if $r < s$ **then** $PP_i^s \leftarrow PP_i^r$, $PPP_i^s \leftarrow PPP_i^r$
 else $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L$; $PPP_i^s \leftarrow$ Choose-$\pi(PP_i^s)$

$MP^s \leftarrow PPP_1^s \oplus PPP_2^s \oplus \cdots \oplus PPP_{m^s}^s \oplus T^s$
$MC^s \leftarrow$ Choose-$\pi(MP^s)$
for $i \in [2 \,..\, m]$ **do** $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
$CCC_1^s \leftarrow MC^s \oplus CCC_2^s \oplus \cdots \oplus CCC_m^s \oplus T$

for $i \leftarrow 1$ **to** m^s **do**
 $CC_i^s \leftarrow$ Choose-$\pi(CCC_i^s)$; $C_i^s \leftarrow CC_i^s \oplus 2^{i-1}L$

return $C_1 \cdots C_{m^s}$

A DECIPHER QUERY, $\mathsf{Dec}(T^s; C_1^s \cdots C_{m^s}^s)$:

for $i \leftarrow 1$ **to** m^s **do**
 Let $r = r[s,i]$ be the smallest index s.t. $C_i^s = C_i^r$
 if $r < s$ **then** $CC_i^s \leftarrow CC_i^r$, $CCC_i^s \leftarrow CCC_i^r$
 else $CC_i^s \leftarrow C_i^s \oplus 2^{i-1}L$; $CCC_i^s \leftarrow$ Choose-$\pi^{-1}(CC_i^s)$

$MC^s \leftarrow CCC_1^s \oplus CCC_2^s \oplus \cdots \oplus CCC_{m^s}^s \oplus T^s$
$MP^s \leftarrow$ Choose-$\pi^{-1}(MC^s)$
for $i \in [2 \,..\, m]$ **do** $PPP_i^s \leftarrow CCC_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
$PPP_1^s \leftarrow MP^s \oplus PPP_2^s \oplus \cdots \oplus PPP_m^s \oplus T$

for $i \leftarrow 1$ **to** m^s **do**
 $PP_i^s \leftarrow$ Choose-$\pi^{-1}(PPP_i^s)$; $P_i^s \leftarrow PP_i^s \oplus 2^{i-1}L$

return $P_1 \cdots P_{m^s}$

Figure 5: Game EME1 describes the attack of A on $\mathrm{EME}[\mathrm{Perm}(n)]$, where the permutation π is chosen "on the fly" as needed. Game RND1 is the same as game EME1, except we do not execute the shaded statements.

Initialization:

$bad \leftarrow \mathsf{false}$; $EZ \xleftarrow{\$} \{0,1\}^n$; Domain $\leftarrow$ Range $\leftarrow \{0^n, EZ\}$; $L \leftarrow 2EZ$

Respond to the s-th adversary query as follows:

An encipher query, $\mathsf{Enc}(T^s; P_1^s \cdots P_{m^s}^s)$:

for $i \leftarrow 1$ **to** m^s **do**
  Let $r = r[s,i]$ be the smallest index s.t. $P_i^s = P_i^r$
  if $r < s$ **then** $PP_i^s \leftarrow PP_i^r$, $PPP_i^s \leftarrow PPP_i^r$
  else $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L$; **if** $PP_i^s \in$ Domain **then** $bad \leftarrow \mathsf{true}$
    $PPP_i^s \xleftarrow{\$} \{0,1\}^n$; **if** $PPP_i^s \in$ Range **then** $bad \leftarrow \mathsf{true}$
    Domain $\leftarrow$ Domain $\cup \{PP_i^s\}$, Range $\leftarrow$ Range $\cup \{PPP_i^s\}$

$MP^s \leftarrow PPP_1^s \oplus PPP_2^s \oplus \cdots \oplus PPP_{m^s}^s \oplus T^s$; **if** $MP^s \in$ Domain **then** $bad \leftarrow \mathsf{true}$
$MC^s \xleftarrow{\$} \{0,1\}^n$; **if** $MC^s \in$ Range **then** $bad \leftarrow \mathsf{true}$
Domain $\leftarrow$ Domain $\cup \{MP^s\}$, Range $\leftarrow$ Range $\cup \{MC^s\}$
for $i \in [2 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
$CCC_1^s \leftarrow MC^s \oplus CCC_2^s \oplus \cdots \oplus CCC_m^s \oplus T^s$

for $i \leftarrow 1$ **to** m^s **do**
  $C_i^s \xleftarrow{\$} \{0,1\}^n$;
  $CC_i^s \leftarrow C_i^s \oplus 2^{i-1}L$; **if** $CC_i^s \in$ Range **then** $bad \leftarrow \mathsf{true}$
  if $CCC_i^s \in$ Domain **then** $bad \leftarrow \mathsf{true}$
  Domain $\leftarrow$ Domain $\cup \{CCC_i^s\}$, Range $\leftarrow$ Range $\cup \{CC_i^s\}$

return $C_1 \cdots C_{m^s}$

A decipher query, $\mathsf{Dec}(T^s; C_1^s \cdots C_{m^s}^s)$, is treated symmetrically

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

the complement of these sets relative to $\{0,1\}^n$. The game, denoted EME1, is shown in Figure 5. Since game EME1 accurately represent the attack scenario, we have that

$$\Pr[\,A^{\mathbf{E}_\pi\,\mathbf{D}_\pi} \Rightarrow 1\,] \;=\; \Pr[\,A^{\mathrm{EME1}} \Rightarrow 1\,] \tag{3}$$

Game RND1. We next modify game EME1 by omitting the statements that immediately follow the setting of *bad* to true. (This is the usual trick under the game-substitution approach.) Namely, 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, and we reset out choice of Y as needed. Now we still make these checks and set the flag *bad*, but we do not reset the chosen value of Y. 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 EME1 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

$$\Pr[\,A^{\mathrm{EME1}} \Rightarrow 1\,] - \Pr[\,A^{\mathrm{RND1}} \Rightarrow 1\,] \;\le\; \Pr[\,A^{\mathrm{RND1}} \text{ sets } bad\,] \tag{4}$$

Game RND2. We now make three 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 $C_i^s \leftarrow CC_i^s \oplus 2^i L$ (in encipher queries, line 131), we will now choose $C_i^s \xleftarrow{\$} \{0,1\}^n$ and then set $CC_i^s \leftarrow C_i^s \oplus 2^i L$. 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 decipher queries; we could just as well have chosen P_i^s at random and defined PP_i^s using it. The last change that we make is to artificially add 0^n also to the Range set, and the value EZ (which was meant to represent $\pi(0^n)$) to the Domain set. This last change is meant to make the encipher and decipher directions completely symmetric, so that we can reduce the number of cases that we need to handle in the analysis to come. Note that this last change has no effect on the answers that are returned to the adversary. Its only effect is to slightly increase the probability that the flag *bad* is set.

