EME*: extending EME to handle arbitrary-length messages with associated data

(Preliminary Report)

Shai Halevi*

May 26, 2004

Abstract

This work describes a mode of operation, EME*, that turns a regular block cipher into a length-preserving enciphering scheme for messages of (almost) arbitrary length. Specifically, the resulting scheme can handle any bit-length, not shorter than the block size of the underlying cipher, and it also handles associated data of arbitrary bit-length. Such a scheme can either be used directly in applications that need encryption but cannot afford length expansion, or serve as a convenient building block for higher-level modes.

The mode EME* is a refinement of the EME mode of Halevi and Rogaway, and it inherits the efficiency and parallelism from the original EME.

1 Introductions

Adding secrecy protection to existing (legacy) protocols and applications raises some unique problems. One of these problems is that existing protocols sometimes require that the encryption be "transparent", and in particular preclude length-expansion. One example is encryption of storage data "at the sector level", where both the higher-level operating system and the lower-level disk expect the data to be stored in blocks of 512 bytes, and so any encryption method would have to accept 512-byte plaintext and produce 512-byte ciphertext.

Clearly, insisting on a length-preserving (and hence deterministic) transformation has many drawbacks. Indeed, even the weakest acceptable notion of "secure encryption" (i.e., semantic security [5]) cannot be achieved by deterministic encryption. Still, there may be cases where length-preservation is a hard requirement (due to technical, economical or even political constrains), and in such cases one may want to use some encryption scheme that gives better protection than no encryption at all. The strongest notions of security for a length-preserving transformation is "strong pseudo-random permutation" (SPRP) as defined by Luby and Rackoff [10], and its extension to "tweakable SPRP" by Liskov et al. [9]. A "tweak" is an additional input to the enciphering and deciphering procedures that need not be kept secret. This report uses the terms "tweak" and "associated data" pretty much interchangably, except that "associated data" hints that it can be of arbitrary length, wheras "tweak" is sometimes thought of as a fixed-length quantity.

Motivated by the application for "sector level encryption", some efficient modes of operation that implement "tweakable SPRP" on large blocks were recently described by Halevi and Rogaway

*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/

[6, 7]. As "general purpose modes", however, these modes are somewhat limited, in that they can only be applied to input messages whose size is a multiple of n, the block-size of the underlying cipher. Also, the mode CMC from [6] is inherently sequential (and it was only proven secure against attack model where all the messages are of the same length), and the mode EME from [7] is limited to messages of at most n^2 bits. The current work is aimed at eliminating these limitations.

The mode EME*, presented below, takes a standard cipher with n-bit blocks and turns it into a tweakable enciphering scheme with message space $\mathcal{M} = \{0,1\}^{n+}$ (i.e., any string of at least n bits) and tweak space $\mathcal{T} = \{0,1\}^*$. The key for EME* consists of one key of the underlying cipher and two additional n-bit blocks. The mode EME* has similar structure to the mode EME from [7]. Roughly, it consists of two layers of masked ECB encryption, with a layer of "lightweight mixing" in between. As a consequence, EME* is highly parallelizeable,[1] and also quite work-efficient. Processing an m-block query with ℓ blocks of associated data takes at most $\ell + 2m + \lceil m/n \rceil$ block encryptions (or decryptions). (We note that another mode for arbitrary-length messages, following the Luby-Rackoff approach, was recently proposed by McGrew and Viaga [11].)

1.1 What about very short blocks?

The mode EME* can handle blocks of any bit-length *but not less that the block size of the underlying cipher*. The underlying structure of EME*, being based on ECB encryption, does not lend itself to handling shorter blocks. In fact, in my opinion there is no good solution today for handling arbitrary short blocks. The solutions that I am aware of are the following:

- For blocks that are not too short (say, at least 64 bits), one can simply switch to using a different block cipher. For example, one could use EME*[AES] to process blocks that are 128 bits or more, and use a separately keyed EME*[3DES] to handle blocks of length between 64 and 127 bits.

  This solution, however, is quite expensive, as it mandates the implementation of two different ciphers. (Of course, one could use EME*[3DES] also to handle longer messages, but then the security parameter would be much reduced.) Moreover this solution does not address blocks shorter than 64 bits.

- For very short blocks (e.g., one byte) it is possible to pre-compute a pseudorandom permutation and store it in a table. This approach, however, clearly runs out of steam for blocks longer than two bytes, and it is extremely wasteful of space even before that. (Also, it is not clear how to incorporate a "tweak" into this approach.)

- Alternatively, one could apply the Luby-Rackoff construction to implement the narrow-block cipher, using the underlying cipher for the pseudorandom functions. (Indeed, the ABL mode of McGrew and Viaga [11] does just that.) This solution extends to handle messages of any length, but at a price of a severely reduced security-parameter. For example, although 128-bit blocks may enjoy "128 bits of security", 127-bit blocks only enjoy "63 bits of security". Even worse, 64-bit blocks have to make due with a pathetic "32 bits of security".

  It is possible to use six or more rounds of the Luby-Rackoff construction to make the security parameter a little less miserable (cf. Patarin's work [12]), but the price is an extremely slow mode for small blocks.

[1] In EME*, the longest execution path for any input consists of at most five block encryption. If the input length is a multiple of the block length then only longest path has only four encryptions, and only three if in addition the input is shorter than n blocks.

- Another approach is to use a parameterizable cipher (e.g., RC5 [13]) as the underlying block cipher. Parameterizable ciphers can be instantiated to handle various block sizes, so in particular they can be used in their narrow-block instantiation to handle the small blocks. However, to the best of my knowledge there is a fairly small number of such ciphers, and they were never seriously analyzed for small blocks. So it unlikely that they provide very good security, especially in the very small block sizes. Worse still, it is likely that using the same key for different block sizes would have disastrous consequences.

I view the problem of handling arbitrary small blocks as wide open. The two plausible approaches for addressing it are either to design a mode of operation with good security-performance tradeoff for small blocks, or to design an efficient block cipher that can handle small blocks securely. I believe that a good cipher is more likely to be possible than a good mode of operation (but perhaps this is only because I know more about modes of operation than about block ciphers.)

Organization

Section 2 recalls some standard definitions (this section is taken almost verbatim from [7]). Section 3 describes the EME* mode with a brief discussion of the extensions of EME* over EME. The security of EME* is stated in Section 4 and proven in the appendix.

Acknowledgments

I thank John Viaga for showing me his ABL mode of operation. I also thank Eli Biham for a discussion about the state of block ciphers for very short blocks.

2 Preliminaries

Basics. 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 $I\subseteq\mathbb{N}$) 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*. By Perm(n) we mean the set of all permutations on $\{0,1\}^n$. 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.

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 shows, in the brackets, an experiment to the left of the colon and an event to the right of the colon. We are looking at the probability of the indicated event after performing the specified experiment. By $X\xleftarrow{\$}\mathcal{X}$ we mean to choose X at random from the finite set $\mathcal{X}$. 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 $\mathbf{Adv}^{\text{xxx}}_{\Pi}(A)$ has been defined, we write $\mathbf{Adv}^{\text{xxx}}_{\Pi}(\mathcal{R})$ for the maximal value of $\mathbf{Adv}^{\text{xxx}}_{\Pi}(A)$ 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 $n \ge 1$ is a number). The query complexity σ_n is just the total number of n-bit blocks in all the queries that the adversary makes (including both the data and the associated data). Namely, the query complexity of any one call (T, P) is $\lceil |T|/n \rceil + \lceil |P|/n \rceil$, and the query complexity of an attack is the sum of the query complexity of all the calls. The name of an argument (e.g., t, q, 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 field 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 [3] 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 group 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* Mode

Consider a block cipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then $\text{EME}^*[E]$: $(\mathcal{K} \times \{0,1\}^{2n}) \times \mathcal{T} \times \mathcal{M} \to \mathcal{M}$ is an enciphering scheme with associated data, where $\mathcal{K}$ is the same as the underlying cipher, $\mathcal{T} = \{0,1\}^{0..n(2^n-3)}$, and $\mathcal{M} = \{0,1\}^{n..n(2^n-2)}$. In words, the key for $\text{EME}^*[E]$ consists of one key K of the underlying block cipher E and two n-bit blocks, L and R. $\text{EME}^*[E]$ accepts messages of any bit length grater than or equal to n (but no more than $n(2^n - 2)$), and associated data of arbitrary bit-length (but no more than $n(2^n - 3)$). Obviously, in paractical terms the upper limits are no limitation at all.

The scheme $\text{EME}^*[E]$ follows the same general principles of the tweakable scheme EME from [7]. Roughly, it consists of two layers of masked ECB encryption, with a layer of "lightweight mixing" in between. A complete specification of the enciphering scheme $\text{EME}^*[E]$ is given in Figure 1, and an illustration (for a message of $n + 2$ full blocks and one partial block) is provided in Figure 2. For those familiar with EME, the differences between EME and EME* are as follows:

- *Hashing the "tweak".* The original EME scheme requires that the "tweak value" be an n-bit string, whereas here we allow associated data of any length. For this purpose, we hash the

```
function $H_{K,R}(T_1 \cdots T_{\ell-1}, T_\ell)$:     // $|T_1| = \cdots = |T_{\ell-1}| = n$, $0 < |T_\ell| \le n$
01  if $T$ is empty return $E_K(R)$
10  for $i \in [1..\ell-1]$ do $TTT_i \leftarrow E_K(2^i R \oplus T_i)$
11  if $|T_\ell| = n$ then $TTT_\ell \leftarrow E_K(2^\ell R \oplus T_\ell)$
12  else $TTT_\ell \leftarrow E_K(2^{\ell+1} R \oplus (T_\ell 10..0))$
13  return $TTT_1 \oplus \cdots \oplus TTT_\ell$
```

```
Algorithm $\mathbf{E}_{K,L,R}(T; P_1 \cdots P_m)$
//   $|P_1| = \cdots = |P_{m-1}| = n$, $0 < |P_m| \le n$
101  if $|P_m| = n$ then $lastFull \leftarrow m$
102  else $lastFull \leftarrow m-1$
103          $PPP_m \leftarrow P_m$ padded with 10..0

110  for $i \leftarrow 1$ to $lastFull$ do
111          $PP_i \leftarrow 2^{i-1} L \oplus P_i$
112          $PPP_i \leftarrow E_K(PP_i)$

120  $SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
121  $MP_1 \leftarrow PPP_1 \oplus SP \oplus H_{K,R}(T)$
122  if $|P_m| = n$ then $MC_1 \leftarrow E_K(MP_1)$
123  else $MM \leftarrow E_K(MP_1)$
124          $MC_1 \leftarrow E_K(MM)$
125          $C_m \leftarrow P_m \oplus (MM$ truncated$)$
126          $CCC_m \leftarrow C_m$ padded with 10..0
127  $M_1 \leftarrow MP_1 \oplus MC_1$

130  for $i = 2$ to $lastFull$ do
131          $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132          if $k = 0$ then
133                  $MP_j \leftarrow PPP_i \oplus M_1$
134                  $MC_j \leftarrow E_K(MP_j)$
135                  $M_j \leftarrow MP_j \oplus MC_j$
136                  $CCC_i \leftarrow MC_j \oplus M_1$
137          else $CCC_i \leftarrow PPP_i \oplus 2^k M_j$

140  $SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
141  $CCC_1 \leftarrow MC_1 \oplus SC \oplus H_{K,R}(T)$
142  for $i \leftarrow 1$ to $lastFull$ do
143          $CC_i \leftarrow E_K(CCC_i)$
144          $C_i \leftarrow CC_i \oplus 2^{i-1} L$

150  return $C_1 \ldots C_m$
```

```
Algorithm $\mathbf{D}_{K,L,R}(T; C_1 \cdots C_m)$
//   $|C_1| = \cdots = |C_{m-1}| = n$, $0 < |C_m| \le n$
201  if $|C_m| = n$ then $lastFull \leftarrow m$
202  else $lastFull \leftarrow m-1$
203          $CCC_m \leftarrow C_m$ padded with 10..0

210  for $i \leftarrow 1$ to $lastFull$ do
211          $CC_i \leftarrow 2^{i-1} L \oplus C_i$
212          $CCC_i \leftarrow E_K^{-1}(CC_i)$

220  $SC \leftarrow CCC_2 \oplus \cdots \oplus CCC_m$
221  $MC_1 \leftarrow CCC_1 \oplus SC \oplus H_{K,R}(T)$
222  if $|C_m| = n$ then $MP_1 \leftarrow E_K^{-1}(MC_1)$
223  else $MM \leftarrow E_K^{-1}(MC_1)$
224          $MP_1 \leftarrow E_K^{-1}(MM)$
225          $P_m \leftarrow C_m \oplus (MM$ truncated$)$
226          $PPP_m \leftarrow P_m$ padded with 10..0
227  $M_1 \leftarrow MP_1 \oplus MC_1$

230  for $i = 2$ to $lastFull$ do
231          $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
232          if $k = 0$ then
233                  $MC_j \leftarrow CCC_i \oplus M_1$
234                  $MP_j \leftarrow E_K^{-1}(MC_j)$
235                  $M_j \leftarrow MP_j \oplus MC_j$
236                  $PPP_i \leftarrow MP_j \oplus M_1$
237          else $PPP_i \leftarrow CCC_i \oplus 2^k M_j$

240  $SP \leftarrow PPP_2 \oplus \cdots \oplus PPP_m$
241  $PPP_1 \leftarrow MP_1 \oplus SP \oplus H_{K,R}(T)$
242  for $i \leftarrow 1$ to $lastFull$ do
243          $PP_i \leftarrow E_K^{-1}(PPP_i)$
244          $P_i \leftarrow PP_i \oplus 2^{i-1} L$

250  return $P_1 \ldots P_m$
```

