A New Mode of Encryption Secure Against Symmetric Nonce Respecting Adversaries

Debrup Chakraborty and Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute
203 B.T. Road
Kolkata 700108, India
email: {debrup_r, palash}@isical.ac.in

Abstract

We present MEM, which is a new mode of encryption using a block cipher. MEM is proved to be a strong pseudo-random permutation (SPRP) against *symmetric* nonce respecting adversaries, where a symmetric nonce respecting adversary is one which does not repeat nonces to either the encryption or the decryption oracle. Against such adversaries, MEM provides a secure, length preserving, tagless mode of encryption. In our construction, the number of block cipher calls is approximately half that of the earlier known more general constructions CMC, EME and EME* of tweakable SPRPs. In situations where the appropriate adversary can be assumed, and where a tagless mode of encryption is required, our construction is the most efficient solution till date.
Keywords: mode of operation, nonce based encryption, strong pseudo-random permutation.

1 Introduction

A block cipher is a fundamental primitive in cryptography. A block cipher by itself can encrypt only fixed length strings. Applications in general require encryption of long and arbitrary length strings. A mode of operation of a block cipher is used to extend the domain of applicability from fixed length strings to long and variable length strings. The mode of operation must be secure in the sense that if the underlying block cipher satisfies a certain notion of security, then the extended domain mode of operation also satisfies an appropriate notion of security.

A formal model of security for a block cipher is a pseudo-random permutation [7] which is formalized as a keyed family of permutations. Pseudo-randomness of the permutation family requires a computationally bounded adversary to be unable to distinguish between a random permutation and a permutation picked at random from the family. Strong pseudo-random permutations (SPRPs) require computational indistinguishability even when the adversary has access to the inverse permutations. Tweakable encryption was introduced by Liskov, Rivest and Wagner [6] which added an extra input called tweak to a block cipher. This allows simplification of several applications.

A mode of encryption usually provides two security assurances – privacy and authenticity. For example, OCB [11] is a mode of operation (providing both privacy and autheticity) which extends the domain of a block-cipher to arbitrary strings. It proves privacy by showing that the extended domain construction is a PRP if the underlying block cipher is a PRP. Authentication of OCB is

also proved under the same assumption. On the other hand, the constructions CMC, EME and EME* [3, 4, 2] are proved to be tweakable SPRPs under the assumption that the underlying block cipher is an SPRP.

An SPRP (or a tweakable SPRP) which can encrypt arbitrary length strings, can be viewed as a mode of operation with a somewhat different goal. Such a mode of operation is length preserving and no tag is produced. Hence, authentication is of limited nature. A change in the ciphertext cannot be detected but the decryption of the tampered ciphertext will result in a plaintext which is indistinguishable from a random string. Additional redundancy in the message introduced by higher level applications might even allow the detection of the tampering. This point is discussed in details by Bellare and Rogaway [1].

Another important notion discussed in [1] and in the more recent paper by Rogaway [10] is that of nonce based symmetric cryptography. Nonces have been present in cryptographic constructions for quite a long time. They are used as initialization vectors for stream ciphers, modes of operations (counter mode), and have various other applications. A systematic treatment of nonces, their security implications and possible efficiency improvements in using them are discussed in the above two papers.

A nonce respecting adversary is one which does not repeat a nonce in querying the encryption oracle. Consideration of nonce respecting adversaries require the sender to be stateful. As the nonce is also required for decryption, it has to be either transmitted to the receiver or requires a synchronized generation of nonces between sender and receiver. In the later case, the decryptor also has to maintain state. In any case, the nonce is not considered to be a part of the ciphertext.

Our security model assumes a *symmetric* nonce respecting adversary, i.e., an adversary who is nonce respecting while querying both encryption and decryption oracles. In other words, the additional requirement over nonce respecting adversary is that it does not repeat the same nonce to its decryption oracle. Note that an adversary is allowed to present the same nonce to both the encryption and the decryption oracles.