The resulting 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.) It is clear that the changes we made do has no effect on the probability that A returns one (as they do not change anything in the interaction between A and its oracles), and they can only increase the probability of setting flag *bad*. Hence we conclude that

$$\Pr[\, A^{\mathrm{RND1}} \Rightarrow 1\,] = \Pr[\, A^{\mathrm{RND2}} \Rightarrow 1\,] \quad \text{and} \quad \Pr[\, A^{\mathrm{RND1}} \text{ sets } bad\,] \le \Pr[\, A^{\mathrm{RND2}} \text{ sets } bad\,] \qquad (5)$$

We note that in game RND2 we respond to any m-block Enc-query by returning nm random bits, $C_1 \cdots C_m$. Similarly, we respond to any m-block Dec-query by returning nm random bits, $P_1 \cdots P_m$. Thus RND2 provides an adversary with an identical view to a pair of random-bit oracles,

$$\Pr[\, A^{\mathrm{RND2}} \Rightarrow 1\,] \quad = \quad \Pr[\, A^{\widetilde{\pm\mathrm{rnd}}} \Rightarrow 1\,] \qquad (6)$$

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

$$\begin{aligned}\mathbf{Adv}^{\widetilde{\pm\mathrm{rnd}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) &= \Pr[\, A^{\mathrm{EME1}} \Rightarrow 1\,] - \Pr[\, A^{\mathrm{RND2}} \Rightarrow 1\,] \\ &= \Pr[\, A^{\mathrm{EME1}} \Rightarrow 1\,] - \Pr[\, A^{\mathrm{RND1}} \Rightarrow 1\,] \\ &\le \Pr[\, A^{\mathrm{RND1}} \text{ sets } bad\,] \\ &\le \Pr[\, A^{\mathrm{RND2}} \text{ sets } bad\,] \qquad (7)\end{aligned}$$

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

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 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 $C^s = C_1^s \cdots C_{m^s}^s$ or $P^s = P_1^s \cdots P_{m^s}^s$ 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:

$$\Pr[\, A^{\mathrm{RND2}} \text{ sets } bad\,] \quad = \quad \Pr[\, A^{\mathrm{RND3}} \text{ sets } bad\,] \qquad (8)$$

Respond to the s-th adversary query as follows:

AN ENCIPHER QUERY, $\mathsf{Enc}(T^s; P_1^s \cdots P_{m^s}^s)$:

010 $\quad ty^s \leftarrow \mathsf{Enc}$
011 $\quad C^s = C_1^s \cdots C_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
012 $\quad$ **return** C^s

A DECIPHER QUERY, $\mathsf{Dec}(T^s; C_1^s \cdots C_{m^s}^s)$:

020 $\quad ty^s \leftarrow \mathsf{Dec}$
021 $\quad P^s = P_1^s \cdots P_{m^s}^s \xleftarrow{\$} \{0,1\}^{nm^s}$
022 $\quad$ **return** P^s

Finalization:

FIRST PHASE

050 $\quad EZ \xleftarrow{\$} \{0,1\}^n;\ \ L \leftarrow 2EZ;\ \ \mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \{0^n, EZ\}$ $\qquad$ // $\mathfrak{D}, \mathfrak{R}$ are *multisets*
051 $\quad$ **for** $s \leftarrow 1$ **to** q **do**
100 $\qquad$ **if** $ty^s = \mathsf{Enc}$ **then**
110 $\qquad\quad$ **for** $i \leftarrow 1$ **to** m^s **do**
111 $\qquad\qquad$ Let $r = r[s,i]$ be the smallest index s.t. $P_i^s = P_i^r$
112 $\qquad\qquad$ **if** $r < s$ **then** $PP_i^s \leftarrow PP_i^r,\ \ PPP_i^s \leftarrow PPP_i^r$
113 $\qquad\qquad$ **else** $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$
114 $\qquad\qquad\quad PPP_i^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$
120 $\qquad\quad MP^s \leftarrow PPP_1^s \oplus PPP_2^s \oplus \cdots \oplus PPP_{m^s}^s \oplus T^s;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s\}$
121 $\qquad\quad MC^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC^s\}$
122 $\qquad\quad$ **for** $i \in [2 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
123 $\qquad\quad CCC_1^s \leftarrow MC^s \oplus CCC_2^s \oplus \cdots \oplus CCC_m^s \oplus T^s$
130 $\qquad\quad$ **for** $i \leftarrow 1$ **to** m^s **do**
131 $\qquad\qquad CC_i^s \leftarrow C_i^s \oplus 2^{i-1}L;\ \ \mathfrak{R} \leftarrow \mathrm{Range} \cup \{CC_i^s\}$
132 $\qquad\qquad \mathfrak{D} \leftarrow \mathrm{Domain} \cup \{CCC_i^s\}$
200 $\qquad$ The case $ty^s = \mathsf{Dec}$ is treated symmetrically

SECOND PHASE

300 $\quad bad \leftarrow$ (some value appears more than once in $\mathfrak{D}$)
$\qquad\qquad$ **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 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 RND3. We may assume that the adversary is deterministic, and so the probability is over the random choices $P^s \xleftarrow{\$} \{0,1\}^{nm^s}$ and $C^s \xleftarrow{\$} \{0,1\}^{nm^s}$ that are made while answering the queries (in lines 011 and 021), and the random choices $EZ \xleftarrow{\$} \{0,1\}^n$, $PPP_i^s \xleftarrow{\$} \{0,1\}^n$, $MP^s \xleftarrow{\$} \{0,1\}^n$, $MC^s \xleftarrow{\$} \{0,1\}^n$, and $CCC_i^s \xleftarrow{\$} \{0,1\}^n$ that are made in the first finalization phase (lines 050, 113, 120, 213, and 220). 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.