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

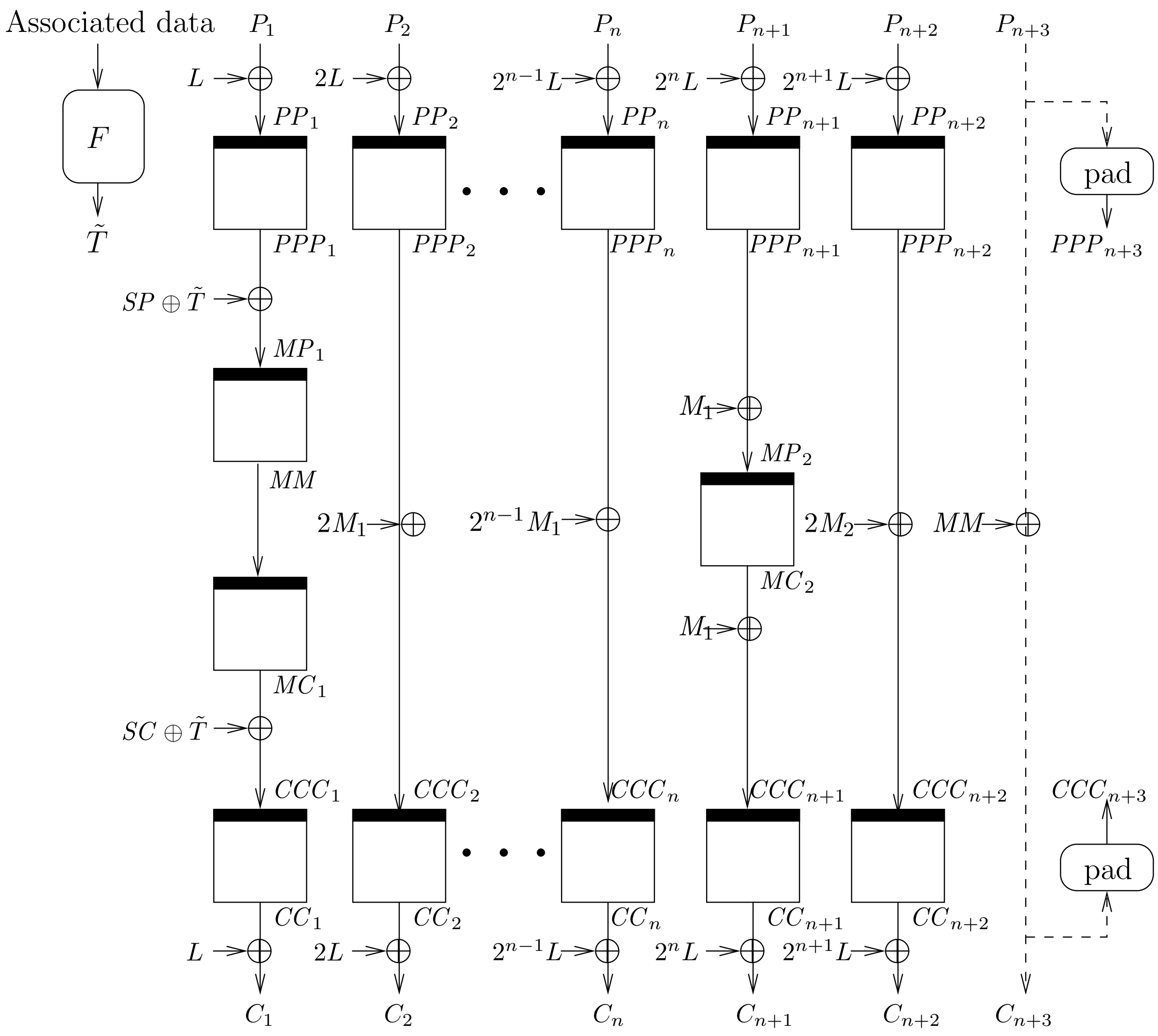

Figure 2: Enciphering under EME* a buffer with $n + 2$ full blocks and one partial block. The boxes represent E_K. We set the masks as $SP = PPP_2 \oplus \cdots PPP_{n+3}$, $M_i = MP_i \oplus MC_i$, and $SC = CCC_2 \oplus \cdots \oplus CCC_{n+3}$.

associated data to an n-bit string. The hash function need only be xor-universal, yet I chose to implement it using the underlying block cipher in a PMAC-like mode [2].

- *More than one mask.* The EME scheme uses (multiples of) a single mask value M in the "lightweight masking" layer. It was shown in [7], however, that this masking technique with just one mask cannot be used for messages longer than n^2 bits.

  Longer messages are handled in EME* using the approach that was proposed in the appendix of [7]. The message is broken to chunks of at most n^2 bits each, and a different mask value is used for every chunk. To handle the last partial block (if any), yet another mask is computed and xor-ed into the last partial plaintext block, thus getting the last partial ciphertext block.

We comment that it is possible to derive the two key blocks L, R from the cipher key K, say by setting $L = 2E_K(0)$ and $R = 3E_K(0)$.[2] The proof below does not prove this variant, since proving it would mean adding a few more pages to a proof that is already way too long.

4 Security of EME*

The following theorem relates the advantage of an adversary in attacking EME$^*[E]$ to the advantage an adversary in attacking the block cipher E.

Theorem 1 [EME* security] Any adversary that tries to distinguish EME*[Perm(n)] from a truly random tweakable length-preserving permutation, using at most q queries totaling at most σ_n blocks (some of which may be partial), has advantage at most $(2.5\sigma_n + 3q)^2/2^{n+1}$. Using the notations from Section 2, we have

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME}^*[\mathrm{Perm}(n)]}(q, \sigma_n) \le \frac{(2.5\sigma_n + 3q)^2}{2^{n+1}} \tag{1}$$

Corollary 1 Fix $n, t, q, \sigma_n \in \mathbb{N}$ and a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then

$$\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME}^*[E]}(t, q, \sigma_n) \le \frac{(2.5\sigma_n + 3q)^2}{2^{n+1}} \; + \; 2\,\mathbf{Adv}^{\pm\mathrm{prp}}_{E}\left(t',\; 2q + (2 + \frac{1}{n})\sigma_n\right)$$

where $t' = t + O(n\sigma_n)$. □

Note that the theorem and corollary *do not* restrict messages to one particular length: proven security is for a variable-input-length (VIL) cipher, not just fixed-input-length (FIL) one. The proof of Theorem 1 is given in Appendix A. Corollary 1 embodies the standard way to pass from the information-theoretic setting to the complexity-theoretic one.

References

[1] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 197–215. Springer-Verlag, 2000.

[2]The maximum length of messages and associated input would have to be somewhat reduced for this to work. But for $n = 128$ we can still prove security for messages and associated data that are shorter than, say, 2^{120} blocks. (The upper bound is actually $2^n - 1 - \log_2 3$. With the representation of $FG(2^{128})$ as above, we have $\log_2 3 \approx 3.39 \times 10^{38} \approx 2^{128} - 2^{120}$. See [14].)

[2] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer-Verlag, 2002.

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

[4] S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, 1997.

[5] S. Goldwasser and S. Micali. "Probabilistic encryption". *J. of Computer and System Sciences*, 28, April 1984.

[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*, pages 482–499. Springer-Verlag, 2003. Full version available on the ePrint archive, http://eprint.iacr.org/2003/148/.

[7] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In *The RSA conference – Cryptographer's track, RSA-CT'04*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer-Velrag, 2004. Full version available on the ePrint archive, http://eprint.iacr.org/2003/147/.

[8] 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.

[9] M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, 2002. www.cs.berkeley.edu/∼daw/.

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

[11] D. A. McGrew and J. Viega. ABL mode: security without data expansion. Private communication, 2004.

[12] J. Patarin. Luby-Rackoff: 7 rounds are enough for $2^{n(1-\varepsilon)}$ security. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 513–529. Springer-Verlag, 2003.

[13] R. L. Rivest. The RC5 encryption algorithm. In *Fast Software Encryption (FSE '94)*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer, 1994.

[14] P. Rogaway. Efficient instantiations of tweakable block ciphers and refinements to modes OCB and PMAC. Available on-line from `http://www.cs.ucdavis.edu/∼rogaway/papers/`, 2004.

A Proof of Theorem 1 — Security of EME*

A personal comment. The proof below spans more than 23 pages, and as much as I tried to simplify and to explain clearly, it is quite a pain to read. Frankly, I don't believe that anyone will ever go through the trouble of reading and verifying it. Assuming this is the case, one can still get some assurance in the correctness of the mode, even from a proof that no one reads: At least it implies that the author went carefully through all the different cases and was convinced that they all work. Indeed, the proof below uses the same mechanism that was used to prove CMC [6] and EME [7], and this mechanism in effect forces one to cover all the cases. Also, the mode EME* is close enough to the original mode EME, so that one who verified the proof for EME (which is shorter) may be able to be convinced of the correctness of EME* just "by inspection".

A useful lemma. The proof of security is divided into two parts: in Section A.1 we carry out a game-substitution argument, reducing the analysis of EME* to the analysis of a simpler probabilistic game. In Section A.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, $\mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{rnd}}}$, by

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

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 $q \geq 1$ be a number. Then $|\mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{prp}}}(q) - \mathbf{Adv}_{\mathbf{E}}^{\pm\widetilde{\mathrm{rnd}}}(q)| \leq q(q-1)/2^{N+1}$ where N is the length of a shortest string in the message space $\mathcal{M}$. $\square$

A.1 The game-substitution sequence

Fix n, σ_n, and q. Let A be an adversary that asks q oracle queries (none pointless) totaling σ_n blocks (of both data and associated data, potentially some of them partial blocks). Our goal in this part is to tie the advantage $\mathbf{Adv}_{\mathrm{EME}[\mathrm{Perm}(n)]}^{\pm\widetilde{\mathrm{rnd}}}(A)$ to the probability $\Pr[\,\mathrm{N2} \text{ sets } \mathit{bad}\,]$, where N2 is some probability space and "N2 sets *bad*" is an event defined there. Later we bound $\Pr[\,\mathrm{N2} \text{ sets } \mathit{bad}]$, and, putting that together with Lemma 2, we get Eq. (1) of Theorem 1. Game N2 is obtained by a game-substitution argument, as carried out in works like [8]. 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.

Abstracting the function $H_{K,R}$: The analysis below turns out to be quite complicated. We somewhat simplify it by replacing the function $H_{K,R}$ by an abstract function $h : \{0,1\}^* \to \{0,1\}^n$, chosen from a pairwise independent family $\mathcal{H}$. The properties of h that we use in the analysis are:

(i) For a fixed $\mathsf{T} \in \{0,1\}^*$, $h(\mathsf{T})$ is uniform in $\{0,1\}^n$ when h is chosen at random from $\mathcal{H}$.

(ii) For fixed $\mathsf{T} \neq \mathsf{T}' \in \{0,1\}^*$, $h(\mathsf{T}) \oplus h(\mathsf{T}')$ is uniform in $\{0,1\}^n$ when $h \xleftarrow{\$} \mathcal{H}$.

(iii) The choice $h \xleftarrow{\$} \mathcal{H}$ is independent of all the other random choices in the game.

We can justify these assumptions on h by replacing the computation of $E_K(T \oplus jR)$ (with j a constant) in lines 10, 11, and 12 of Figure 1, by the computation $f_j(T)$ where for each j we have an independent random function $f_j : \{0,1\}^n \to \{0,1\}^n$. It is known that replacing a masked random permutation by a collection of random functions this way entails only a negligible difference on the view of the adversary. Specifically, one could prove the following: Fix some integers $n, q_p, q_f \in \mathbb{N}$ and an adversary with three oracles $A^{E(\cdot),D(\cdot),F(\cdot,\cdot)}$, and consider the two following experiments.

- In the first experiment (Expr1), we choose at random a permutation π over $\{0,1\}^n$ and a string $R \in \{0,1\}^n$. Then for $x, y, j \in \{0,1\}^n$, an oracle-query $E(x)$ is answered by $\pi(x)$, an oracle query $D(y)$ is answered by $\pi^{-1}(y)$, and an oracle query $F(j,x)$ is answered by $\pi(x \oplus jR)$ (where the multiplication jR is over $GF(2^n)$).
- In the second experiment (Expr2), we choose at random a permutation π over $\{0,1\}^n$, and 2^n functions $\{f_j : \{0,1\}^n \to \{0,1\}^n\}_{j \in \{0,1\}^n}$. Then for $x, y, j \in \{0,1\}^n$, the oracle-queries $E(x)$ and $D(y)$ are answered as before by $\pi(x)$ and $\pi^{-1}(y)$, respectively, but an oracle query $F(j,x)$ is answered by $f_j(x)$.

Lemma 3 Fix some $n, q_p, q_f \in \mathbb{N}$. For any adversary $A^{E(\cdot),D(\cdot),F(\cdot,\cdot)}$ as above that makes at most q_p queries to E and D, and at most q_f queries to F, it holds that

$$\left| \Pr_{\text{Expr1}}[\, A^{E,D,F} \Rightarrow 1\,] \;-\; \Pr_{\text{Expr2}}[\, A^{E,D,F} \Rightarrow 1\,] \right| \;\le\; q_f(q_f + 2q_p)/2^n \qquad \square$$

This lemma is pretty much folklore by now, although I could not find a reference where it is proven. Similar results were proven by by Even and Mansour [4]. A proof for a special case of this lemma can be found in [1, Lemma 4], and that proof can easily be extended to prove Lemma 3 itself.

Using Lemma 3, we can replace the function $H_{K,R}$ from Figure 1 by the following function h (that depends on the 2^n random functions f_j). In the code below, the constants 2^i are computed in the finite field $GF(2^n)$.

```
function $h(T_1 \cdots T_{\ell-1}, T_\ell)$:    // $|T_1| = \cdots = |T_{\ell-1}| = n$, $0 < |T_\ell| \le n$
01 if $T$ is empty return $f_1(0)$
10 for $i \in [1..\ell-1]$ do $TTT_i \leftarrow f_{2^i}(T_i)$
11 if $|T_\ell| = n$ then $TTT_\ell \leftarrow f_{2^\ell}(T_\ell)$
12 else $TTT_\ell \leftarrow f_{2^{\ell+1}}(T_\ell 10..0))$
13 return $TTT_1 \oplus \cdots \oplus TTT_\ell$
```

Divide the total number of blocks σ_n in an attack on EME* into $\sigma_n = \sigma_n^d + \sigma_n^a$ where σ_n^d is the number of blocks in the data itself, and σ_n^a is the number of blocks in the associated data. Let N_{be} denote the *total number of block encryptions* that are used throughout the attack (not counting the computation of H), and we can bound it by

$$N_{\text{be}} < (2 + \frac{1}{n})\sigma_n^d + 2q \tag{2}$$

Then from Lemma 3 it follows that the statistical distance in the view of the adversary due to the replacement of $H_{K,R}$ by h is bounded by $\sigma_n^a(\sigma_n^a + 2N_{\text{be}})/2^n$. Once we made that replacement, it is clear that the choice of h is now independent of all the other random choices in the attack, so we only need to prove the properties (i) and (ii). This is done next:

```
Subroutine Choose-$\pi(X)$:
010  $Y \xleftarrow{\$} \{0,1\}^n$;  if $Y \in$ Range then $bad \leftarrow$ true , $Y \xleftarrow{\$} \overline{\text{Range}}$
011  if $X \in$ Domain then $bad \leftarrow$ true , $Y \leftarrow \pi(X)$
012  $\pi(X) \leftarrow Y$,  Domain $\leftarrow$ Domain $\cup \{X\}$,  Range $\leftarrow$ Range $\cup \{Y\}$;  return $Y$
Subroutine Choose-$\pi^{-1}(Y)$:
020  $X \xleftarrow{\$} \{0,1\}^n$;  if $X \in$ Domain then $bad \leftarrow$ true , $X \xleftarrow{\$} \overline{\text{Domain}}$
021  if $Y \in$ Range then $bad \leftarrow$ true , $X \leftarrow \pi^{-1}(Y)$
022  $\pi(X) \leftarrow Y$,  Domain $\leftarrow$ Domain $\cup \{X\}$,  Range $\leftarrow$ Range $\cup \{Y\}$;  return $X$
```

Figure 3: The procedures that are used in games E1 and R1. The shaded statements are executed in Game E1 but not in Game R1.

Claim 2 When 2^n functions $\{f_j : \{0,1\}^n \to \{0,1\}^n\}_{j\in\{0,1\}^n}$ are chosen at random and h is defined as above, it holds that:
(i) For any fixed $T \in \{0,1\}^{0..n(2^n-3)}$, $h(T)$ is uniform in $\{0,1\}^n$.
(ii) For any fixed $T \neq T' \in \{0,1\}^{0..n(2^n-3)}$, $h(T) \oplus h(T')$ is uniform in $\{0,1\}^n$.

Proof: Property (i) is obvious, since the output of h at any point T depend on at least one application of one of the functions f_j, and these are all random functions. To prove Property (ii), fix some $T \neq T'$, and denote $T = T_1 \ldots T_\ell$ and similarly $T = T_1 \ldots T_{\ell'}$, where $\ell = \lceil |T|/n \rceil$ and $\ell' = \lceil |T'|/n \rceil$. (The proof below use the fact that 2 is a primitive element in $GF(2^n)$ and $\ell' \leq 2^n - 3$, so for any $i \neq i' \leq \ell' + 1$ we have $2^i \neq 2^{i'}$ in $GF(2^n)$.)