Symmetric nonce respecting adversary is a meaningful assumption for certain scenarios. Consider the following protocol: Two users A and B share a common session key. They alternately send messages to each other, starting with user A. Hence, A sends the messages numbered $1, 3, 5 \ldots$ to B, and B sends messages numbered $2, 4, 6, \ldots$ to A. The message numbers are used as nonces. In such a situation, users A and B keep a record of the message numbers they have already sent or received. As a result, an adversary cannot repeat a message number to either of the users A or B, as such repetition will immediately be detected. Hence, any adversary is forced to be symmetric nonce respecting.

Our Contributions: We present a new mode of operation called MEM. MEM uses a block cipher which can encrypt an n-bit string to construct an encryption algorithm which can encrypt mn-bit strings for any $m \geq 1$. The main novelty of MEM is that it is proved to be secure against symmetric nonce respecting adversaries. This is proved by showing that if the underlying block cipher is an SPRP, then MEM *provides an SPRP against symmetric nonce respecting adversaries.* Hence, MEM provides a length preserving, tagless mode of encryption. The authentication guarantee is that provided by an SPRP.

The security of MEM comes somewhere in between previously known constructions. Modes of operations like OCB provide PRP against nonce respecting adversaries. (Note that a PRP against nonce respecting adversaries is not necessarily an SPRP against symmetric nonce respecting adversaries.) On the other hand, the more powerful modes of encryptions like CMC, EME, EME*

Table 1: Security and efficiency of different modes of encryption. Here m is the number of blocks in the message.

mode of operation	adversary restriction	security		no. of block cipher calls
		priv.	auth.	
OCB [11]	nonce respecting	PRP	tag	$\approx m$
CMC [3], EME* [2]	arbitrary tweak	SPRP		$\approx 2m$
MEM	sym. nonce respecting	SPRP		$\approx m$

are proved to be tweakable SPRP against adversaries which can repeat the tweaks in an arbitrary manner. Hence, these are certainly SPRP against symmetric nonce respecting adversaries. On the other hand, to the best of our knowledge, there is no known mode of operation with efficiency similar to that of MEM and which can be proved to be an SPRP against symmetric nonce respecting adversaries. In Table 1, we relate the efficiency and security model of MEM with other constructions.

2 Specification of MEM

We construct a nonce based enciphering scheme MEM (Mask Encrypt Mask) from a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and call it MEM[E]. The key space of MEM[E] is same as that of the underlying block cipher E and the nonce space is $\mathcal{N} = \{0,1\}^n$. The message space consists of all binary strings of size mn where $m \geq 1$.

An n-bit string can also be viewed as an element in $GF(2^n)$. Thus, we will consider each n-bit string in the specification of MEM as a polynomial over $GF(2)$ modulo a fixed sparse irreducible polynomial $\tau(x)$ of degree n. It is usually possible to choose $\tau(x)$ to be a trinomial or a pentanomial. In such cases, multiplying a polynomial by x^i, $0 \leq i \leq n-1$, modulo $\tau(x)$ is particularly efficient and can be achieved by a few bitwise shifts and XORs. The expression $p_i(x)M_1$ represents multiplication of the polynomials $p_i(x)$ and M_1 modulo $\tau(x)$. Similarly xEN denotes the multiplication of x and EN mod $\tau(x)$.

The specification of MEM consists of three cases: $m = 1$; $m = 2$; and $m > 2$. The cases $m = 1$ and $m = 2$ are shown in Figure 2. Figure 3 shows the encryption of a 4 block message. The complete encryption and decryption algorithms are shown in Figure 1.

The cases $m = 1$ and $m = 2$ are straightforward and do not require any explanation. We consider the case $m > 2$. Note that a plaintext block P_i is masked by $p_i(x)M_1$ to obtain the block PP_i. Similarly, CC_i is masked by $p_i(x)M_2$ to obtain C_i. First we explain how M_1 and M_2 are obtained. Later we will explain what $p_i(x)$'s are.