050 $EZ \xleftarrow{\$} \{0,1\}^n;\ \ L \leftarrow 2EZ;\ \ \mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \{0^n, EZ\}$ // $\mathfrak{D}, \mathfrak{R}$ are *multisets*

051 **for** $s \leftarrow 1$ **to** q **do**
100   **if** $\mathsf{ty}^s = \mathsf{Enc}$ **then**
110   **for** $i \leftarrow 1$ **to** m^s **do**
111   Let $r = r[s,i]$ be the smallest index s.t. $\mathsf{P}_i^s = \mathsf{P}_i^r$
112   **if** $r < s$ **then** $PP_i^s \leftarrow PP_i^r,\ \ PPP_i^s \leftarrow PPP_i^r$
113   **else** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus 2^{i-1}L;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$
114                 $PPP_i^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$

120         $MP^s \leftarrow PPP_1^s \oplus PPP_2^s \oplus \cdots \oplus PPP_{\mathsf{m}^s}^s \oplus \mathsf{T}^s;\ \ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s\}$
121         $MC^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC^s\}$
122   **for** $i \in [2 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
123         $CCC_1^s \leftarrow MC^s \oplus CCC_2^s \oplus \cdots \oplus CCC_m^s \oplus \mathsf{T}^s$

130   **for** $i \leftarrow 1$ **to** m^s **do**
131             $CC_i^s \leftarrow \mathsf{C}_i^s \oplus 2^{i-1}L;\ \ \mathfrak{R} \leftarrow \mathrm{Range} \cup \{CC_i^s\}$
132             $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$

200   **else** ($\mathsf{ty}^s = \mathsf{Dec}$)
210   **for** $i \leftarrow 1$ **to** m^s **do**
211   Let $r = r[s,i]$ be the smallest index s.t. $\mathsf{C}_i^s = \mathsf{C}_i^r$
212   **if** $r < s$ **then** $CC_i^s \leftarrow CC_i^r,\ \ CCC_i^s \leftarrow CCC_i^r$
213   **else** $CC_i^s \leftarrow \mathsf{C}_i^s \oplus 2^{i-1}L;\ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC_i^s\}$
214                 $CCC_i^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$

220         $MC^s \leftarrow CCC_1^s \oplus CCC_2^s \oplus \cdots \oplus CCC_{\mathsf{m}^s}^s \oplus \mathsf{T}^s;\ \ \ \mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC^s\}$
221         $MP^s \xleftarrow{\$} \{0,1\}^n;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s\}$
222   **for** $i \in [2 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
223         $PPP_1^s \leftarrow MP^s \oplus PPP_2^s \oplus \cdots \oplus PPP_m^s \oplus \mathsf{T}^s$

230   **for** $i \leftarrow 1$ **to** m^s **do**
231             $PP_i^s \leftarrow \mathsf{P}_i^s \oplus 2^{i-1}L;\ \ \mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$
232             $\mathfrak{R} \leftarrow \mathrm{Range} \cup \{PPP_i^s\}$

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 $\boldsymbol{\tau} = (\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$ is a fixed, allowed transcript.

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 i'th ciphertext block in the two answers is the same, $C_i^r = C_i^s$. Then the flag *bad* is sure to be set, since we will have a "collision" between CC_i^r and CC_i^s. Formally, since in line 131 we set $CC_i^r = C_i^r \oplus 2^{i-1}L = C_i^s \oplus 2^{i-1}L = CC_1^s$, and since both CC_i^r and CC_i^s are added to $\mathfrak{R}$ we would set *bad* when we examine their values in line 300. A similar example can be shown for decipher queries. We call such collisions *immediate* collisions. Formally, an immediate collision happens whenever we have $C_i^r = C_i^s$ ($r < s$) and query s is an encipher query, and whenever we have $P_i^r = P_i^s$ ($r < s$) and query s is a decipher query. The probability of an immediate collision in game RND3 is at most

$$\sum_{s=1}^{q} \frac{m^s(s-1)}{2^n} \; < \; \frac{q}{2^n} \sum_{s=1}^{q} m^s \; = \; \frac{q\sigma_n}{2^n}$$

We make from the Finalization part of game RND3 a new game, game NON1 (for "noninteractive"). This game silently depends on a fixed transcript $\boldsymbol{\tau} = \langle \mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C} \rangle$ with $\mathbf{ty} = (\mathsf{ty}^1, \cdots, \mathsf{ty}^q)$, $\mathbf{T} =$

$(\mathsf{T}^1, \cdots, \mathsf{T}^q)$, and $\mathbf{P} = (\mathsf{P}^1, \cdots, \mathsf{P}^q)$, and $\mathbf{C} = (\mathsf{C}^1, \cdots, \mathsf{C}^q)$ where $\mathsf{ty}^s \in \{\mathsf{Enc}, \mathsf{Dec}\}$, $\mathsf{T}^s \in \{0,1\}^n$, and $\mathsf{P}^s = \mathsf{P}^s_1 \cdots \mathsf{P}^s_{\mathsf{m}^s}$ and $\mathsf{C}^s = \mathsf{C}^s_1 \cdots \mathsf{C}^s_{\mathsf{m}^s}$ for $|\mathsf{P}^r_i| = |\mathsf{C}^r_i| = n$. 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) $(\mathsf{T}^s, \mathsf{P}^s) \neq (\mathsf{T}^r, \mathsf{P}^r)$ and (ii) $\mathsf{C}^s_i \neq \mathsf{C}^r_i$ for any $i \in [1..m]$; while if $\mathsf{ty}^s = \mathsf{Dec}$ then (i) $(\mathsf{T}^s, \mathsf{C}^s) \neq (\mathsf{T}^r, \mathsf{C}^r)$ and (ii) $\mathsf{P}^s_i \neq \mathsf{P}^r_i$ for any $i \in [1..m]$. 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

$$\Pr[\, A^{\mathrm{RND3}} \text{ sets } bad\,] \quad \le \quad \Pr[\,\mathrm{NON1} \text{ sets } bad\,] + \frac{q\sigma_n}{2^n} \tag{9}$$