If $\ell = \ell'$ then there must be at least one index $i \leq \ell$ such that $T_i \neq T'_i$. If T_i and T'_i are full blocks then $h(T) \oplus h(T') =$ something-independent-of-$f_{2^i} \oplus f_{2^i}(T_i) \oplus f_{2^i}(T'_i)$, which is uniform since f_{2^i} is a random function. If they are both partial blocks (so $i = \ell$) then we get $h(T) \oplus h(T') =$ something-independent-of-$f_{2^{\ell+1}} \oplus f_{2^{\ell+1}}(T_i 10..0) \oplus f_{2^{\ell+1}}(T'_i 10..0)$, which is again uniform since $T_i \neq T'_i$ implies that also $T_i 10..0 \neq T'_i 10..0$ and $f_{2^{\ell+1}}$ is a random function. If T_i is a full block and T'_i is partial, then we similarly get $h(T) \oplus h(T') =$ something-independent-of-$f_{2^{\ell+1}} \oplus f_{2^{\ell+1}}(T'_i 10..0)$.

If $\ell \neq \ell'$, then assume that $\ell' > \ell$. If T'_i is a partial block then as before we get $h(T) \oplus h(T') =$ something-independent-of-$f_{2^{\ell'+1}} \oplus f_{2^{\ell'+1}}(T'_i 10..0)$. Similarly if T'_i is a full block and either $\ell' > \ell + 1$ or T_ℓ is a full block, then $h(T) \oplus h(T') =$ something-independent-of-$f_{2^{\ell'}} \oplus f_{2^{\ell'}}(T'_i)$. The last case is when $\ell' = \ell + 1$ and $T'_{\ell'}$ is a full block and T_ℓ is a partial block. In this case $h(T')$ includes the term $f_{2^\ell}(T'_\ell)$ but $h(T)$ is independent of f_{2^ℓ}, so again $h(T) \oplus h(T')$ is uniform. ▌

The game E1. We describe the attack scenario of A against EME[Perm(n)] (with the abstraction of h as above) 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 π: $\{0,1\}^n \to \{0,1\}^n$ is everywhere undefined. When we need $\pi(X)$ and π isn't yet defined at X we choose this value randomly among the available range values. When we need $\pi^{-1}(Y)$ and there is no X for which $\pi(X)$ has been set to Y we likewise choose X at random from the available domain values. As we fill in π its domain and its range thus grow. In the game we keep track of the domain and range of π by maintaining two sets, Domain and Range, that include all the points for which π is already defined. We let $\overline{\text{Domain}}$ and $\overline{\text{Range}}$ be the complement of these sets relative to $\{0,1\}^n$. The game, denoted E1, is shown in Figures 3 and 4. Since game E1 accurately represent the attack scenario, we have that

$$\Pr[\, A^{\mathbf{E}_\pi\, \mathbf{D}_\pi} \Rightarrow 1\,] \;\; \leq \;\; \Pr[\, A^{\text{E1}} \Rightarrow 1\,] + \frac{\sigma_n^a(\sigma_n^a + 2N_{\text{be}})}{2^n} \qquad (3)$$

```
**Initialization:**
050  Domain ← Range ← $\emptyset$;  **for** all $X \in \{0,1\}^n$ **do** $\pi(X) \leftarrow$ undef
051  $bad \leftarrow$ false;  $L \xleftarrow{\$} \{0,1\}^n$;  $h \leftarrow \mathcal{H}$

**Respond to the $s$-th adversary query as follows:**
```

```
AN ENCIPHER QUERY, $\mathsf{Enc}(T^s; P_1^s \cdots P_{m^s}^s)$:
102  **if** $|P_{m^s}^s| = n$ **then** $lastFull^s \leftarrow m^s$
103  **else** $lastFull^s \leftarrow m^s - 1$
104        $PPP_{m^s}^s \leftarrow P_{m^s}^s$ padded with 10..0

110  **for** $i \leftarrow 1$ **to** $lastFull^s$ **do**
111        $r = r[s,i]$ is the 1st index s.t. $P_i^s = P_i^r$
112        **if** $r < s$ **then** $PP_i^s \leftarrow PP_i^r$
113              $PPP_i^s \leftarrow PPP_i^r$
114        **else** $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L$
115              $PPP_i^s \leftarrow$ Choose-$\pi(PP_i^s)$

120  $MP_1^s \leftarrow PPP_1^s \oplus \cdots \oplus PPP_{m^s}^s \oplus h(T^s)$
121  **if** $|P_{m^s}^s| = n$ **then** $MC_1^s \leftarrow$ Choose-$\pi(MP_1^s)$
122  **else** $MM^s \leftarrow$ Choose-$\pi(MP_1^s)$
123        $MC_1^s \leftarrow$ Choose-$\pi(MM^s)$
124        $C_{m^s}^s \leftarrow P_{m^s}^s \oplus (MM^s$ truncated$)$
125        $CCC_{m^s}^s \leftarrow C_{m^s}^s$ padded with 10..0
126  $M_1^s \leftarrow MP_1^s \oplus MC_1^s$

130  **for** $i \leftarrow 2$ **to** $lastFull^s$ **do**
131        $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132        **if** $k = 0$ **then**
133              $MP_j^s \leftarrow PPP_i^s \oplus M_1^s$
134              $MC_j^s \leftarrow$ Choose-$\pi(MP_j^s)$
135              $M_j^s \leftarrow MP_j^s \oplus MC_j^s$
136              $CCC_i^s \leftarrow MC_j^s \oplus M_1^s$
137        **else** $CCC_i^s \leftarrow PPP_i^s \oplus 2^k M_j^s$
138  $CCC_1^s \leftarrow MC_1^s \oplus CCC_2^s \oplus \cdots CCC_{m^s}^s \oplus h(T^s)$

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

150  **return** $C_1^s \cdots C_{m^s}^s$
```

```
A DECIPHER QUERY, $\mathsf{Dec}(T^s; C_1^s \cdots C_{m^s}^s)$:
202  **if** $|C_{m^s}^s| = n$ **then** $lastFull^s \leftarrow m^s$
203  **else** $lastFull^s \leftarrow m^s - 1$
204        $CCC_{m^s}^s \leftarrow C_{m^s}^s$ padded with 10..0

210  **for** $i \leftarrow 1$ **to** $lastFull^s$ **do**
211        $r = r[s,i]$ is the 1st index s.t. $C_i^s = C_i^r$
212        **if** $r < s$ **then** $CC_i^s \leftarrow CC_i^r$
213              $CCC_i^s \leftarrow CCC_i^r$
214        **else** $CC_i^s \leftarrow C_i^s \oplus 2^{i-1}L$
215              $CCC_i^s \leftarrow$ Choose-$\pi^{-1}(CC_i^s)$

220  $MC_1^s \leftarrow CCC_1^s \oplus \cdots \oplus CCC_{m^s}^s \oplus h(T^s)$
221  **if** $|C_{m^s}^s| = n$ **then** $MP_1^s \leftarrow$ Choose-$\pi^{-1}(MC_1^s)$
222  **else** $MM^s \leftarrow$ Choose-$\pi^{-1}(MC_1^s)$
223        $MP_1^s \leftarrow$ Choose-$\pi^{-1}(MM^s)$
224        $P_{m^s}^s \leftarrow C_{m^s}^s \oplus (MM^s$ truncated$)$
225        $PPP_{m^s}^s \leftarrow P_{m^s}^s$ padded with 10..0
226  $M_1^s \leftarrow MP_1^s \oplus MC_1^s$

230  **for** $i \leftarrow 2$ **to** $lastFull^s$ **do**
231        $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
232        **if** $k = 0$ **then**
233              $MC_j^s \leftarrow CCC_i^s \oplus M_1^s$
234              $MP_j^s \leftarrow$ Choose-$\pi^{-1}(MC_j^s)$
235              $M_j^s \leftarrow MP_j^s \oplus MC_j^s$
236              $PPP_i^s \leftarrow MP_j^s \oplus M_1^s$
237        **else** $PPP_i^s \leftarrow CCC_i^s \oplus 2^k M_j^s$
238  $PPP_1^s \leftarrow MP_1^s \oplus PPP_2^s \oplus \cdots PPP_{m^s}^s \oplus h(T^s)$

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

250  **return** $P_1^s \cdots P_{m^s}^s$
```

Figure 4: Game E1 describes the attack of A on $\mathrm{EME}[\mathrm{Perm}(n)]$, where the permutation π is chosen "on the fly" as needed. Game R1 is the same as game E1, except we do not execute the shaded statements in the procedures from Figure 3.

(where the additive factor is due to the abstraction of h). Looking ahead to the game-substitution sequence, we structured the code in Figures 3 and 4 in a way that makes it easier to present the following games. In particular, here are some things to note about this code:

- *Notations.* We denote all the quantities that are encountered during the processing of query s with a superscript s. For example, the number of blocks in the query is denoted m^s, and the plaintext is denoted $P^s = P_1^s \cdots P_{m^s}^s$ (where $|P_i^s| = n$ for $i < m^s$ and $|P_{m^s}^s| \leq n$).

- *The notation* $r[s,i]$. When handling the s-th adversary query, we look for each block of the query to see if it is a "new block": if this is an encipher query $P^s = (P_1^s \cdots P_{m^s}^s)$ we look for an earlier plaintext $P^r = (P_1^r \cdots P_{m^r}^r)$ with the same i'th block $P_i^s = P_i^r$. Since we use "masked ECB" encryption, we only expect to choose a new value for π when there is no such prior plaintext. If this is a decipher query then for any i we likewise look for an earlier ciphertext C^r with the same i'th block, $C_i^s = C_i^r$. We define $r[s,i]$ to be the index of the first such plaintext or ciphertext. Namely, we define

$$r[s,i] \stackrel{\text{def}}{=} \begin{cases} \min\{\, r \leq s \;:\; P_i^r = P_i^s\} & \text{if query } s \text{ is an encipher query} \\ \min\{\, r \leq s \;:\; C_i^r = C_i^s\} & \text{if query } s \text{ is a decipher query} \end{cases}$$

- *Filling in* π *and* π^{-1} *values.* When we need to define π on what is likely to be a new domain point X, setting $\pi(X) \leftarrow Y$ for some Y, we do the following: We first sample Y from $\{0,1\}^n$; then *re-sample*, this time from $\overline{\text{Range}}$, if the initially chosen sample Y was already in the range of π; finally, if π already had a value at X, then we forget about the newly chosen value Y and use the previous value of $\pi(X)$. We behave analogously for $\pi^{-1}(Y)$ values. In Figure 3 we highlight the places where we have to reset a choice we tentatively made. Whenever we do so we set a flag *bad*. The flag *bad* is never seen by the adversary A that interacts with the E1 game—it is only present to facilitate the subsequent analysis.

Game R1. We next modify game E1 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. The game R1 is described in Figure 5. (In this figure we omitted the function π from the code, since it is never used anymore.)

These changes mean that π may end up not being a permutation, and moreover we may reset its value on previously chosen points. Still, the games E1 and R1 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

$$\Pr[\, A^{\text{E1}} \Rightarrow 1\,] - \Pr[\, A^{\text{R1}} \Rightarrow 1\,] \;\;\leq\;\; \Pr[\, A^{\text{R1}} \text{ sets } \mathit{bad}\,] \qquad (4)$$

Initialization:

```
050 Domain $\leftarrow$ Range $\leftarrow \emptyset$;  $bad \leftarrow$ false;  $L \xleftarrow{\$} \{0,1\}^n$;  $h \leftarrow \mathcal{H}$
```

Respond to the s-th adversary query as follows:

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

```
101 if $|P_{m^s}^s| = n$ then $lastFull^s \leftarrow m^s$
102 else $lastFull^s \leftarrow m^s - 1$
103     $PPP_{m^s}^s \leftarrow P_{m^s}^s$ padded with 10..0

110 for $i \leftarrow 1$ to $lastFull^s$ do
111     $r = r[s,i]$ is the 1st index s.t. $P_i^s = P_i^r$
112     if $r < s$ then $PP_i^s \leftarrow PP_i^r$;  $PPP_i^s \leftarrow PPP_i^r$
113     else $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L$;  $PPP_i^s \xleftarrow{\$} \{0,1\}^n$
114         if $PP_i^s \in$ Domain or $PPP_i^s \in$ Range then $bad \leftarrow$ true
115         Domain $\leftarrow$ Domain $\cup \{PP_i^s\}$;  Range $\leftarrow$ Range $\cup \{PPP_i^s\}$

120 $MP_1^s \leftarrow PPP_1^s \oplus \cdots \oplus PPP_{m^s}^s \oplus h(T^s)$
121 if $|P_{m^s}^s| = n$ then $MC_1^s \xleftarrow{\$} \{0,1\}^n$;  $M_1^s \leftarrow MP_1^s \oplus MC_1^s$
122     if $MP_1^s \in$ Domain or $MC_1^s \in$ Range then $bad \leftarrow$ true
123     Domain $\leftarrow$ Domain $\cup \{MP_1^s\}$;  Range $\leftarrow$ Range $\cup \{MC_1^s\}$
124 else $MM^s \xleftarrow{\$} \{0,1\}^n$ ; $MC_1^s \xleftarrow{\$} \{0,1\}^n$;  $M_1^s \leftarrow MP_1^s \oplus MC_1^s$
125     if $MP_1^s \in$ Domain or $MM^s \in$ Range then $bad \leftarrow$ true
126     if $MM^s \in$ Domain $\cup \{MP_1^s\}$ or $MC_1^s \in$ Range $\cup \{MM^s\}$ then $bad \leftarrow$ true
127     Domain $\leftarrow$ Domain $\cup \{MP_1^s, MM^s\}$;  Range $\leftarrow$ Range $\cup \{MM^s, MC_1^s\}$
128     $C_{m^s}^s \leftarrow P_{m^s}^s \oplus (MM^s$ truncated);  $CCC_{m^s}^s \leftarrow C_{m^s}^s$ padded with 10..0

130 for $i \leftarrow 2$ to $lastFull^s$ do
131     $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132     if $k = 0$ then
133         $MP_j^s \leftarrow PPP_i^s \oplus M_1^s$;  $MC_j^s \xleftarrow{\$} \{0,1\}^n$;  $M_j^s \leftarrow MP_j^s \oplus MC_j^s$
134         if $MP_j^s \in$ Domain or $MC_j^s \in$ Range then $bad \leftarrow$ true
135         Domain $\leftarrow$ Domain $\cup \{MP_j^s\}$;  Range $\leftarrow$ Range $\cup \{MC_j^s\}$
136         $CCC_i^s \leftarrow MC_j^s \oplus M_1^s$
137     else $CCC_i^s \leftarrow PPP_i^s \oplus 2^k M_j^s$
138 $CCC_1^s \leftarrow MC_1^s \oplus CCC_2^s \oplus \cdots CCC_{m^s}^s \oplus h(T^s)$

140 for $i \leftarrow 1$ to $lastFull^s$ do
141     $CC_i^s \xleftarrow{\$} \{0,1\}^n$;  $C_i^s \leftarrow CC_i^s \oplus 2^{i-1}L$
142     if $CCC_i^s \in$ Domain or $CC_i^s \in$ Range then $bad \leftarrow$ true
143     Domain $\leftarrow$ Domain $\cup \{CCC_i^s\}$;  Range $\leftarrow$ Range $\cup \{CC_i^s\}$

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

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

Figure 5: Game R1 is similar to E1, but does not reset the random choices.