Initially, M_1 is obtained by encrypting MPP, which is the XOR of all the plaintext blocks. After each subsequent $(n+1)$ plaintext blocks have been processed, the value of M_1 is updated by $M_1 = E_K(M_1)$. We will denote the first value of M_1 by $M_{1,0}$ and the subsequent values by $M_{1,1}, M_{1,2}, \ldots$. A similar strategy is used to update the mask M_2 and we will use $M_{2,0}, M_{2,1}, M_{2,2}, \ldots$ to denote the different values of M_2.

We next define the polynomials $p_i(x)$. For $1 \leq i \leq n$, define $\alpha_i(x) = x^{i-1} \oplus x^i$; and $\alpha_{n+1}(x) = x^n \oplus 1$. For $1 \leq i \leq m-1$, define $p_i(x) = \alpha_j(x)$, $j = (i-1) \bmod (n+1)+1$; and $p_m(x) = x^{m-1} \oplus 1$. With this definition, the following two properties hold.

1. Let $m = m_1 \times (n+1) + m_2$, where $m_2 = m \bmod (n+1)$. Then for $0 \leq k \leq m_1$, $\bigoplus_{i=1}^{n+1} p_{k(n+1)+i}(x) = 0$ and $\bigoplus_{i=m_1(n+1)}^{m} p_i(x) = 0$.
2. If $\lfloor i/(n+1) \rfloor = \lfloor j/(n+1) \rfloor$, then degree of $p_i(x) \oplus p_j(x)$ is at most n. Consequently, for such i and j, $\tau(x) \not| (p_i(x) \oplus p_j(x))$.

We explain where these properties are required. Assume that $m > 2$ and note that during encryption, the mask M_2 is obtained as $M_2 = E_K(CC_1 \oplus \cdots \oplus CC_m \oplus E_K(xE_K(N)))$. During decryption, N (and hence $E_K(xE_K(N))$) will be known. However, the quantities $CC_1, \ldots, CC_m$ will not be known. On the other hand, to compute M_2 we need to know $CC_1 \oplus \cdots \oplus CC_m$. This can be done in the following manner. As before, let $m = m_1 \times (n+1) + m_2$, where $m_2 = m \bmod (n+1)$. Since $C_i = CC_i \oplus p_i(x)M_{2,k}$, where $k = \lfloor i/(n+1) \rfloor$, we have using the first property,

$$\begin{aligned}
C_1 \oplus \cdots \oplus C_m &= CC_1 \oplus \cdots \oplus CC_m \bigoplus \\
&\quad \left(\bigoplus_{k=0}^{m_1-1} \bigoplus_{i=1}^{n+1} p_{k(n+1)+i}(x) M_{2,k} \right) \bigoplus \left(\bigoplus_{i=m_1(n+1)}^{m} p_i(x) M_{2,m_1} \right) \\
&= CC_1 \oplus \cdots \oplus CC_m \bigoplus \\
&\quad \left(\bigoplus_{k=0}^{m_1-1} M_{2,k} \bigoplus_{i=1}^{n+1} p_{k(n+1)+i}(x) \right) \bigoplus \left(M_{2,m_1} \bigoplus_{i=m_1(n+1)}^{m} p_i(x) \right) \\
&= CC_1 \oplus \cdots \oplus CC_m.
\end{aligned}$$

The last equality holds due to the definition of the $p_i(x)$'s. Since the C_i's are known, the XOR of the CC_i's can be computed from the XOR of the C_i's. The second property is required in the security proof to rule out certain kinds of collisions (see the comments on Table 3 in Section A.).

3 Features of MEM

Here we discuss some of the important features and limitations of MEM.

Message Length: MEM does not produce any ciphertext expansion as it does not produce any tag. Similar to OCB [11] the nonce is not considered as a part of the ciphertext. The current version of MEM can only handle messages whose length is a multiple of n. Modification of MEM to handle arbitrary length messages is a future task.