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

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

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

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

$$\Pr[\,\mathrm{NON1} \text{ sets } bad\,] \quad \le \quad 2 \cdot \Pr[\,\mathrm{NON2} \text{ sets } bad\,] \tag{10}$$

B.2 Analysis of the non-interactive game

In the analysis we view the multiset $\mathfrak{D}$ as a set of formal variables (rather than a multiset containing the values that these variables assume). Namely, whenever we set $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{X\}$ for some variable X we 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,

$$\begin{aligned}
\mathfrak{D} \;=\; & \{\mathrm{Zero}, EZ\} \;\cup\; \{MP^s \mid \; s \le q\} \\
& \cup\; \{PP^s_i \mid \; \mathsf{ty}^s = \mathsf{Dec},\; i \le \mathsf{m}^s\} \;\cup\; \{PP^s_i \mid \; \mathsf{ty}^s = \mathsf{Enc},\; i \le \mathsf{m}^s,\; s = \mathsf{r}[s,i]\} \\
& \cup\; \{CCC^s_i \mid \; \mathsf{ty}^s = \mathsf{Enc},\; i \le \mathsf{m}^s\} \;\cup\; \{CCC^s_i \mid \; \mathsf{ty}^s = \mathsf{Dec},\; i \le \mathsf{m}^s,\; s = \mathsf{r}[s,i]\}
\end{aligned}$$

050 $EZ \xleftarrow{\$} \{0,1\}^n$; $L \leftarrow 2EZ$; $\boxed{\mathfrak{D} \leftarrow \{0^n, EZ\}}$
051 **for** $s \leftarrow 1$ **to** q **do**
100   **if** $\mathsf{ty}^s = \mathsf{Enc}$ **then**
110   **for** $i \leftarrow 1$ **to** m^s **do**
111   Let $r = r[s,i]$ be the smallest index s.t. $\mathsf{P}_i^s = \mathsf{P}_i^r$
112   **if** $r < s$ **then** $PP_i^s \leftarrow PP_i^r$, $PPP_i^s \leftarrow PPP_i^r$
113   **else** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus 2^{i-1}L$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}}$
114                 $PPP_i^s \xleftarrow{\$} \{0,1\}^n$
120         $MP^s \leftarrow PPP_1^s \oplus PPP_2^s \oplus \cdots \oplus PPP_{\mathsf{m}^s}^s \oplus \mathsf{T}^s$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s\}}$
121         $MC^s \xleftarrow{\$} \{0,1\}^n$
122   **for** $i \in [2 \mathinner{..} m]$ **do** $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
123         $CCC_1^s \leftarrow MC^s \oplus CCC_2^s \oplus \cdots \oplus CCC_m^s \oplus \mathsf{T}^s$
130   **for** $i \leftarrow 1$ **to** m^s **do** $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}}$
200   **else** ($\mathsf{ty}^s = \mathsf{Dec}$)
210   **for** $i \leftarrow 1$ **to** m^s **do**
211   Let $r = r[s,i]$ be the smallest index s.t. $\mathsf{C}_i^s = \mathsf{C}_i^r$
212   **if** $r < s$ **then** $CCC_i^s \leftarrow CCC_i^r$
213   **else** $CCC_i^s \xleftarrow{\$} \{0,1\}^n$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}}$
220         $MC^s \leftarrow CCC_1^s \oplus CCC_2^s \oplus \cdots \oplus CCC_{\mathsf{m}^s}^s \oplus \mathsf{T}^s$
221         $MP^s \xleftarrow{\$} \{0,1\}^n$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s\}}$
222   **for** $i \in [2 \mathinner{..} m]$ **do** $PPP_i^s \leftarrow CCC_i^s \oplus 2^{i-1}(MP^s \oplus MC^s)$
223         $PPP_1^s \leftarrow MP^s \oplus PPP_2^s \oplus \cdots \oplus PPP_m^s \oplus \mathsf{T}^s$
230   **for** $i \leftarrow 1$ **to** m^s **do** $PP_i^s \leftarrow \mathsf{P}_i^s \oplus 2^{i-1}L$; $\boxed{\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}}$
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 random selection by shading, and statements that grow $\mathfrak{D}$ by boxing.

(where "Zero" is a formal constant that always assumes the value 0^n). 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 $\boldsymbol{\tau}$. Our goal is to show the following:

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 variable $X, X' \in \mathfrak{D}$ we have that $\Pr[X = X'] \leq 2^{-n}$.

Before proving Claim 1, we show how to use it to complete the proof of Theorem 1. Due to our conventions on how to measure the query complexity there are no more than $2\sigma_n + 1$ variables in $\mathfrak{D}$ so the union bound gives us that

$$\Pr[\,\text{NON2 sets } bad\,] \leq \binom{2\sigma_n + 1}{2} / 2^n \tag{11}$$

Combining Lemma 2 with Equations 7, 8, 9, 10 and 11 we are done:

$$\begin{aligned}
\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) &\le \mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) + q(q-1)/2^{n+1} \\
&\le 2\cdot \Pr[\text{NON2 sets } \mathit{bad}] + q\sigma_n/2^n + q(q-1)/2^{n+1} \\
&\le 2\cdot \binom{2\sigma_n+1}{2}/2^n + q\sigma_n/2^n + q(q-1)/2^{n+1} \\
&\le \frac{(2\sigma_n+1)(2\sigma_n)+\sigma_n^2+0.5\sigma_n^2}{2^n} \\
&\le \frac{7\,\sigma_n^2}{2^n}
\end{aligned}$$