```
Initialization:
050  Domain $\leftarrow$ Range $\leftarrow \emptyset$;  $bad \leftarrow$ false;  $L \xleftarrow{\$} \{0,1\}^n$;  $h \leftarrow \mathcal{H}$

Respond to the $s$-th adversary query as follows:
AN ENCIPHER QUERY, $\mathsf{Enc}(T^s; P^s_1 \cdots P^s_{m^s})$:
101  if $|P^s_{m^s}| = n$ then $lastFull^s \leftarrow m^s$
102  else $lastFull^s \leftarrow m^s - 1$
103      $PPP^s_{m^s} \leftarrow P^s_{m^s}$ padded with 10..0

110  for $i \leftarrow 1$ to $lastFull^s$ do
111      $r = r[s,i]$ is the 1st index s.t. $P^s_i = P^r_i$
112      if $r < s$ then $PP^s_i \leftarrow PP^r_i$;  $PPP^s_i \leftarrow PPP^r_i$
113      else $PP^s_i \leftarrow P^s_i \oplus 2^{i-1}L$;  $PPP^s_i \xleftarrow{\$} \{0,1\}^n$
114          if $PP^s_i \in$ Domain or $PPP^s_i \in$ Range then $bad \leftarrow$ true
115          Domain $\leftarrow$ Domain $\cup \{PP^s_i\}$;  Range $\leftarrow$ Range $\cup \{PPP^s_i\}$

120  $C^s_* \xleftarrow{\$} \{0,1\}^n$;  $M^s_1 \xleftarrow{\$} \{0,1\}^n$
121  $MP^s_1 \leftarrow PPP^s_1 \oplus \cdots \oplus PPP^s_{m^s} \oplus h(T^s)$;  $MC^s_1 \leftarrow MP^s_1 \oplus M^s_1$;  $MM^s \leftarrow PPP^s_{m^s} \oplus C^s_*$
122  if $MP^s_1 \in$ Domain or $MM^s \in$ Range then $bad \leftarrow$ true
123  if $MM^s \in$ Domain $\cup \{MP^s_1\}$ or $MC^s_1 \in$ Range $\cup \{MM^s\}$ then $bad \leftarrow$ true
124  Domain $\leftarrow$ Domain $\cup \{MP^s_1, MM^s\}$;  Range $\leftarrow$ Range $\cup \{MM^s, MC^s_1\}$
125  if $|P^s_{m^s}| = n$ then
126      $C^s_{m^s} \leftarrow (C^s_*$ truncated$)$;  $CCC^s_{m^s} \leftarrow C^s_{m^s}$ padded with 10..0

130  for $i \leftarrow 2$ to $lastFull^s$ do
131      $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132      if $k = 0$ then
133          $MP^s_j \leftarrow PPP^s_i \oplus M^s_1$;  $M^s_j \xleftarrow{\$} \{0,1\}^n$;  $MC^s_j \leftarrow MP^s_j \oplus M^s_j$
134          if $MP^s_j \in$ Domain or $MC^s_j \in$ Range then $bad \leftarrow$ true
135          Domain $\leftarrow$ Domain $\cup \{MP^s_j\}$;  Range $\leftarrow$ Range $\cup \{MC^s_j\}$
136      $CCC^s_i \leftarrow PPP^s_i \oplus 2^k M^s_j$
137  $CCC^s_1 \leftarrow PPP^s_1 \oplus M^s_1 \oplus (PPP^s_2 \oplus CCC^s_2) \oplus \cdots \oplus (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$

140  for $i \leftarrow 1$ to $lastFull^s$ do
141      $C^s_i \xleftarrow{\$} \{0,1\}^n$;  $CC^s_i \leftarrow C^s_i \oplus 2^{i-1}L$
142      if $CCC^s_i \in$ Domain or $CC^s_i \in$ Range then $bad \leftarrow$ true
143      Domain $\leftarrow$ Domain $\cup \{CCC^s_i\}$;  Range $\leftarrow$ Range $\cup \{CC^s_i\}$

150  return $C^s_1 \cdots C^s_{m^s}$

A DECIPHER QUERY, $\mathsf{Dec}(T^s; C^s_1 \cdots C^s_{m^s})$, IS TREATED SYMMETRICALLY
```

Figure 6: Game R2 is indistinguishable from Game R1 but chooses some of its variables in different order.

Game R2. We now make several changes to the order in which variables are chosen in game R1. Specifically, we make the following changes to the code:

- 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 line 141), we choose $C_i^s \xleftarrow{\$} \{0,1\}^n$ and then set $CC_i^s \leftarrow C_i^s \oplus 2^i L$.
- Similarly, instead of choosing $MC_j^s \xleftarrow{\$} \{0,1\}^n$ and setting $M_j^s \leftarrow MP_j^s \oplus MC_j^s$ (lines 121, 124 and 133), we choose $M_j^s \xleftarrow{\$} \{0,1\}^n$ and set $MC_j^s \leftarrow MP_j^s \oplus M_j^s$.
- Instead of choosing $MM \xleftarrow{\$} \{0,1\}^n$ and setting $C_{m^s}^s \leftarrow P_{m^s}^s \oplus (MM^s$ truncated) (lines 124 and 128) we choose $C_*^s \xleftarrow{\$} \{0,1\}^n$ and set $C_{m^s}^s \leftarrow (C_*^s$ truncated) and $MM^s \leftarrow (P_{m^s}^s 10..0) \oplus C_*^s$.
- We replace the assignment $CCC_i^s \leftarrow MC_j^s \oplus M_1^s$ in line 136 by the equivalent assignment $CCC_i^s \leftarrow PPP_i^s \oplus M_j^s$. This is equivalent since $MC_j^s = MP_j^s \oplus M_j^s = PPP_i^s \oplus M_1^s \oplus M_j^s$.
- We replace the assignment $CCC_1^s \leftarrow MC_1^s \oplus CCC_2^s \oplus \cdots \oplus CCC_{\mathsf{m}^s}^s \oplus h(\mathsf{T}^s)$ in line 138 by the equivalent assignment $CCC_1^s \leftarrow PPP_1^s \oplus M_1^s \oplus (PPP_2^s \oplus CCC_2^s) \oplus \cdots \oplus (PPP_{\mathsf{m}^s}^s \oplus CCC_{\mathsf{m}^s}^s)$. This is indeed equivalent since $MC_1^s = MP_1^s \oplus M_1^s = PPP_1^s \oplus \cdots \oplus PPP_{\mathsf{m}^s}^s \oplus h(\mathsf{T}^s) \oplus M_1^s$.

Clearly, these changes preserve the distribution of all those variables, and we make the symmetric changes also for decryption queries.

In addition to these changes, we also slightly simplify the logic of the game by assigning value to MM^s and adding it to Domain and Range even in the case that $P_{m^s}^s$ is a full block ($|P_{m^s}^s| = n$). This has no effect on the answers that are returned to the adversary, but it may increase the probability of the flag *bad* being set (since we may introduce collisions that were not present before).

The resulting game R2 is described in Figure 6. 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

$$\Pr[\,A^{\mathrm{R1}} \Rightarrow 1\,] = \Pr[\,A^{\mathrm{R2}} \Rightarrow 1\,] \quad \text{and} \quad \Pr[\,A^{\mathrm{R1}} \text{ sets } bad\,] \le \Pr[\,A^{\mathrm{R2}} \text{ sets } bad\,] \tag{5}$$

We note that in game R2 we respond to any encipher query P^s by returning $|P^s|$ random bits, and similarly, we respond to any decipher query C^s by returning $|C^s|$ random bits. Thus R2 provides an adversary with an identical view to a pair of random-bit oracles,

$$\Pr[\,A^{\mathrm{R2}} \Rightarrow 1\,] \quad = \quad \Pr[\,A^{\widetilde{\pm\mathrm{rnd}}} \Rightarrow 1\,] \tag{6}$$

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

$$\begin{aligned}
\mathbf{Adv}^{\widetilde{\pm\mathrm{rnd}}}_{\mathrm{EME[Perm}(n)]}(A) &= \Pr[\,A^{\mathrm{E1}} \Rightarrow 1\,] + \frac{\sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})}{2^n} - \Pr[\,A^{\mathrm{R2}} \Rightarrow 1\,] \\
&= \Pr[\,A^{\mathrm{E1}} \Rightarrow 1\,] - \Pr[\,A^{\mathrm{R1}} \Rightarrow 1\,] + \frac{\sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})}{2^n} \\
&\le \Pr[\,A^{\mathrm{R1}} \text{ sets } bad\,] + \frac{\sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}}}{2^n} \\
&\le \Pr[\,A^{\mathrm{R2}} \text{ sets } bad\,] + \frac{\sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})}{2^n}
\end{aligned} \tag{7}$$

Our task is thus to bound $\Pr[\,A^{\mathrm{R2}} \text{ sets } bad\,]$.

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 $ty^s \leftarrow \mathsf{Enc}$
011 $(C_1^s \cdots C_{m^s-1}^s C_*^s) \xleftarrow{\$} \{0,1\}^{nm^s}$
012 $C_{m^s}^s \leftarrow$ 1st $|P_{m^s}^s|$ bits of $C_*^s$
013 return $C^s = C_1^s \cdots C_{m^s}^s$
```

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

```
020 $ty^s \leftarrow \mathsf{Dec}$
021 $(P_1^s \cdots P_{m^s-1}^s P_*^s) \xleftarrow{\$} \{0,1\}^{nm^s}$
022 $P_{m^s}^s \leftarrow$ 1st $|C_{m^s}^s|$ bits of $P_*^s$
023 return $P^s = P_1^s \cdots P_{m^s}^s$
```

Finalization:

First phase

```
050 $\mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \emptyset$;  $L \xleftarrow{\$} \{0,1\}^n$;  $h \xleftarrow{\$} \mathcal{H}$          // $\mathfrak{D}, \mathfrak{R}$ are multisets
051 repeat the following for all $s \in [1..q]$:

100 if $ty^s = \mathsf{Enc}$ then
101     if $|P_{m^s}^s| = n$ then $lastFull^s \leftarrow m^s$
102     else $lastFull^s \leftarrow m^s - 1$
103         $PPP_{m^s}^s \leftarrow P_{m^s}^s$ padded with 10..0;  $CCC_{m^s}^s \leftarrow C_{m^s}^s$ padded with 10..0

110     for $i \leftarrow 1$ to $lastFull^s$ do
111         $r = r[s,i]$ is the 1st index s.t. $P_i^s = P_i^r$
112         if $r < s$ then $PP_i^s \leftarrow PP_i^r$;  $PPP_i^s \leftarrow PPP_i^r$
113         else $PP_i^s \leftarrow P_i^s \oplus 2^{i-1}L$;  $PPP_i^s \xleftarrow{\$} \{0,1\}^n$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP_i^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP_i^s\}$

120     $M_1^s \xleftarrow{\$} \{0,1\}^n$
121     $MP_1^s \leftarrow PPP_1^s \oplus \cdots \oplus PPP_{m^s}^s \oplus h(T^s)$;  $MC_1^s \leftarrow MP_1^s \oplus M_1^s$;  $MM^s \leftarrow PPP_{m^s}^s \oplus C_*^s$
122     $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP_1^s, MM^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MM^s, MC_1^s\}$

130     for $i \leftarrow 2$ to $lastFull^s$ do
131         $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132         if $k = 0$ then
133             $MP_j^s \leftarrow PPP_i^s \oplus M_1^s$;  $M_j^s \xleftarrow{\$} \{0,1\}^n$;  $MC_j^s \leftarrow MP_j^s \oplus M_j^s$
134             $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP_j^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC_j^s\}$
135         $CCC_i^s \leftarrow PPP_i^s \oplus 2^k M_j^s$
136     $CCC_1^s \leftarrow PPP_1^s \oplus M_1^s \oplus (PPP_2^s \oplus CCC_2^s) \oplus \cdots \oplus (PPP_{\mathsf{m}^s}^s \oplus CCC_{\mathsf{m}^s}^s)$

140     for $i \leftarrow 1$ to $lastFull^s$ do
141         $CC_i^s \leftarrow C_i^s \oplus 2^{i-1}L$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC_i^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC_i^s\}$

200 The case $ty^s = \mathsf{Dec}$ is treated symmetrically
```

Second phase

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

Figure 7: Game R3 is adversarially indistinguishable from game RND2 but defers the setting of bad.

Game R3. Next we reorganize game R2 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 before that game R2 replies to any z-bit query with z random bits. Now, in game R3, shown in Figure 7, we make that even more clear by choosing the blocks $C_1^s \cdots C_{m^s-1}^s C_*^s$ or $P_1^s \cdots P_{m^s-1}^s P_*^s$ 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, and returning the answer to the adversary.

When the adversary finishes all of its oracle queries and halts, we execute the "finalization" step of game R3. First, we go over all the variables of the game and determine their values, just as we do in game R2. 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 R2. The following is thus clear:

$$\Pr[\, A^{\text{R2}} \text{ sets } bad\,] \;=\; \Pr[\, A^{\text{R3}} \text{ sets } bad\,] \tag{8}$$

Game N1. 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 R3. We may assume that the adversary is deterministic, and so the probability is over the random choices that are made while answering the queries (in lines 011 and 021), and the random choices in the finalization phase of the game (lines 050, 113, 120, 133, 213, 220, and 233). 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 chosen in lines 011 and 021, and for any set of queries (none pointless) associated to them, the finalization step of game R3 rarely sets *bad*. However, this statement isn't quite true. For example, assume that queries r and s ($r < s$) 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 141 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.

Another example is when encipher queries r, s have last blocks $P_{m^r}^r$, $P_{m^s}^s$, respectively, that are partial (namely $|P_{m^r}^r|, |P_{m^s}^s| < n$), and the blocks C_*^s, C_*^r that are chosen at random in line 11 satisfy $(P_{m^r}^r 10..0) \oplus C_*^r = (P_{m^s}^s 10..0) \oplus C_*^s$. In this case, we would have $MM^r = MM^s$ and since both are added to $\mathfrak{D}$ in line 122 we would set *bad* when we examine their values in line 300. Similar examples can be shown for decipher queries.

We call such collisions *immediate* collisions. Formally, an immediate collision on encipher happens whenever s is an encipher query and for some $r < s$ we have either $C_i^s = C_i^r$ for some $i \le \mathit{lastFull}^s$, or $C_*^s = (P_{m^s}^s 10..0) \oplus (P_{m^r}^r 10..0) \oplus C_*^r$ when $|P_{m^r}^r|, |P_{m^s}^s| < n$. An immediate collision on decipher happens whenever s is an decipher query and for some $r < s$ we have either $P_i^s = P_i^r$ for some $i \le \mathit{lastFull}^s$, or $P_*^s = (C_{m^s}^s 10..0) \oplus (C_{m^r}^r 10..0) \oplus P_*^r$ when $|C_{m^r}^r|, |C_{m^s}^s| < n$. The probability of an immediate collision (on either encipher or decipher) in game R3 is at most

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

```
050 $\mathfrak{D} \leftarrow \mathfrak{R} \leftarrow \emptyset$;  $L \xleftarrow{\$} \{0,1\}^n$;  $h \xleftarrow{\$} \mathcal{H}$          // $\mathfrak{D}, \mathfrak{R}$ are *multisets*
051 **for** $s \leftarrow 1$ **to** $q$ **do**
100     **if** $\mathsf{ty}^s = \mathsf{Enc}$ **then**
101         $\mathsf{C}^s_{\mathsf{m}^s} \leftarrow$ 1st $|\mathsf{P}^s_{m^s}|$ bits of $\mathsf{C}^s_*$
102         **if** $|\mathsf{P}^s_{\mathsf{m}^s}| = n$ **then** $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s$
103         **else** $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s - 1$;  $PPP^s_{\mathsf{m}^s} \leftarrow \mathsf{P}^s_{\mathsf{m}^s}$ padded with 10..0;  $CCC^s_{\mathsf{m}^s} \leftarrow \mathsf{C}^s_{\mathsf{m}^s}$ padded with 10..0
110         **for** $i \leftarrow 1$ **to** $\mathsf{lastFull}^s$ **do**
111             $r = r[s,i]$ is the 1st index s.t. $\mathsf{P}^s_i = \mathsf{P}^r_i$
112             **if** $r < s$ **then** $PP^s_i \leftarrow PP^r_i$;  $PPP^s_i \leftarrow PPP^r_i$
113             **else** $PP^s_i \leftarrow \mathsf{P}^s_i \oplus 2^{i-1}L$;  $PPP^s_i \xleftarrow{\$} \{0,1\}^n$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP^s_i\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP^s_i\}$
120         $M^s_1 \xleftarrow{\$} \{0,1\}^n$
121         $MP^s_1 \leftarrow PPP^s_1 \oplus \cdots \oplus PPP^s_{\mathsf{m}^s} \oplus h(\mathsf{T}^s)$;  $MC^s_1 \leftarrow MP^s_1 \oplus M^s_1$;  $MM^s \leftarrow PPP^s_{\mathsf{m}^s} \oplus \mathsf{C}^s_*$
122         $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_1, MM^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MM^s, MC^s_1\}$
130         **for** $i \leftarrow 2$ **to** $\mathsf{lastFull}^s$ **do**
131             $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132             **if** $k = 0$ **then**
133                 $MP^s_j \leftarrow PPP^s_i \oplus M^s_1$;  $M^s_j \xleftarrow{\$} \{0,1\}^n$;  $MC^s_j \leftarrow MP^s_j \oplus M^s_j$
134                 $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_j\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC^s_j\}$
135             $CCC^s_i \leftarrow PPP^s_i \oplus 2^k M^s_j$
136         $CCC^s_1 \leftarrow PPP^s_1 \oplus M^s_1 \oplus (PPP^s_2 \oplus CCC^s_2) \oplus \cdots \oplus (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$
140         **for** $i \leftarrow 1$ **to** $\mathsf{lastFull}^s$ **do**
141             $CC^s_i \leftarrow \mathsf{C}^s_i \oplus 2^{i-1}L$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC^s_i\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC^s_i\}$
200     **else**          // $\mathsf{ty}^s = \mathsf{Dec}$
201         $\mathsf{P}^s_{\mathsf{m}^s} \leftarrow$ 1st $|\mathsf{C}^s_{m^s}|$ bits of $\mathsf{P}^s_*$
202         **if** $|\mathsf{C}^s_{\mathsf{m}^s}| = n$ **then** $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s$
203         **else** $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s - 1$;  $PPP^s_{\mathsf{m}^s} \leftarrow \mathsf{P}^s_{\mathsf{m}^s}$ padded with 10..0;  $CCC^s_{\mathsf{m}^s} \leftarrow \mathsf{C}^s_{\mathsf{m}^s}$ padded with 10..0
210         **for** $i \leftarrow 1$ **to** $\mathsf{lastFull}^s$ **do**
211             $r = r[s,i]$ is the 1st index s.t. $\mathsf{C}^s_i = \mathsf{C}^r_i$
212             **if** $r < s$ **then** $CC^s_i \leftarrow CC^r_i$;  $CCC^s_i \leftarrow CCC^r_i$
213             **else** $CC^s_i \leftarrow \mathsf{C}^s_i \oplus 2^{i-1}L$;  $CCC^s_i \xleftarrow{\$} \{0,1\}^n$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC^s_i\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{CC^s_i\}$
220         $M^s_1 \xleftarrow{\$} \{0,1\}^n$
221         $MC^s_1 \leftarrow CCC^s_1 \oplus \cdots \oplus CCC^s_{\mathsf{m}^s} \oplus h(\mathsf{T}^s)$;  $MP^s_1 \leftarrow MC^s_1 \oplus M^s_1$;  $MM^s \leftarrow CCC^s_{\mathsf{m}^s} \oplus \mathsf{P}^s_*$
222         $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_1, MM^s\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MM^s, MC^s_1\}$
230         **for** $i \leftarrow 2$ **to** $\mathsf{lastFull}^s$ **do**
231             $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
232             **if** $k = 0$ **then**
233                 $MC^s_j \leftarrow CCC^s_i \oplus M^s_1$;  $M^s_j \xleftarrow{\$} \{0,1\}^n$;  $MP^s_j \leftarrow MC^s_j \oplus M^s_j$
234                 $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_j\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{MC^s_j\}$
235             $PPP^s_i \leftarrow CCC^s_i \oplus 2^k M^s_j$
236         $PPP^s_1 \leftarrow CCC^s_1 \oplus M^s_1 \oplus (PPP^s_2 \oplus CCC^s_2) \oplus \cdots \oplus (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$
240         **for** $i \leftarrow 1$ **to** $\mathsf{lastFull}^s$ **do**
241             $PP^s_i \leftarrow \mathsf{C}^s_i \oplus 2^{i-1}L$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP^s_i\}$;  $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{PPP^s_i\}$
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 N1 is based on game R3 but now $\boldsymbol{\tau} = (\mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C})$ is a fixed, allowed transcript.

We make from the Finalization part of game R3 a new game, game N1 (for "noninteractive"). This game silently depends on a fixed transcript $\boldsymbol{\tau} = \langle \mathbf{ty}, \mathbf{T}, \mathbf{P}, \mathbf{C} \rangle$ with ty^s the "type" of query s ($\mathsf{ty}^s \in \{\mathsf{Enc}, \mathsf{Dec}\}$) and $\mathsf{T}^s \in \{0,1\}^*$ the associated data to query s. Also for an encipher query s we have $\mathsf{P}^s = \mathsf{P}^s_1 \cdots \mathsf{P}^s_{\mathsf{m}^s}$ and $\mathsf{C}^s = \mathsf{C}^s_1 \cdots \mathsf{C}^s_{\mathsf{m}^s-1}, \mathsf{C}^s_*$, and for a decipher query we have $\mathsf{P}^s = \mathsf{P}^s_1 \cdots \mathsf{P}^s_{\mathsf{m}^s-1} \mathsf{P}^s_*$ and $\mathsf{C}^s = \mathsf{C}^s_1 \cdots \mathsf{C}^s_{\mathsf{m}^s}$.

Below we let $\mathsf{lastFull}^s$ denote either m^s if the last block in query s is full or $\mathsf{m}^s - 1$ if it is partial. Also, for an encipher query we denote by P^s_* the padding of $\mathsf{P}^s_{\mathsf{m}^s}$, $\mathsf{P}^s_* = \mathsf{P}^s_{\mathsf{m}^s} 10..0$, and by $\mathsf{C}^s_{\mathsf{m}^s}$ we denote the first $|\mathsf{P}^s_{\mathsf{m}^s}|$ bits of C^s_*. Similarly, for a decipher query we denote $\mathsf{C}^s_* = \mathsf{C}^s_{\mathsf{m}^s} 10..0$, and denote by $\mathsf{P}^s_{\mathsf{m}^s}$ the first $|\mathsf{C}^s_{\mathsf{m}^s}|$ bits of P^s_*. Since the transcript $\boldsymbol{\tau}$ is fixed, then also all these quantities are fixed.

This fixed transcript $\boldsymbol{\tau}$ 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) $(\mathsf{T}^s, \mathsf{P}^s) \neq (\mathsf{T}^r, \mathsf{P}^r)$,
(ii) $\mathsf{C}^s_i \neq \mathsf{C}^r_i$ for any $i \in [1 .. \mathsf{lastFull}^s]$,
(iii) If $|\mathsf{P}^s_{\mathsf{m}^s}|, |\mathsf{P}^r_{\mathsf{m}^r}| < n$ then $\mathsf{C}^s_* \neq (\mathsf{P}^s_{\mathsf{m}^s} 10..0) \oplus (\mathsf{P}^r_{\mathsf{m}^r} 10..0) \oplus \mathsf{C}^r_*$;
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 $i \in [1 .. \mathsf{lastFull}^s]$,
(iii) If $|\mathsf{C}^s_{\mathsf{m}^s}|, |\mathsf{C}^r_{\mathsf{m}^r}| < n$ then $\mathsf{P}^s_* \neq (\mathsf{C}^s_{\mathsf{m}^s} 10..0) \oplus (\mathsf{C}^r_{\mathsf{m}^r} 10..0) \oplus \mathsf{P}^r_*$.
Now fix an allowed transcript $\boldsymbol{\tau}$ that *maximizes the probability of the flag bad being set.* This one transcript $\boldsymbol{\tau}$ is hardwired into game N1. We have that

$$\Pr[\, A^{\mathrm{R3}} \text{ sets } \mathit{bad}\,] \;\leq\; \Pr[\,\mathrm{N1} \text{ sets } \mathit{bad}\,] + \frac{q\sigma_n^d}{2^n} \tag{9}$$

This step can be viewed as conditioning on the 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 R3 to game N1 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 R3, 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 N2. 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 N1

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

and therefore $\Pr[\,\mathrm{N1} \text{ sets } \mathit{bad}\,] \leq 2 \cdot \Pr[\text{some value appears more than once in } \mathfrak{D}]$. We therefore replace the game N1 by game N2, in which we only set the flag *bad* if there is a collision in $\mathfrak{D}$. We

```
050 $\mathfrak{D} \leftarrow \emptyset$; $\boxed{L \xleftarrow{\$} \{0,1\}^n}$; $\boxed{h \xleftarrow{\$} \mathcal{H}}$          // $\mathfrak{D}$ is a *multiset*
051 for $s \leftarrow 1$ to $q$ do
100     if $\mathsf{ty}^s = \mathsf{Enc}$ then
101         $\mathsf{C}^s_{\mathsf{m}^s} \leftarrow$ 1st $|\mathsf{P}^s_{m^s}|$ bits of $\mathsf{C}^s_*$
102         if $|\mathsf{P}^s_{\mathsf{m}^s}| = n$ then $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s$
103         else $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s - 1$;  $PPP^s_{\mathsf{m}^s} \leftarrow \mathsf{P}^s_{\mathsf{m}^s}$ padded with 10..0;  $CCC^s_{\mathsf{m}^s} \leftarrow \mathsf{C}^s_{\mathsf{m}^s}$ padded with 10..0

110         for $i \leftarrow 1$ to $\mathsf{lastFull}^s$ do
111             $r = r[s,i]$ is the 1st index s.t. $\mathsf{P}^s_i = \mathsf{P}^r_i$
112             if $r < s$ then $PP^s_i \leftarrow PP^r_i$;  $PPP^s_i \leftarrow PPP^r_i$
113             else $PP^s_i \leftarrow \mathsf{P}^s_i \oplus 2^{i-1}L$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP^s_i\}$ ;  $\boxed{PPP^s_i \xleftarrow{\$} \{0,1\}^n}$

120         $\boxed{M^s_1 \xleftarrow{\$} \{0,1\}^n}$ ;  $MP^s_1 \leftarrow PPP^s_1 \oplus \cdots \oplus PPP^s_{\mathsf{m}^s} \oplus h(\mathsf{T}^s)$;  $MM^s \leftarrow PPP^s_{\mathsf{m}^s} \oplus \mathsf{C}^s_*$
121         $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_1, MM^s\}$

130         for $i \leftarrow 2$ to $\mathsf{lastFull}^s$ do
131             $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
132             if $k = 0$ then $MP^s_j \leftarrow PPP^s_i \oplus M^s_1$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_j\}$ ;  $\boxed{M^s_j \xleftarrow{\$} \{0,1\}^n}$
134             $CCC^s_i \leftarrow PPP^s_i \oplus 2^k M^s_j$
135         $CCC^s_1 \leftarrow PPP^s_1 \oplus M^s_1 \oplus (PPP^s_2 \oplus CCC^s_2) \oplus \cdots \oplus (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$

140         for $i \leftarrow 1$ to $\mathsf{lastFull}^s$ do  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC^s_i\}$

200     else                // $\mathsf{ty}^s = \mathsf{Dec}$
201         $\mathsf{P}^s_{\mathsf{m}^s} \leftarrow$ 1st $|\mathsf{C}^s_{m^s}|$ bits of $\mathsf{P}^s_*$
202         if $|\mathsf{C}^s_{\mathsf{m}^s}| = n$ then $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s$
203         else $\mathsf{lastFull}^s \leftarrow \mathsf{m}^s - 1$;  $CCC^s_{\mathsf{m}^s} \leftarrow \mathsf{C}^s_{\mathsf{m}^s}$ padded with 10..0

210         for $i \leftarrow 1$ to $\mathsf{lastFull}^s$ do
211             $r = r[s,i]$ is the 1st index s.t. $\mathsf{C}^s_i = \mathsf{C}^r_i$
212             if $r < s$ then $CCC^s_i \leftarrow CCC^r_i$
213             else $\boxed{CCC^s_i \xleftarrow{\$} \{0,1\}^n}$ ;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{CCC^s_i\}$

220         $\boxed{M^s_1 \xleftarrow{\$} \{0,1\}^n}$ ;  $MP^s_1 \leftarrow CCC^s_1 \oplus \cdots \oplus CCC^s_{\mathsf{m}^s} \oplus h(\mathsf{T}^s) \oplus M^s_1$;  $MM^s \leftarrow CCC^s_{\mathsf{m}^s} \oplus \mathsf{P}^s_*$
221         $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MM^s, MP^s_1\}$

230         for $i \leftarrow 2$ to $\mathsf{lastFull}^s$ do
231             $j = \lceil i/n \rceil$, $k = (i-1) \bmod n$
232             if $k = 0$ then $\boxed{M^s_j \xleftarrow{\$} \{0,1\}^n}$ ;  $MP^s_j \leftarrow CCC^s_i \oplus M^s_1 \oplus M^s_j$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{MP^s_j\}$
234             $PPP^s_i \leftarrow CCC^s_i \oplus 2^k M^s_j$
235         $PPP^s_1 \leftarrow CCC^s_1 \oplus M^s_1 \oplus (PPP^s_2 \oplus CCC^s_2) \oplus \cdots \oplus (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$

240         for $i \leftarrow 1$ to $\mathsf{lastFull}^s$ do $PP^s_i \leftarrow \mathsf{P}^s_i \oplus 2^{i-1}L$;  $\mathfrak{D} \leftarrow \mathfrak{D} \cup \{PP^s_i\}$