Since A was an arbitrary adversary with query complexity of σ_n we are done. ▮

THE CASE ANALYSIS. We now need to prove Claim 1. We first prove a few claims (Claims 4 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 $EZ \xleftarrow{\$} \{0,1\}^n$ in line 050, the selections $PPP_i^s \xleftarrow{\$} \{0,1\}^n$ and $MC^s \xleftarrow{\$} \{0,1\}^n$ on encipher (lines 114, 121), and the selections $CCC_i^s \xleftarrow{\$} \{0,1\}^n$ and $MP^s \xleftarrow{\$} \{0,1\}^n$ on decipher (lines 213, 221). Specifically, the variables that are directly chosen at random are EZ, the variables PPP_i^s from encipher queries such that $s = \mathsf{r}[s,i]$, the variables CCC_i^s from decipher queries such that $s = \mathsf{r}[s,i]$, the variables MP^s from encipher queries, and the variables MC^s from decipher queries. Hereafter we refer to these variables as the *free variables* of the game, and we let $\mathfrak{F}$ denote the set of them:

$$\begin{aligned}
\mathfrak{F} \;=\; &\{EZ\} \;\cup\; \{MP^s \mid \;\mathsf{ty}^s = \mathsf{Dec}\} \;\cup\; \{MC^s \mid \;\mathsf{ty}^s = \mathsf{Enc}\} \\
&\cup\; \{PPP_i^s \mid \;\mathsf{ty}^s = \mathsf{Enc},\; i \le \mathsf{m}^s,\; s = \mathsf{r}[s,i]\} \\
&\cup\; \{CCC_i^s \mid \;\mathsf{ty}^s = \mathsf{Dec},\; i \le \mathsf{m}^s,\; s = \mathsf{r}[s,i]\}
\end{aligned}$$

The value of any other variable in the game can be expressed as a function in these free variables. In fact, the bulk of the argument below is to show that for any pair of variables in $\mathfrak{D}$, either their sum is some non-zero constant, or else it depends linearly on at least one free variable. We first prove a little helpful observation.

Claim 2 If r is an encipher query ($ty^r = \mathsf{Enc}$) and $I \subseteq [1..\mathsf{m}^r]$ is a non-empty set, then we have $\sum_{i\in I} CCC_i^r = aMC^r + \beta$, where $a \ne 0$ is a constant (that depends on the set I) and β is an expression involving only constants and free variables that are determined before MC^r in the game NON2.

Likewise, if r is a decipher query ($ty^r = \mathsf{Dec}$) and $I \subseteq [1..\mathsf{m}^r]$ is a non-empty set, then $\sum_{i\in I} PPP_i^r = aMP^r + \beta$, where $a \ne 0$ is a constant and β is an expression involving only constants and free variables that are determined before MP^r in the game NON2.

Proof: We prove here only the first assertion. The proof of the other assertion is completely symmetric. Since r is an encipher query, then the values of all the CCC_i^r's except the first (i.e., $i \in [2..\mathsf{m}^r]$), are set in line 122, $CCC_i^r \leftarrow PPP_i^r \oplus 2^{i-1}(MP^r \oplus MC^r)$. If $1 \notin I$ then we have

$$\begin{aligned}
\sum_{i\in I} CCC_i^r &= \sum_{i\in I} PPP_i^r \oplus 2^{i-1}(MP^r \oplus MC^r) \\
&= \text{things-that-were-determined-before-}MC^r \;\;\oplus \left(\sum_{i\in I} 2^{i-1}\right) MC^r
\end{aligned}$$

and the coefficient of MC^r is non-zero since I is non-empty.[4]

For the case where $1 \in I$, since in line 123 we set $CCC_1^r \leftarrow MC^r \oplus \sum_{i=2}^{\mathsf{m}^r} CCC_i^r$, we get

$$\begin{aligned}
\sum_{i \in I} CCC_i^r &= CCC_1^r \oplus \sum_{i \in I,\ i \neq 1} CCC_i^r \\
&= \left(MC^r \oplus \sum_{i=2}^{\mathsf{m}^r} CCC_i^r \right) \oplus \sum_{i \in I,\ i \neq 1} CCC_i^r \\
&= MC^r \oplus \sum_{i \notin I,\ 2 \leq i \leq \mathsf{m}^r} CCC_i^r \\
&= MC^r \oplus \sum_{i \notin I,\ 2 \leq i \leq \mathsf{m}^r} PPP_i^r \oplus 2^{i-1}(MP^r \oplus MC^r) \\
&= \text{things-that-were-determined-before-}MC^r \ \oplus \left(1 \oplus \sum_{i \notin I,\ 2 \leq i \leq \mathsf{m}^r} 2^{i-1} \right) MC^r
\end{aligned}$$

and again, the coefficient of MC^r is non-zero. ▮

To simplify the case analysis to come, we consider, for each variable $X \in \mathfrak{D}$, the *last free variable* (in the ordering of the game NON2) that X depends on, denoted $\phi(X)$. Formally, we have a function $\phi\colon \mathfrak{D} \to \mathfrak{F} \cup \{\mathsf{none}\}$ that is defined as follows:

- As "Zero" is a constant, we denote $\phi(\text{Zero}) = \mathsf{none}$.
- For the formal variables $EZ, PP_i^s \in \mathfrak{D}$, this last free variable is EZ, $\phi(EZ) = \phi(PP_i^s) = EZ$.
- For a formal variable $CCC_i^s \in \mathfrak{D}$ this last free variable $\phi(CCC_i^s)$ is either CCC_i^s itself (on decipher)[5] or MC^s (on encipher). (The last assertion is a corollary of Claim 2 for $I = \{i\}$.)
- The rules for MP^s are a bit more involved. Clearly, on decipher we have $\phi(MP^s) = MP^s$. For encipher, recall that we set (in line 120) $MP^s \leftarrow \mathsf{T}^s \oplus \sum_{i=1}^{m} PPP_i^s$, so the last free variable that MP^s depends on, is the "last of the free variables that any PPP_i^s depends on".