300 $bad \leftarrow$ Some value appears more than once in $\mathfrak{D}$
```

Figure 9: Game N2. Twice the probability that bad gets set in this game bounds the probability that bad gets set in game N1. We highlight random selection by boxing, statements that grow $\mathfrak{D}$ by shading.

now can drop the code that handles $\mathfrak{R}$, as well as anything else that doesn't affect the multiset $\mathfrak{D}$. Specifically, we make the following changes in the code of the game N1:

- We drop the multiset $\mathfrak{R}$ from the code.
- We replace the assignment $MP_1^s \leftarrow MC_1^s \oplus M_1^s$ from line 221 in game N1 by the equivalent assignment $MP_1^s \leftarrow CCC_1^s \oplus \cdots \oplus CCC_{\mathsf{m}^s}^s \oplus h(\mathsf{T}^s) \oplus M_1^s$. Similarly, we replace the assignment $MP_j^s \leftarrow MC_j^s \oplus M_j^s$ from line 233 by the equivalent assignment $MP_j^s \leftarrow CCC_i^s \oplus M_1^s \oplus M_j^s$.
- Now the variable CC_i^s and MC_j^s are never used in the code, so we drop them altogether.

The resulting game is described in Figure 9, and we have

$$\Pr[\,\text{N1 sets } \mathit{bad}\,] \quad \leq \quad 2 \cdot \Pr[\,\text{N2 sets } \mathit{bad}\,] \tag{10}$$

A.2 Analysis of the non-interactive game

We are now ready to analyze the resulting game N2, showing that the event "N2 sets *bad*" only happens with small probability. 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 N2. 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 N2; specifically,

$$\begin{aligned}\mathfrak{D} \;=\; & \{MM^s \mid \; s \leq q\} \;\cup\; \{MP_j^s \mid \; s \leq q,\; j \leq \lceil \mathsf{lastFull}^s/n \rceil\} \\ & \cup\; \{PP_i^s \mid \; \mathsf{ty}^s = \mathsf{Dec},\; i \leq \mathsf{lastFull}^s\} \;\cup\; \{PP_i^s \mid \; \mathsf{ty}^s = \mathsf{Enc},\; i \leq \mathsf{lastFull}^s,\; s = \mathsf{r}[s,i]\} \\ & \cup\; \{CCC_i^s \mid \; \mathsf{ty}^s = \mathsf{Enc},\; i \leq \mathsf{lastFull}^s\} \;\cup\; \{CCC_i^s \mid \; \mathsf{ty}^s = \mathsf{Dec},\; i \leq \mathsf{lastFull}^s,\; s = \mathsf{r}[s,i]\}\end{aligned}$$

We view the formal variables in $\mathfrak{D}$ as *ordered* according to when they are assigned a value in the execution of game N2. This ordering too is fixed, depending only on the fixed transcript $\boldsymbol{\tau}$.

Throughout the remainder of this section, in all probability claims, the implicit experiment is that of game N2. 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 its value.) The rest of this section is devoted to case analysis, proving the following claim:

Claim 3 For any two distinct variable $X, X' \in \mathfrak{D}$ we have that $\Pr[X = X'] \leq 2^{-n}$.

Before proving Claim 3, we show how to use it to complete the proof of Theorem 1. Recall that we denote the total number of block encryptions or decryptions by N_{be}, so there are no more than N_{be} variables in $\mathfrak{D}$ and the union bound gives us

$$\Pr[\,\text{N2 sets } \mathit{bad}\,] \leq \binom{N_{\text{be}}}{2} / 2^n \tag{11}$$

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

$$\begin{aligned}
\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) &\le \mathbf{Adv}^{\pm\widetilde{\mathrm{rnd}}}_{\mathrm{EME}[\mathrm{Perm}(n)]}(A) + q(q-1)/2^{n+1} \\
&\le 2\cdot \Pr[\text{N2 sets } \mathit{bad}] + q\sigma_n^d/2^n + \sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})/2^n + q(q-1)/2^{n+1} \\
&\le 2\cdot \binom{N_{\mathrm{be}}}{2}/2^n + q\sigma_n^d/2^n + \sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})/2^n + q(q-1)/2^{n+1}
\end{aligned}$$

Using the bound $N_{\mathrm{be}} < (2+\frac{1}{n})\sigma_n^d + 2q$ from Eq. (2) and substituting $\sigma_n = \sigma_n^d + \sigma_n^a$ (and assuming that $n > 32$), it can be shown that

$$2\frac{\binom{N_{\mathrm{be}}}{2}}{2^n} + \frac{q\sigma_n^d}{2^n} + \frac{\sigma_n^a(\sigma_n^a + 2N_{\mathrm{be}})}{2^n} + \frac{q(q-1)}{2^{n+1}} < \frac{(2.5\sigma_n + 3q)^2}{2^{n+1}}$$

Since A was an arbitrary adversary with query complexity of q and σ_n, we are done. ▌

A.2.1 The case analysis

We now need to prove Claim 3. We first prove a few claims, each covering some special cases of collisions (Claim 7 through Corollary 15 below), and then show that all possible cases are indeed covered by these claims.

Inspecting the code of game N2 we see that some of the variables in this game are directly chosen at random from $\{0,1\}^n$, while others are assigned values deterministically. Specifically, the variables that are directly chosen at random (other than the function h) are L, all the variables M_j^s, the variables PPP_i^s such that $\mathsf{ty}^s = \mathsf{Enc}$, $i \le \mathsf{lastFull}^s$ and $s = \mathsf{r}[s,i]$, and the variables CCC_i^s such that $\mathsf{ty}^s = \mathsf{Dec}$, $i \le \mathsf{lastFull}^s$ and $s = \mathsf{r}[s,i]$. Hereafter we refer to these variables as the *free variables* of the game, and we let $\mathfrak{F}$ denote the set of them:

$$\begin{aligned}
\mathfrak{F} \;=\; &\{L\} \;\cup\; \{M_j^s \mid \; j \le \lceil \mathsf{lastFull}^s/n \rceil\} \\
&\cup\; \{PPP_i^s \mid \; \mathsf{ty}^s = \mathsf{Enc},\; i \le \mathsf{lastFull}^s,\; s = \mathsf{r}[s,i]\} \\
&\cup\; \{CCC_i^s \mid \; \mathsf{ty}^s = \mathsf{Dec},\; i \le \mathsf{lastFull}^s,\; s = \mathsf{r}[s,i]\}
\end{aligned}$$

The value of any other variable in the game can be expressed as a deterministic function in these free variables (and in the function h). The bulk of the argument below is roughly to show that for any pair of variables in $\mathfrak{D}$, their sum is either some non-zero constant, or it depends linearly on some free variable.[3]

We start with some helpful observations regarding the sum of CCC's (or PPP's) from the same query. Fix some $s \le q$ and a non-empty set of indices $I \subseteq [1..\mathsf{lastFull}^s]$, and denote its complement by $\overline{I} \overset{\mathrm{def}}{=} [1..\mathsf{lastFull}^s] \setminus I$. Also let

$$j(I) \overset{\mathrm{def}}{=} \begin{cases} \lceil \max(I)/n \rceil & \text{if } 1 \notin I \\ \lceil \max(\overline{I})/n \rceil & \text{if } 1 \in I, \overline{I} \ne \emptyset \\ 1 & \text{if } \overline{I} = \emptyset \end{cases}$$

(Roughly, $j(I)$ is either the index of the "last chunk of n blocks that intersects with I" or the index of the "last chunk that intersects with $\overline{I}$", depending on whether or not $1 \in I$.)

[3]In some cases we show that this sum depends on the choice of h in a way that ensures that it is almost always non-zero.

Claim 4 For an encipher query s and a non-empty set $I \subseteq [1 \,..\, \mathsf{lastFull}^s]$, we have $\sum_{i\in I} CCC_i^s = aM^s_{j(I)} \oplus \beta$, where $a \neq 0$ is a constant (that depends on the set I), and β is an expression that depends only on constants and variables that were determined before $M^s_{j(I)}$ in the game N2.

Likewise, if r is a decipher query ($ty^s = \mathsf{Dec}$), then $\sum_{i\in I} PPP_i^s = aM^s_{j(I)} \oplus \beta$, where $a \neq 0$ is a constant and β is an expression that depends only on constants and variables that were determined before $M^s_{j(I)}$ in the game N2.

Proof: We prove here only the first assertion. The proof of the other assertion is symmetric. Consider first the case where $1 \notin I$, and denote $I_{\text{last}} \stackrel{\text{def}}{=} \{i \in I \mid \lceil i/n \rceil = j(I)\}$. Notice that $I_{\text{last}} \neq \emptyset$. Since s is an encipher query and $1 \notin I$, then for all $i \in I$, the value of CCC_i^s is set in line 134 to $CCC_i^s \leftarrow PPP_i^s \oplus 2^{(i-1) \bmod n} \cdot M^s_{\lceil i/n \rceil}$. Thus we have

$$\begin{aligned}\sum_{i\in I} CCC_i^s &= \sum_{i\in I} PPP_i^s \oplus 2^{(i-1) \bmod n} \cdot M^s_{\lceil i/n \rceil} \\ &= \text{things-that-were-determined-before-}M^s_{j(I)} \oplus \left(\sum_{i\in I_{\text{last}}} 2^{(i-1) \bmod n}\right) \cdot M^s_{j(I)}\end{aligned}$$

It is left to show that the coefficient of $M^s_{j(I)}$ is non-zero. Let $j \stackrel{\text{def}}{=} j(I)$ and recall that $I_{\text{last}} \subseteq \{(j-1)n+1, \ldots, jn\}$. Hence, if we denote $I'_{\text{last}} = \{i-(j-1)n \mid i \in I_{\text{last}}\}$ then $I'_{\text{last}} \subseteq \{1, \ldots n\}$ and $I'_{\text{last}} \neq \emptyset$, and therefore $\sum_{i\in I_{\text{last}}} 2^{(i-1) \bmod n} = \sum_{i' \in I'_{\text{last}}} 2^{i'-1} \neq 0$.

For the case where $1 \in I$, let X^s be the constant which is either 0 if the last block in query s is full ($|\mathsf{P}^s_{\mathsf{m}^s}| = n$, $\mathsf{lastFull}^s = \mathsf{m}^s$) or $X^s = (PPP^s_{\mathsf{m}^s} \oplus CCC^s_{\mathsf{m}^s})$ if it is a partial block ($|\mathsf{P}^s_{\mathsf{m}^s}| < n$, $\mathsf{lastFull}^s = \mathsf{m}^s - 1$). (Note that X^s is indeed a constant: if $\mathsf{P}^r_{\mathsf{m}^s}$ is a partial block then X^s is equal to $\mathsf{P}^s_{\mathsf{m}^s} \oplus \mathsf{C}^s_{\mathsf{m}^s}$, padded with zeros to n bits.) Using this notation we can express the value of CCC_1^s as

$$CCC_1^s = PPP_1^s \oplus M_1^s \oplus \sum_{i=2}^{\mathsf{m}^s} (PPP_i^s \oplus CCC_i^s) \;=\; PPP_1^s \oplus M_1^s \oplus \sum_{i=2}^{\mathsf{lastFull}^s} (PPP_i^s \oplus CCC_i^s) \oplus X^s$$

Recall that in this case $1 \in I$ so $\overline{I} = [2..\mathsf{lastFull}^s] \setminus I$. Thus we can write

$$\begin{aligned}\sum_{i\in I} CCC_i^s &= CCC_1^s \oplus \sum_{i\in I,\ i>1} CCC_i^s \\ &= \left(PPP_1^s \oplus M_1^s \oplus \sum_{i=2}^{\mathsf{lastFull}^s} (PPP_i^s \oplus CCC_i^s) \oplus X^s\right) \oplus \sum_{i\in I,\ i>1} CCC_i^s \\ &= X^s \oplus \sum_{i=1}^{\mathsf{lastFull}^s} PPP_i^s \oplus M_1^s \oplus \sum_{i\in \overline{I}} CCC_i^s \\ &= X^s \oplus \sum_{i=1}^{\mathsf{lastFull}^s} PPP_i^s \oplus M_1^s \oplus \sum_{i\in \overline{I}} (PPP_i^s \oplus 2^{(i-1) \bmod n} \cdot M^s_{\lceil i/n \rceil}) \\ &= \text{things-that-were-determined-before-}M_1^s \oplus M_1^s \oplus \sum_{i\in \overline{I}} 2^{(i-1) \bmod n} \cdot M^s_{\lceil i/n \rceil}\end{aligned}$$

Denote $\overline{I}_{\text{last}} \stackrel{\text{def}}{=} \{i \in \overline{I} \mid \lceil i/n \rceil = j(I)\}$, and note that $\overline{I}_{\text{last}} = \emptyset$ if and only if $\overline{I} = \emptyset$.

Now, if $j(I) > 1$ (which means that $\overline{I} \neq \emptyset$ and in particular $\overline{I}_{\text{last}} \neq \emptyset$), then the coefficient of $M^s_{j(I)}$ in the expression above is $\sum_{i \in \overline{I}_{\text{last}}} 2^{(i-1) \bmod n}$, which is non-zero since $\overline{I}_{\text{last}}$ is non-empty. If $j(I) = 1$ then the coefficient of $M^s_{j(I)} = M^s_1$ is $(1 \oplus \sum_{i \in \overline{I}_{\text{last}}} 2^{i-1})$, which is non-zero since $1 \notin \overline{I}_{\text{last}}$. ▌

Corollary 5 For any query s and any fixed non-empty set $I \subseteq [1..\mathsf{lastFull}^s]$, we have $\Pr[\sum_{i \in I} CCC^s_i = 0] = 2^{-n}$ and similarly $\Pr[\sum_{i \in I} PPP^s_i = 0] = 2^{-n}$.

Proof: Again, due to symmetry it is sufficient to prove only the case of $\sum_i CCC^s_i$. If s is an encipher query then this follows directly from Claim 4. If s is a decipher query, then each CCC^s_i is either itself a free variable (if it is a "new block", $\mathsf{r}[s,i] = s$) or it is set equal to $CCC^{\mathsf{r}[s,i]}_i$ (where $\mathsf{r}[s,i]$ is the last query such that $\mathsf{C}^r_i = \mathsf{C}^s_i$). Hence we can write $\sum_{i \in I} CCC^s_i = \sum_{i \in I} CCC^{\mathsf{r}[s,i]}_i$.

Let r^* be the largest value $\mathsf{r}[s,i]$ for any $i \in I$ (for example, $r^* = s$ if any of the CCC^s_i's is a "new block"). Also, let I^* be all the indices $i \in I$ such that $\mathsf{r}[s,i] = r^*$, and let i^* be the largest index in I^*. That is, we define