  Each of these PPP_i^s's can either be a free variable itself (if this is a "new block", $s = \mathsf{r}[s,i]$), or it can be set equal to some prior PPP_i^r (if $r = \mathsf{r}[s,i] < s$). In the latter case, PPP_i^r is either a free variable (if query r is encipher), or else it depends on MP^r (if query r is decipher). To define $\phi(MP^s)$, we therefore denote

  $$\mathsf{rmax}[s] \stackrel{\text{def}}{=} \max\{\ \mathsf{r}[s,i] \ \mid\ 1 \leq i \leq m\}\ , \qquad \mathsf{imax}[s] \stackrel{\text{def}}{=} \max\{\ i \ \mid\ \mathsf{r}[s,i] = \mathsf{rmax}[s]\}$$

  and then, on encipher ($\mathsf{ty}^s = \mathsf{Enc}$) we have

  $$\phi(MP^s) = \begin{cases} MP^{\mathsf{rmax}[s]} & \text{if } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Dec} \\ PPP_{\mathsf{imax}[s]}^{\mathsf{rmax}[s]} & \text{if } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Enc} \end{cases}$$

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:

[4] Here we use the fact that $\mathsf{m}^r \leq n$.

[5] Note that $CCC_i^s \in \mathfrak{D}$, which means that C_i^s is "a new block", $s = \mathsf{r}[s,i]$.

$$
\begin{array}{rcll}
\phi(\text{Zero}) & = & \mathsf{none} & & \text{Z} \\
\phi(EZ) & = & EZ & & \text{EZ} \\
\phi(PP_i^s) & = & EZ & \text{if } \mathsf{ty}^s = \mathsf{Dec} \text{ or } s = \mathsf{r}[s,i] & \text{PP} \\
\phi(CCC_i^s) & = & \begin{cases} CCC_i^s \\ MC^s \end{cases} & \begin{array}{l} \text{if } \mathsf{ty}^s = \mathsf{Dec} \text{ and } s = \mathsf{r}[s,i] \\ \text{if } \mathsf{ty}^s = \mathsf{Enc} \end{array} & \begin{array}{l} \text{CCC1} \\ \text{CCC2} \end{array} \\
\phi(MP^s) & = & \begin{cases} MP^s \\ MP^{\mathsf{rmax}[s]} \\ PPP^{\mathsf{rmax}[s]}_{\mathsf{imax}[s]} \end{cases} & \begin{array}{l} \text{if } \mathsf{ty}^s = \mathsf{Dec} \\ \text{if } \mathsf{ty}^s = \mathsf{Enc} \text{ and } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Dec} \\ \text{if } \mathsf{ty}^s = \mathsf{Enc} \text{ and } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Enc} \end{array} & \begin{array}{l} \text{MM1} \\ \text{MM2} \\ \text{MM3} \end{array}
\end{array}
$$

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

Claim 3 Let $X \in \mathfrak{D}$ be a formal variable, and let $Y = \phi(X)$. If $Y \neq \mathsf{none}$ then the value that X assumes in game NON2 can be expressed as $\text{value}(X) = a \cdot \text{value}(Y) \oplus \beta$ where $a \neq 0$ is a constant (that depends on the name of the formal variable X and the fixed transcript $\boldsymbol{\tau}$) and β is an expression involving only constants and free variables that are determined before Y in the game NON2. □

As an immediate corollary of Claim 3, we get the following.

Claim 4 Let $X, X' \in \mathfrak{D}$ be formal variables such that $\phi(X) \neq \phi(X')$. Then $\Pr[X = X'] = 2^{-n}$.

Proof: let $Y = \phi(X)$ and let $Y' = \phi(X')$, and assume that Y' occurs before Y in NON2. By Claim 3 above, we have $X \oplus X' = a \cdot Y \oplus \beta \;\oplus\; a' \cdot Y' \oplus \beta'$ where $a \neq 0$ is a constant, and $\beta \oplus \; a' \cdot Y' \;\oplus \beta'$ is an expression involving only constants and free variables that are determined before Y. As the value of Y is chosen at random from $\text{GF}(2^n)$, independently of the other free variables, it follows that $\Pr[X = X'] = 2^{-n}$. ▮

Claim 4 leaves us with the task of analyzing collisions between variables that depend on the same last free variable. These are handled in the next three propositions.

Claim 5 Let $X, X' \in \mathfrak{D}$ be two distinct formal variables, such that $\phi(X) = \phi(X') = EZ$. Then $\Pr[X = X'] \leq 2^{-n}$.

Proof: Recall that $L = 2EZ$. The types of collisions that we need to analyze are either EZ vs. PP_i^s for some $PP_i^s \in \mathfrak{D}$, or PP_i^s vs. $PP_{i'}^{s'}$ for $PP_i^s, PP_{i'}^{s'} \in \mathfrak{D}$. We start with collisions of the type PP_i^s vs. $PP_{i'}^{s'}$. If $i \neq i'$ then we have

$$PP_i^s \oplus PP_{i'}^{s'} = (\mathsf{P}_i^s \oplus 2^{i-1}L) \oplus (\mathsf{P}^{s'}{}'_i \oplus 2^{i'-1}L) = \mathsf{P}_i^s \oplus \mathsf{P}^{s'}{}'_i \oplus (2^i \oplus 2^{i'})EZ$$

and as $i \neq i'$, the coefficient of EZ is non-zero, and therefore $\Pr[PP_i^s \oplus PP_{i'}^{s'} = 0^n] = 2^{-n}$. If $i = i'$ and $s' < s$ then necessarily $\mathsf{P}_i^s \neq \mathsf{P}_{i'}^{s'}$. (Otherwise, either query s is encipher, in which case $r[s,i] < s$

and $PP_i^s \notin \mathfrak{D}$, or query s is encipher, which means that the transcript $\boldsymbol{\tau}$ specifies an immediate collision.) Therefore, with probability one we have $PP_i^s \oplus PP_{i'}^{s'} = (\mathsf{P}_i^s \oplus 2^{i-1}L) \oplus (\mathsf{P}^{s'}{}_i \oplus 2^{i-1}L) = \mathsf{P}_i^s \oplus \mathsf{P}^{s'}{}_i \neq 0$. For the other type of collisions, EZ vs. PP_i^s, we have

$$PP_i^s \oplus EZ = (\mathsf{P}_i^s \oplus 2^{i-1}L) \oplus EZ = (\mathsf{P}_i^s \oplus 2^i EZ) \oplus EZ = \mathsf{P}_i^s \oplus (1 \oplus 2^i)EZ$$

and again, since $i \geq 1$ the coefficient of EZ is non-zero. ▌

Claim 6 For any two distinct variables $CCC_i^s, CCC_{i'}^{s'} \in \mathfrak{D}$ with $\phi(CCC_i^s) = \phi(CCC_{i'}^{s'})$, $\Pr[CCC_i^s = CCC_{i'}^{s'}] = 2^{-n}$.