$$r^* \stackrel{\text{def}}{=} \max\{\mathsf{r}[s,i] \mid i \in I\}, \quad I^* \stackrel{\text{def}}{=} \{i \in I \mid \mathsf{r}[s,i] = r^*\}, \quad i^* \stackrel{\text{def}}{=} \max(I^*)$$

By definition, since I is non-empty then I^* must also be non-empty. Also, for any $i \in I \setminus I^*$ we have $\mathsf{r}[s,i] < r^*$. If query r^* is an encipher query, then $CCC^{r^*}_{i^*}$ is a free variable and we can write

$$\begin{aligned}\sum_{i \in I} CCC^s_i &= \sum_{i \in I \setminus \{i^*\}} CCC^{\mathsf{r}[s,i]}_i \oplus CCC^{r^*}_{i^*} \\ &= \text{things-that-were-determined-before-}CCC^{r^*}_{i^*} \oplus CCC^{r^*}_{i^*}\end{aligned}$$

Hence the probability that $\sum_{i \in I} CCC^s_i = 0$, over the random choice of $CCC^{r^*}_{i^*}$, is exactly 2^{-n}. On the other hand, if query r^* is a decipher query, then we can apply Claim 4 to query r^* and get

$$\begin{aligned}\sum_{i \in I} CCC^s_i &= \sum_{i \in I^*} CCC^{r^*}_i \oplus \sum_{i \in I \setminus I^*} CCC^{\mathsf{r}[s,i]}_i \\ &= \left(\alpha M^{r^*}_{j(I^*)} \oplus \beta\right) \oplus \sum_{i \in I \setminus I^*} CCC^{\mathsf{r}[s,i]}_i \quad = \quad \alpha M^{r^*}_{j(I^*)} \oplus \beta'\end{aligned}$$

where $\alpha \neq 0$ and β' is an expression that depends only on constants and variables that were determined before $M^{r^*}_{j(I^*)}$ in the game N2. Again, the probability of $\sum_{i \in I} CCC^s_i = 0$, over the random choice of $M^{r^*}_{j(I^*)}$, is exactly 2^{-n}. ▌

The "last free variable". In the case analysis to come, we consider for each variable $X \in \mathfrak{D}$, the *last free variable* (in the ordering of the game N2) that X depends on, denoted $\phi(X)$. Formally, we have a function ϕ: $\mathfrak{D} \to \mathfrak{F} \cup \{\mathsf{none}\}$ that is defined as follows:

- As the variables MM^s are all constants, depending only on $\mathsf{P}^s_{\mathsf{m}^s}$ and C^s_* (or $\mathsf{C}^s_{\mathsf{m}^s}$ and P^s_*), we denote $\phi(MM^s) = \mathsf{none}$ for all s.
- For the formal variables $PP^s_i \in \mathfrak{D}$, this last free variable is L, $\phi(PP^s_i) = L$.

$$\phi(MM^s) \;=\; \text{none} \qquad\qquad \text{MM}$$

$$\phi(PP_i^s) \;=\; L \qquad \text{if } \mathsf{ty}^s = \mathsf{Dec} \text{ or } s = \mathsf{r}[s,i] \qquad \text{PP}$$

$$\phi(CCC_i^s) \;=\; \begin{cases} CCC_i^s & \text{if } \mathsf{ty}^s = \mathsf{Dec} \text{ and } s = \mathsf{r}[s,i] \quad \text{CCC1} \\ M^s_{\lceil i/n \rceil} & \text{if } \mathsf{ty}^s = \mathsf{Enc} \text{ and } i > 1 \quad \text{CCC2} \\ M^s_{\lceil \mathsf{lastFull}^s/n \rceil} & \text{if } \mathsf{ty}^s = \mathsf{Enc} \text{ and } i = 1 \quad \text{CCC3} \end{cases}$$

$$\phi(MP_j^s) \;=\; \begin{cases} M_j^s & \text{if } \mathsf{ty}^s = \mathsf{Dec} \quad \text{MM1} \\ M_1^s & \text{if } \mathsf{ty}^s = \mathsf{Enc} \text{ and } j > 1 \quad \text{MM2} \\ PPP^{\mathsf{rmax}[s]}_{\mathsf{imax}[s]} & \text{if } \mathsf{ty}^s = \mathsf{Enc},\ j = 1, \text{ and } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Enc} \quad \text{MM3} \\ M^{\mathsf{rmax}[s]}_{\mathsf{jmax}[s]} & \text{if } \mathsf{ty}^s = \mathsf{Enc},\ j = 1, \text{ and } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Dec} \quad \text{MM4} \end{cases}$$

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

- 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)[4] or $M^s_{\lceil i/n \rceil}$ (on encipher, if $i > 1$), or $M^s_{\lceil \mathsf{lastFull}^s/n \rceil}$ (on encipher, if $i = 1$). The last two assertions are corollaries of Claim 4 for $I = \{i\}$.

- The rules for the MP_j^s's are a bit more involved. Clearly, on decipher we have $\phi(MP_j^s) = M_j^s$ for all j, and on encipher we have $\phi(MP_j^s) = M_1^s$ for all $j > 1$. To define $\phi(MP_1^s)$ on encipher, recall that we set (in line 120) $MP_1^s \leftarrow h(\mathsf{T}^s) \oplus \sum_{i=1}^m PPP_i^s$, 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 $M^r_{\lceil i/n \rceil}$ (if query r is decipher and $i > 1$) or on $M^r_{\lceil \mathsf{lastFull}^r/n \rceil}$ (if query r is decipher and $i = 1$). To define $\phi(MP_1^s)$, we therefore denote

$$\mathsf{rmax}[s] \overset{\text{def}}{=} \max\{\mathsf{r}[s,i] \;\mid\; 1 \le i \le \mathsf{lastFull}^s\}$$
$$\mathsf{imax}[s] \overset{\text{def}}{=} \max\{i \le \mathsf{lastFull}^s \;\mid\; \mathsf{r}[s,i] = \mathsf{rmax}[s]\}$$
$$\text{and} \quad \mathsf{jmax}[s] \overset{\text{def}}{=} \begin{cases} \left\lceil \mathsf{lastFull}^{\mathsf{rmax}[s]}/n \right\rceil & \text{if } \mathsf{imax}[s] = 1 \\ \lceil\ \mathsf{imax}[s]/n\ \rceil & \text{if } \mathsf{imax}[s] > 1 \end{cases}$$

  Thus, when $\mathsf{ty}^s = \mathsf{Enc}$ we have

$$\phi(MP_1^s) = \begin{cases} PPP^{\mathsf{rmax}[s]}_{\mathsf{imax}[s]} & \text{if } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Enc} \\ M^{\mathsf{rmax}[s]}_{\mathsf{jmax}[s]} & \text{if } \mathsf{ty}^{\mathsf{rmax}[s]} = \mathsf{Dec} \end{cases}$$

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

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 N2. Justifying the name "last free variable" we observe the following, which follows from the preceding discussion:

Claim 6 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 N2 can be expressed as value$(X) = a \cdot$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 N2. $\square$

As an immediate corollary of Claim 6, we get the following.

Claim 7 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 N2. By Claim 6 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 GF(2^n), independently of the other free variables, it follows that $\Pr[X = X'] = 2^{-n}$. ▌

Claim 7 leaves us with the task of analyzing collisions between variables that depend on the same last free variable. These are handled in the next few claims.

Claim 8 For any two distinct variables $MM^s, MM^t \in \mathfrak{D}$, we have $MM^s \neq MM^t$ (with probability one).

Proof: This follows from the fact that the transcript $\boldsymbol{\tau}$ is allowed: Assume, for example, that $\mathsf{ty}^s = \mathsf{Enc}$ and $\mathsf{ty}^t = \mathsf{Dec}$ (the other cases are symmetric). Then $MM^s = \mathsf{C}^s_* \oplus (\mathsf{P}^s_{\mathsf{m}^s}10..0)$ and $MM^t = \mathsf{P}^t_* \oplus (\mathsf{C}^t_{\mathsf{m}^t}10..0)$, so $MM^s = MM^t$ implies $\mathsf{P}^t_* = \mathsf{C}^s_* \oplus (\mathsf{P}^s_{\mathsf{m}^s}10..0) \oplus (\mathsf{C}^s_{\mathsf{m}^t}10..0)$ which is not allowed. ▌

Claim 9 For any two distinct variables $PP^s_i, PP^t_{i'} \in \mathfrak{D}$, we have $\Pr[PP^s_i = PP^t_{i'}] \leq 2^{-n}$.

Proof: If $i \neq i'$ then we have

$$PP^s_i \oplus PP^t_{i'} = (\mathsf{P}^s_i \oplus 2^{i-1}L) \oplus (\mathsf{P}^t_{i'} \oplus 2^{i'-1}L) = \mathsf{P}^s_i \oplus \mathsf{P}^t_{i'} \oplus (2^i \oplus 2^{i'})L$$

and as $i \neq i'$, the coefficient of L is non-zero, and therefore $\Pr[PP^s_i \oplus PP^t_{i'} = 0^n] = 2^{-n}$. If $i = i'$ and $t < s$ then necessarily $\mathsf{P}^s_i \neq \mathsf{P}^t_{i'}$. (Otherwise, either query s is encipher, in which case $r[s,i] < s$ and $PP^s_i \notin \mathfrak{D}$, or query s is encipher, which means that the transcript $\boldsymbol{\tau}$ specifies an immediate collision.) Therefore, with probability one we have $PP^s_i \oplus PP^t_{i'} = (\mathsf{P}^s_i \oplus 2^{i-1}L) \oplus (\mathsf{P}^t{}_i \oplus 2^{i-1}L) = \mathsf{P}^s_i \oplus \mathsf{P}^t{}_i \neq 0$. ▌

Claim 10 For any two distinct variables $CCC^s_i, CCC^t_{i'} \in \mathfrak{D}$, we have $\Pr[CCC^s_i = CCC^t_{i'}] = 2^{-n}$.

Proof: By inspecting Figure 10, we see that for two variable $CCC^s_i, CCC^t_{i'} \in \mathfrak{D}$, if $s \neq t$ then $\phi(CCC^s_i) \neq \phi(CCC^t_{i'})$ and then by Claim 7 we get $\Pr[CCC^s_i = CCC^s_{i'}] = 2^{-n}$. On the other hand, if $s = t$ and $i \neq i'$ then $\Pr[CCC^s_i = CCC^s_{i'}] = 2^{-n}$ by Corollary 5. ▌

Claim 11 For any two variables $CCC^s_i, MP^t_j \in \mathfrak{D}$, $\Pr[CCC^s_i = MP^t_j] = 2^{-n}$.

Proof: From the definition of $\phi(\cdot)$ in Figure 10 it follows that:

- If s is a decipher query then $\phi(CCC_i^s) = CCC_i^s \neq \phi(MP_j^t)$.
- If s is an encipher query and $s > t$ then $\phi(CCC_i^s) = M_{j'}^s \neq \phi(MP_j^t)$, since MP_j^t cannot depend on anything that happens in a later query s.
- If s is an encipher query and $s < t$, then:
  - If t is a decipher query or $j > 1$, then $\phi(MP_j^t) = M_{j'}^t \neq \phi(CCC_i^s)$, since CCC_i^s cannot depend on anything that happens in a later query t.
  - If t is an encipher query and $j = 1$, then $\phi(MP_j^t)$ is of the form either $PPP_{i'}^r$ or $M_{j'}^r$, where $r \overset{\text{def}}{=} \mathsf{rmax}[t]$. If $r \neq s$ then clearly $\phi(MP_j^t) = XXX_*^r \neq YYY_*^s = \phi(CCC_i^s)$. (We use XXX_*^s, YYY_*^s to denote some free variables that are set in queries r, s, respectively.) But if $r = s$ then $\mathsf{ty}^r = \mathsf{Enc}$, so $\phi(MP_j^t) = PPP_{i'}^r \neq \phi(CCC_i^s)$.
- If s is an encipher query and $s = t$ and $j = 1$ then $\phi(MP_j^s)$ is either some $PPP_{i'}^{s'} \neq \phi(CCC_i^s)$ (rule MM3), or some M_j^r for $r < s$ (rule MM4) and again $\phi(MP_j^t) = M_j^r \neq YYY_*^s = \phi(CCC_i^s$.
- If s is an encipher query, $s = t$, $j > 1$, and $i > n$, then $\phi(CCC_i^s) = M_{\lceil i/n \rceil}^s \neq M_1^s = \phi(MP_j^t)$.
- If s is an encipher query and $s = t$ and $j > 1$ and $i = 1$, then $\phi(CCC_i^s) = M_{\lceil \mathsf{lastFull}^s/n \rceil}^s$ and $\phi(MP_j^t) = M_1^s$. But since $j > 1$ it must be that $\mathsf{lastFull}^s > n$, so $\phi(CCC_i^s) \neq \phi(MP_j^t)$.

In any of the cases above, we get $\Pr[CCC_i^s = MP_j^t] = 2^{-n}$ by Claim 7. The only case left to analyze is when $s = t$ is an encipher query, $j > 1$, and $1 < i \leq n$. In this case MP_j^s is assigned value in line 132, $MP_j^s \leftarrow PPP_{i_j}^s \oplus M_1^s$ ($i_j = jn - n + 1$), and CCC_i^s is assigned value in line 134, $CCC_i^s \leftarrow PPP_i^s \oplus 2^{i-1} M_1^s$. Hence

$$MP_j^s \oplus CCC_i^s = (PPP_{i_j}^s \oplus M_1^s) \oplus (PPP_i^s \oplus 2^{i-1} M_1^s) = PPP_{i_j}^s \oplus PPP_i^s \oplus (1 \oplus 2^{i-1}) M_1^s$$

The coefficient of M_1^s is $1 \oplus 2^{i-1} \neq 0$ (since $i > 1$), so the sum $MP_j^s \oplus CCC_i^s$ depends linearly on M_1^s and $\Pr[CCC_i^s = MP_j^s] = 2^{-n}$. ▮

The most involved case to analyze (indeed, the one that embodies the "real reason" that EME* is secure) is collisions of the type $MP_j^s = MP_{j'}^t$. We break the analysis of these collisions into the following three claim: in Claim 12 we analyze the case $s = t$, in Claim 13 we analyze the case $s \neq t$ and either j or j' are different than one, and in Claim 13 we analyze the (hardest) case where $s \neq t$ and $j = j' = 1$.

Claim 12 For any two distinct variables $MP_j^s, MP_{j'}^s \in \mathfrak{D}$, belonging to the same query s, it holds that $\Pr[MP_j^s = MP_{j'}^s] \leq 2^{-n}$.

Proof: Assume w.l.o.g. that $j' > j$. From Figure 10 we see that if $\mathsf{ty}^s = \mathsf{Dec}$ then we would have $\phi(MP_{j'}^s) = M_{j'}^s \neq M_j^s = \phi(MP_j^s)$. Also, if $\mathsf{ty}^s = \mathsf{Enc}$ and $j' > j = 1$, then $\phi(MP_{j'}^s) = M_1^s$, but $\phi(MP_1^s)$ is either some PPP_*^r, or else it is some M_*^r for an earlier query $r < s$. (The latter case corresponds to rule MM4 from Figure 10, and we cannot have $r = s$ since $\mathsf{ty}^s = \mathsf{Enc}$ but $\mathsf{ty}^r = \mathsf{Dec}$.) In any of these cases, we get $\phi(MP_{j'}^s) \neq \phi(MP_j^s)$ and by Claim 7 $\Pr[MP_j^s = MP_{j'}^s] = 2^{-n}$.

We are left with the case where $\mathsf{ty}^s = \mathsf{Enc}$, and $j' > j > 1$. Hence both $MP_j^s, MP_{j'}^s$ were assigned values in line 132 of Game N2, so

$$MP_j^s \oplus MP_{j'}^s = (PPP_i^s \oplus M_1^s) \oplus (PPP_{i'}^s \oplus M_1^s) = PPP_i^s \oplus PPP_{i'}^s$$

(with $i = jn - n + 1$ and $i' = j'n - n + 1$). By Corollary 5 (with $I = \{i, i'\}$), we have $\Pr[MP_j^s = MP_{j'}^s] = \Pr[PPP_i^s \oplus PPP_{i'}^s = 0] = 2^{-n}$. ▌

Claim 13 For any two distinct variables $MP_j^s, MP_{j'}^t \in \mathfrak{D}$, such that $s \neq t$ and at least one of j, j' is *not equal* to one, it holds that $\Pr[MP_j^s = MP_{j'}^t] \leq 2^{-n}$.

Proof: Assume w.l.o.g. that $s < t$. We observe the following from Figure 10:

- If $\mathsf{ty}^t = \mathsf{Dec}$ or $j' > 1$ then $\phi(MP_{j'}^t) = M_{j''}^t$ (for some j''), but MP_j^s cannot depend on $M_{j''}^t$ which is only determined when processing query $t > s$. Hence $\phi(MP_j^s) \neq \phi(MP_{j'}^t)$.
- If $\mathsf{ty}^t = \mathsf{Enc}$ and $j' = 1$ (so $j > 1$) and $r \overset{\text{def}}{=} \mathsf{rmax}[t] \neq s$, then $\phi(MP_{j'}^t)$ is either some PPP_*^r or some M_*^r whereas $\phi(MP_j^s) = M_*^s$, so again $\phi(MP_j^s) \neq \phi(MP_{j'}^t)$.
- If $\mathsf{ty}^t = \mathsf{Enc}$ and $j' = 1$ (so $j > 1$) and $r \overset{\text{def}}{=} \mathsf{rmax}[t] = s$ and $\mathsf{ty}^s = \mathsf{Enc}$, then $\phi(MP_j^s) = M_j^s \neq PPP_*^s = \phi(MP_{j'}^t)$.

In either of these cases we get $\Pr[MP_j^s = MP_{j'}^s] = 2^{-n}$ by Claim 7.

The only case left to analyze for this claim is when $s < t$, $\mathsf{ty}^t = \mathsf{Enc}$, $j' = 1$ (so $j > 1$), $r \overset{\text{def}}{=} \mathsf{rmax}[t] = s$, and $\mathsf{ty}^s = \mathsf{Dec}$. In this case $MP_{j'}^t = MP_1^t$ was assigned value in line 120 of Game N2, $MP_1^t \leftarrow h(\mathsf{T}^t) \oplus \sum_{i=1}^{\mathsf{m}^t} PPP_i^t$, and MP_j^s was assigned value in line 232 in game N2, $MP_j^s \leftarrow CCC_{i_j}^s \oplus M_1^s \oplus M_j^s$ (where $i_j = jn - n + 1 > 1$). Hence we get

$$MP_1^t \oplus MP_j^s \quad = \quad (h(\mathsf{T}^t) \oplus \sum_{i=1}^{\mathsf{m}^t} PPP_i^t) \oplus (CCC_{i_j}^s \oplus M_1^s \oplus M_j^s)$$

Inspecting the code of game N2, we see that the all the PPP_i^r's, CCC_i^r's and M_j^r's are independent of the choice of the function h. Hence by property (i) from Claim 2 we get

$$\Pr[MP_1^t = MP_j^s] = \Pr_h \left[h(\mathsf{T}^t) = CCC_{i_j}^s \oplus M_1^s \oplus M_j^s \oplus \sum_{i=1}^{\mathsf{m}^t} PPP_i^t \right] = 2^{-n}$$

▌

Claim 14 For any two queries $s \neq t$, it holds that $\Pr[MP_1^s = MP_1^t] \leq 2^{-n}$.

Proof: We again assume w.l.o.g. that $s < t$. As in Claim 13, we observer that if $\mathsf{ty}^t = \mathsf{Dec}$ or $\mathsf{rmax}[t] \neq s$ then $\phi(MP_1^t) \neq \phi(MP_1^s)$ and we are done by Claim 7. So from now on we assume that $\mathsf{ty}^t = \mathsf{Enc}$ and $\mathsf{rmax}[t] = s$.

Recall that since the transcript $\boldsymbol{\tau}$ is *allowed*, we know that either $\mathsf{T}^s \neq \mathsf{T}^t$ or $\mathsf{P}^s \neq \mathsf{P}^t$. We start by analyzing the case where $\mathsf{T}^s \neq \mathsf{T}^t$. Observe that regardless of the direction of queries s, t, it holds that $MP_1^s = h(\mathsf{T}^s) \oplus \sum_{i=1}^{\mathsf{m}^s} PPP_i^s$ and $MP_1^t = h(\mathsf{T}^t) \oplus \sum_{i=1}^{\mathsf{m}^s} PPP_i^t$. Thus

$$\begin{aligned} MP_1^t \oplus MP_1^s &= h(\mathsf{T}^t) \oplus \sum_{i=1}^{\mathsf{m}^t} PPP_i^t \oplus h(\mathsf{T}^s) \oplus \sum_{i=1}^{\mathsf{m}^s} PPP_i^s \\ &= h(\mathsf{T}^t) \oplus h(\mathsf{T}^s) \oplus \text{things-that-are-independent-of-}h \end{aligned}$$

and since $\mathsf{T}^s \neq \mathsf{T}^t$, we have $\Pr[MP_1^t = MP_j^s] = 2^{-n}$ by property (ii) from Claim 2.

Next we analyze the case where $\mathsf{T}^s = \mathsf{T}^t$ and $\mathsf{P}^s \neq \mathsf{P}^t$. To facilitate treatment of queries with partial blocks, let us denote for all r

$$Y^r = \begin{cases} PPP^r_{\mathsf{m}^r} & \text{if } |\mathsf{P}^r_{\mathsf{m}^r}| < n \\ 0 & \text{if } |\mathsf{P}^r_{\mathsf{m}^r}| = n \end{cases}$$

and note that Y^r is a constant, depending only on $\mathsf{P}^r_{\mathsf{m}^r}$, and that if $\mathsf{P}^s_{\mathsf{m}^s} \neq \mathsf{P}^t_{\mathsf{m}^t}$ then $Y^s \neq Y^t$. Then we can write

$$\begin{aligned} MP_1^s \oplus MP_1^t &= h(\mathsf{T}^s) \oplus \sum_{i=1}^{\mathsf{lastFull}^s} PPP_i^s \oplus Y^s \oplus h(\mathsf{T}^t) \oplus \sum_{i=1}^{\mathsf{lastFull}^t} PPP_i^t \oplus Y^t \\ &= Y^s \oplus Y^t \oplus \sum_{i=1}^{\mathsf{lastFull}^s} PPP_i^s \oplus \sum_{i=1}^{\mathsf{lastFull}^t} PPP_i^t \end{aligned} \tag{12}$$

An easy sub-case is where P^s and P^t agree on all the "full blocks". That is, denote $\tilde{\mathsf{P}}^s \stackrel{\text{def}}{=} \mathsf{P}_1^s \dots \mathsf{P}^s_{\mathsf{lastFull}^s}$ and $\tilde{\mathsf{P}}^t \stackrel{\text{def}}{=} \mathsf{P}_1^t \dots \mathsf{P}^t_{\mathsf{lastFull}^t}$, and we analyze the case where $\tilde{\mathsf{P}}^s = \tilde{\mathsf{P}}^t$. Since $\mathsf{P}^s \neq \mathsf{P}^t$, it must be the case where they differ in their last block, namely $\mathsf{P}^s_{\mathsf{m}^s} \neq \mathsf{P}^t_{\mathsf{m}^t}$, which means that $Y^s \neq Y^t$. In this case we have $\sum_{i=1}^{\mathsf{lastFull}^s} PPP_i^s = \sum_{i=1}^{\mathsf{lastFull}^t} PPP_i^t$ and therefore $MP_1^s \oplus MP_1^t = Y^s \oplus Y^t \neq 0$ with probability one.

For the rest of this proof we assume that $\mathsf{T}^s = \mathsf{T}^t$, and $\tilde{\mathsf{P}}^s \neq \tilde{\mathsf{P}}^t$. Let E be the set of indexes where $\tilde{\mathsf{P}}^s$ and $\tilde{\mathsf{P}}^t$ agree, $E \stackrel{\text{def}}{=} \{\, i \leq \min(\mathsf{lastFull}^t, \mathsf{lastFull}^s) \;\mid\; \mathsf{P}_i^s = \mathsf{P}_i^t\}$. We can re-write Eq. (12) as

$$MP_1^s \oplus MP_1^t = Y^s \oplus Y^t \oplus \sum_{i \leq \mathsf{lastFull}^s, i \notin E} PPP_i^s \oplus \sum_{i \leq \mathsf{lastFull}^t, i \notin E} PPP_i^t \tag{13}$$

where the equality is justified since $\mathsf{P}_i^s = \mathsf{P}_i^t$ implies $PPP_i^s = PPP_i^t$. Recall that we assume $\mathsf{ty}^t = \mathsf{Enc}$ and $\mathsf{rmax}[t] = s$, and we now distinguish again between two sub-cases:

First sub-case: here we assume that $\mathsf{ty}^s = \mathsf{Dec}$ and furthermore $\tilde{\mathsf{P}}^s$ is not a prefix of $\tilde{\mathsf{P}}^t$. Since $\mathsf{rmax}[t] = s$, then all the blocks P_i^t already appeared in queries no later than s, namely $\mathsf{r}[t,i] \leq s$ for all $i \in [1 \,..\, \mathsf{lastFull}^t]$. Since for $i \notin E$ we have $\mathsf{P}_i^s \neq \mathsf{P}_i^t$, it follows that for these indexes $\mathsf{r}[t,i] < s$. Thus we get

$$\begin{aligned} MP_1^s \oplus MP_1^t &= Y^s \oplus Y^t \oplus \sum_{i \leq \mathsf{lastFull}^s, i \notin E} PPP_i^s \oplus \sum_{i \leq \mathsf{lastFull}^t, i \notin E} PPP_i^t \\ &= Y^s \oplus Y^t \oplus \sum_{i \leq \mathsf{lastFull}^s, i \notin E} PPP_i^s \oplus \sum_{i \leq \mathsf{lastFull}^t, i \notin E} PPP_i^{\mathsf{r}[t,i]} \\ &= \text{things-that-were-determined-before-query-}s \oplus \sum_{i \leq \mathsf{lastFull}^s, i \notin E} PPP_i^s \end{aligned} \tag{14}$$

Since $\tilde{\mathsf{P}}^s$ is not a prefix of $\tilde{\mathsf{P}}^t$, then the set $D_s \stackrel{\text{def}}{=} \{i \leq \mathsf{lastFull}^s \mid \; i \notin E\}$ is non-empty. And since query s is decipher, we can apply Claim 4 to conclude that $\sum_{i \in D} PPP_i^s = \alpha MP^s_{j(D_s)} \oplus \beta$ where $\alpha \neq 0$ and β depends only on things that were determined before $MP^s_{j(D_s)}$. Combining this with

Eq. (14) we conclude that $MP^s_1 \oplus MP^t_1 = \alpha MP^s_{j(D_s)} \oplus \beta'$ for the same non-zero constant α, where β' is a different expression, but it still depends only on things that were determined before $MP^s_{j(D_s)}$. Therefore, $\Pr[MP^s_1 = MP^t_1] = 2^{-n}$.

Second sub-case: Next we analyze the cases where either query s is encipher, $\mathsf{ty}^s = \mathsf{Enc}$, or $\tilde{\mathsf{P}}^s$ is a (proper) prefix of $\tilde{\mathsf{P}}^t$. 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 Eq. (13) as

$$MP^s_1 \oplus MP^t_1 \;=\; Y^s \oplus Y^t \;\oplus \sum_{i \le \mathsf{lastFull}^s, i \notin E} PPP^{\mathsf{r}[s,i]}_i \;\oplus \sum_{i \le \mathsf{lastFull}^t, i \notin E} PPP^{\mathsf{r}[t,i]}_i \tag{15}$$

(In the case that query s is decipher and $\tilde{\mathsf{P}}^s$ is a proper prefix of $\tilde{\mathsf{P}}^t$, the equality follows since the summation on $i \le \mathsf{lastFull}^s, i \notin E$ ranges over an empty set.) Recall that by definition we have $\mathsf{r}[s,i] = \mathsf{r}[t,i]$ iff $P^s_i = P^t_i$ iff $i \in E$.

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

$$\begin{aligned}
R^* &\stackrel{\text{def}}{=} \{\mathsf{r}[s,i] \;\mid\; i \le \mathsf{lastFull}^s,\ i \notin E\} \;\cup\; \{\mathsf{r}[t,i] \;\mid\; i \le \mathsf{lastFull}^t,\ i \notin E\},\\
r^* &\stackrel{\text{def}}{=} \max(R),\\
I^*_s &\stackrel{\text{def}}{=} \{\ i \le \mathsf{lastFull}^s \;\mid\; i \notin E,\ \mathsf{r}[s,i] = r^*\ \},\\
I^*_t &\stackrel{\text{def}}{=} \{\ i \le \mathsf{lastFull}^t \;\mid\; i \notin E,\ \mathsf{r}[t,i] = r^*\ \},\\
I^* &\stackrel{\text{def}}{=} I^*_s \cup I^*_t
\end{aligned}$$

From this definition it follows that the sets I^*_s, I^*_t are disjoint (since $\mathsf{r}[s,i] \ne \mathsf{r}[t,i]$ for $i \notin E$), and their union is non-empty (since R^* is non-empty). Using these notation we can rewrite Eq. (15)

$$\begin{aligned}
MP^s_1 \oplus MP^t_1 \;&=\; Y^s \oplus Y^t \;\oplus\; \left(\sum_{i \in I^*_s} PPP^{\mathsf{r}[s,i]}_i \;\oplus \sum_{i \le \mathsf{lastFull}^s, i \notin (E \cup I^*_s)} PPP^{\mathsf{r}[s,i]}_i\right) \qquad (16)\\
&\qquad \oplus\; \left(\sum_{i \in I^*_t} PPP^{\mathsf{r}[t,i]}_i \;\oplus \sum_{i \le \mathsf{lastFull}^t, i \notin (E \cup I^*_t)} PPP^{\mathsf{r}[t,i]}_i\right)\\
&\qquad =\; \text{things-that-were-determined-before-query-}r^* \;\oplus \sum_{i \in I^*} PPP^{r^*}_i
\end{aligned}$$

If query r^* is decipher, $\mathsf{ty}^{r^*} = \mathsf{Dec}$, we can use Claim 4 to conclude that $\sum_{i \in I^*} PPP^{r^*}_i = \alpha MP^{r^*}_{j(I^*)} \oplus \beta$ where $\alpha \ne 0$ and β only depends on things that are determined before $MP^{r^*}_{j(I^*)}$, and since $MP^{r^*}_{j(I^*)}$ is a free variable, it follows that $\Pr[MP^s_1 = MP^t_1] = 2^{-n}$. If query r^* is encipher, $\mathsf{ty}^r = \mathsf{Enc}$, then all the variables $PPP^{r^*}_i$, $i \in I^*$, are free variables, and again we have $\Pr[MP^s = MP^t] \le 2^{-n}$. ▌

As an immediate corollary from the last three claims we have

Corollary 15 For any two distinct variables $MP^s_j, MP^t_{j'} \in \mathfrak{D}$, $\Pr[MP^s_j = MP^s_{j'}] \le 2^{-n}$. □

Proof of Claim 3. All that is left now is to verify that Claim 7 through Corollary 15 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.

$X =$ **"MM^s".** Here either $X' =$ "MM^t" in which case $\Pr[X = X'] = 0$ by Claim 8, or else $\phi(X') \neq \textsf{none} = \phi(X)$ and we have $\Pr[X = X'] = 2^{-n}$ from Claim 7.

$X =$ **"PP_i^s".** Similarly to the previous case, either $X' =$ "$PP_{i'}^{s'}$" and we have $\Pr[X = X'] = 2^{-n}$ by Claim 9, or else $\phi(X') \neq L = \phi(X)$ and we have the same using Claim 7.

$X =$ **"CCC_i^s".** If $X' =$ "MM^t" or $X' =$ "$MP_{i'}^t$" then $\phi(X') \neq \phi(X)$ and we get $\Pr[X = X'] = 2^{-n}$ from Claim 7. If $X' =$ "$CCC_{i'}^t$" then we get he same from Claim 10, and if $X' =$ "MP_j^t" then we get he same from Claim 11.

$X =$ **"MP_j^s".** If $X' =$ "MM^t" or $X' =$ "$MP_{i'}^t$" then $\phi(X') \neq \phi(X)$ and we get $\Pr[X = X'] = 2^{-n}$ from Claim 7. If $X' =$ "CCC_i^t" then we get he same from Claim 11, and if $X' =$ "$MP_{j'}^t$" then we get he same from Corollary 15.

This completes the proof of Claim 3. ▌