Proof: By inspecting Figure 10, we see that for two variable $CCC_i^s, CCC_{i'}^{s'} \in \mathfrak{D}$, the equality $\phi(CCC_i^s) = \phi(CCC_{i'}^{s'})$ implies that $s = s'$, and that this is a encipher query, $\mathsf{ty}^s = \mathsf{Enc}$ (in which case $\phi(CCC_i^s) = \phi(CCC_{i'}^{s'}) = MC^s$).

By Claim 2 (with $I = \{i, i'\}$), we can write $CCC_i^s \oplus CCC_{i'}^s = aMC^s + \beta$ where a is a non-zero constant and β is an expression involving only constants and variables that were determined before MC^s. Hence we have

$$\Pr[CCC_i^s = CCC_{i'}^{s'}] \;=\; \Pr[aMC^s + \beta = 0^n] \;=\; \Pr[MC^s \;=\; a^{-1}\beta] \;=\; 2^{-n}$$

since MC^s is a free variable. ▌

The most involved case to analyze (indeed, the one that embodies the "real reason" that EME is secure) is collisions of the type $MP^s = MP^{s'}$. These are analyzed in Claim 7 below.

Claim 7 For any two distinct variables $MP^s, MP^t \in \mathfrak{D}$ with $\phi(MP^s) = \phi(MP^t)$, it holds that $\Pr[MP^s = MP^t] \leq 2^{-n}$.

Proof: Fix some $s < t$ such that $\phi(MP^t) = \phi(MP^s)$. In Figure 10 we see that the equality $\phi(MP^s) = \phi(MP^t)$ implies that the later query t must be encipher (i.e., $\mathsf{ty}^t = \mathsf{Enc}$), and either query s is decipher with $\mathsf{rmax}[t] = s$, or query s is encipher with $\mathsf{rmax}[t] = \mathsf{rmax}[s]$.

If the plaintext vectors in both queries are the same (i.e., they have $\mathsf{P}^s = \mathsf{P}^t$) then it must be that the tweaks differ between them, $\mathsf{T}^s \neq \mathsf{T}^t$ (since the transcript $\boldsymbol{\tau}$ does not specify pointless queries). From $\mathsf{P}^s = \mathsf{P}^t$ it follows that $PPP_i^s = PPP_i^t$ for all i, and therefore with probability one we have $MP^s \oplus MP^t = \mathsf{T}^s \oplus \mathsf{T}^t \neq 0$.

So from now on we assume that $\mathsf{P}^s \neq \mathsf{P}^t$. Let E be the set of indexes where $\mathsf{P}^s, \mathsf{P}^t$ are equal, $E \stackrel{\text{def}}{=} \{\, i \leq \min(\mathsf{m}^s, \mathsf{m}^t) \;\mid\; \mathsf{P}_i^s = \mathsf{P}_i^t\}$. We note that the sum $MP^s \oplus MP^t$ can be written as

$$\begin{aligned} MP^s \oplus MP^t &= \mathsf{T}^s \oplus \sum_{i=1}^{\mathsf{m}^s} PPP_i^s \;\oplus \mathsf{T}^t \oplus \sum_{i=1}^{\mathsf{m}^t} PPP_i^t \\ &= \mathsf{T}^s \oplus \mathsf{T}^{s'} \oplus \sum_{i \leq \mathsf{m}^s, i \notin E} PPP_i^s \;\oplus \sum_{i \leq \mathsf{m}^t, i \notin E} PPP_i^t \qquad (12) \end{aligned}$$

where the last equality is justified since $\mathsf{P}_i^s = \mathsf{P}_i^t$ implies $PPP_i^s = PPP_i^t$.

We first analyze the case where query s is decipher, $\mathsf{ty}^s = \mathsf{Dec}$, and furthermore P^s is not a proper prefix of P^t. Since query s is decipher, we have in this case $\mathsf{rmax}[t] = s$, which means that all the blocks P_i^t already appeared in queries no later than s, namely $\mathsf{r}[t,i] \leq s$ for all $i \in [1..\mathsf{m}^t]$. Since for any $i \notin E$ we have $\mathsf{P}^s_i \neq \mathsf{P}^t_i$, it follows that for these indexes we have $\mathsf{r}[t,i] < s$ (if $\mathsf{r}[t,i]$ is defined at all). Thus we get

$$\begin{aligned} MP^s \oplus MP^t &= \mathsf{T}^s \oplus \mathsf{T}^t \oplus \sum_{i \le \mathsf{m}^s, i \notin E} PPP^s_i \oplus \sum_{i \le \mathsf{m}^t, i \notin E} PPP^t_i \\ &= \mathsf{T}^s \oplus \mathsf{T}^t \oplus \sum_{i \le \mathsf{m}^s, i \notin E} PPP^s_i \oplus \sum_{i \le \mathsf{m}^t, i \notin E} PPP^{\mathsf{r}[t,i]}_i \\ &= \text{things-that-were-determined-before-query-}s \oplus \sum_{i \le \mathsf{m}^s, i \notin E} PPP^s_i \qquad (13) \end{aligned}$$

Since P^s is not a proper prefix of P^t, it follows that the set $D_s \stackrel{\text{def}}{=} \{1..\mathsf{m}^s\} - E$ is non-empty. And since query s is decipher, we can apply Claim 2 to conclude that $\sum_{i \in D} PPP^s_i = aMP^s + \beta$ where $a \neq 0$ and β depends only on things that were determined before MP^s. Combining this with Equation (13) we conclude that $MP^s \oplus MP^t = aMP^s + \beta'$ for the same non-zero constant a, where β' is a different expression, but it still depends only on things that were determined before MP^s. Therefore, $\Pr[MP^s = MP^t] = 2^{-n}$.

Next we analyze the cases where wither query s is encipher, $\mathsf{ty}^s = \mathsf{Enc}$, or P^s is a proper prefix of P^t. Recall that query t is encipher, so each PPP^t_i is either a free variable (if it is a "new block", $\mathsf{r}[t,i] = t$) or else it is identically set to equal $PPP^{\mathsf{r}[t,i]}_i$ (if $\mathsf{r}[t,i] < t$). And in the case where query s is encipher, then the same holds for each PPP^s_i. Either way, we can re-write Equation (12) as

$$MP^s \oplus MP^{s'} = \mathsf{T}^s \oplus \mathsf{T}^t \oplus \sum_{i \le \mathsf{m}^s, i \notin E} PPP^{\mathsf{r}[s,i]}_i \oplus \sum_{i \le \mathsf{m}^t, i \notin E} PPP^{\mathsf{r}[t,i]}_i \qquad (14)$$

(In the case that query s is decipher and P^s is a proper prefix of P^t, the equality follows since the summation on $i \le \mathsf{m}^s, i \notin E$ ranges over an empty set.) Recall that by definition we have $\mathsf{r}[s,i] = \mathsf{r}[t,i]$ if and only if $i \in E$. Let query r be "the last query that $MP^s \oplus MP^t$ depends on", and let I_s, I_t be the sets of indexes of PPP^s_i's and PPP^t_i's that "come from query r". That is, we define

$$\begin{aligned} &R \stackrel{\text{def}}{=} \{\mathsf{r}[s,i] \mid i \le \mathsf{m}^s,\ i \notin E\} \cup \{\mathsf{r}[t,i] \mid i \le \mathsf{m}^t,\ i \notin E\}, \qquad r \stackrel{\text{def}}{=} \max(R) \\ &\text{and then } I_s \stackrel{\text{def}}{=} \{\ i \le \mathsf{m}^s \mid i \notin E,\ \mathsf{r}[s,i] = r\ \},\ I_t \stackrel{\text{def}}{=} \{\ i \le \mathsf{m}^t \mid i \notin E,\ \mathsf{r}[t,i] = r\ \} \end{aligned}$$

From this definition it follows that the sets I_s, I_t are disjoint (since $\mathsf{r}[s,i] \neq \mathsf{r}[t,i]$ for $i \notin E$), and their union is non-empty (since R is non-empty). Using these notation we can rewrite Equation (14)

$$\begin{aligned} MP^s \oplus MP^t &= \mathsf{T}^s \oplus \mathsf{T}^t \oplus \left(\sum_{i \in I_s} PPP^{\mathsf{r}[s,i]}_i \oplus \sum_{i \le \mathsf{m}^s, i \notin (E \cup I_s)} PPP^{\mathsf{r}[s,i]}_i \right) \qquad (15) \\ &\qquad \oplus \left(\sum_{i \in I_t} PPP^{\mathsf{r}[t,i]}_i \oplus \sum_{i \le \mathsf{m}^t, i \notin (E \cup I_t)} PPP^{\mathsf{r}[t,i]}_i \right) \\ &= \text{things-that-were-determined-before-query-}r \oplus \sum_{i \in I_s \cup I_t} PPP^r_i \end{aligned}$$

If query r is decipher, $\mathsf{ty}^r = \mathsf{Dec}$, we can use Claim 2 to conclude that $\sum_{i \in I_s \cup I_t} PPP^r_i = aMP^r + \beta$ where $a \neq 0^n$ and β only depends on things that are determined before MP^r, and since MP^r is a free variable, it follows that $\Pr[MP^s = MP^t] = 2^{-n}$. If query r is encipher, $\mathsf{ty}^r = \mathsf{Enc}$, then all the variables PPP^r_i, $i \in I_s \cup I_t$, ar free variables, and again we have $\Pr[MP^s = MP^t] \le 2^{-n}$. ∎

Proof of Claim 1. All that is left now is to verify that Claims 4 through 7 above indeed cover all the possible types of collisions between $X, X' \in \mathfrak{D}$. So let $X, X' \in \mathfrak{D}$ be two distinct variables. We partition the analysis to four cases, depending on the "type" of the variable X.

$X =$ **"Zero".** Here X is the only variable in $\mathfrak{D}$ with $\phi(X) = \mathsf{none}$, so $\phi(X) \neq \phi(X')$. By Claim 4, we have $\Pr[X = X'] = 2^{-n}$.

$X =$ **"EZ" or** $X =$ **"PP_i^s".** In this case we have $\phi(X) = EZ$. If $\phi(X') \neq EZ$ then again we get $\Pr[X = X'] = 2^{-n}$ from Claim 4. On the other hand, if $\phi(X') = \phi(X) = EZ$ then Claim 5 gives us $\Pr[X = X'] \leq 2^{-n}$.

$X =$ **"CCC_i^s".** In this case $\phi(X) \in \{CCC_i^s, MC^s\}$. Again, if $\phi(X') \neq \phi(X)$ then we get the usual $\Pr[X = X'] = 2^{-n}$ from Claim 4. So assume that $\phi(X') = \phi(X) \in \{CCC_i^s, MC^s\}$. This means that X' is also of the form "CCC_j^t", so from Claim 6 we have $\Pr[X = X'] \leq 2^{-n}$.

$X =$ **"MP^s".** In this case $\phi(X) \in \{MP^r\colon\ r \leq q\} \cup \{PPP_i^r\colon\ r \leq q,\ i \leq \mathsf{m}^r\}$. As usual, if $\phi(X') \neq \phi(X)$ then we have $\Pr[X = X'] = 2^{-n}$ from Claim 4. So assume that $\phi(X') = \phi(X)$. This means that X' is also of the form "MP^t", so from Claim 7 we have $\Pr[X = X'] \leq 2^{-n}$.

This completes the proof of Claim 1. ▮