Single Block Cipher Key: MEM uses the same key for all the block cipher calls. IACBC [5] and its parallel version IAPM [5] both require two different keys. OCB which is a refinement of IAPM requires a single block cipher key. The more general constructions of tweakable SPRPs CMC [3] and EME [4] require a single key. On the other hand, the more recent construction, EME*, which is a tweakble SPRP capable of handling arbitrary length messages, requires three keys. A single block cipher key saves key storage space and key setup costs.

Figure 1: Encryption and Decryption using MEM

Algorithm $\mathbf{E}_K^N(P_1, P_2, \ldots, P_m)$

$EN = E_K(N)$;
$EEN = E_K(xEN)$;
$MP = P_1 \oplus P_2 \oplus \ldots \oplus P_m$;

if $m == 1$, **then**
  $MPP = MP \oplus EN$;
  $M_1 = E_K(MPP)$;
  $C_1 = M_1 \oplus xEEN$;
  return C_1;
endif

if $m == 2$, **then**
  $MPP = MP \oplus EN$;
  $M_1 = E_K(MPP)$;
  $PP_1 = M_1 \oplus P_1$;
  $PP_2 = M_1 \oplus EEN \oplus P_2$;
  $CC_1 = E_K(PP_1)$;
  $CC_2 = E_K(PP_2)$;
  $MC = CC_1 \oplus CC_2$;
  $MCC = MC \oplus EEN$;
  $M_2 = E_K(MCC)$;
  $C_1 = M_2 \oplus CC_1$;
  $C_2 = M_2 \oplus EN \oplus CC_2$;
  return C_1, C_2;
endif

if $m > 2$, **then**
  $MPP = MP \oplus EN$;
  $M_1 = E_K(MPP)$; $MC = 0^n$;
  for $i = 1$ to m,
    if ($i - 1 > 0$ **and** $i - 1 \bmod (n+1) == 0$)
      $M_1 = E_K(M_1)$;
    endif
    $PP_i = P_i \oplus p_i(x)M_1$;
    $CC_i = E_K(PP_i)$; $MC = MC \oplus CC_i$;
  endfor
  $MCC = MC \oplus EEN$;
  $M_2 = E_K(MCC)$;
  for $i = 1$ to m,
    if ($i - 1 > 0$ **and** $i - 1 \bmod (n+1) == 0$)
      $M_2 = E_K(M_2)$;
    endif
    $C_i = CC_i \oplus p_i(x)M_2$;
  endfor
  return $C_1, C_2, \ldots, C_m$;
endif

Algorithm $\mathbf{D}_K^N(C_1, C_2, \ldots, C_m)$

$EN = E_K(N)$;
$EEN = E_K(xEN)$;
$MC = C_1 \oplus C_2 \oplus \ldots \oplus C_m$;

if $m == 1$, **then**
  $MCC = MC \oplus xEEN$;
  $M_2 = E_K^{-1}(MCC)$;
  $P_1 = M_2 \oplus EN$
  return P_1;
endif

if $m == 2$, **then**
  $MCC = MC \oplus EN \oplus EEN$;
  $M_2 = E_K(MCC)$;
  $CC_1 = M_2 \oplus C_1$;
  $CC_2 = M_2 \oplus EN \oplus C_2$;
  $PP_1 = E_K^{-1}(CC_1)$;
  $PP_2 = E_K^{-1}(CC_2)$;
  $MP = PP_1 \oplus PP_2 \oplus EEN$;
  $MPP = MP \oplus EN$;
  $M_1 = E_K(MPP)$;
  $P_1 = PP_1 \oplus M_1$;
  $P_2 = PP_2 \oplus M_1 \oplus EEN$;
  return P_1, P_2;
endif

if $m > 2$, **then**
  $MCC = MC \oplus EEN$;
  $M_2 = E_K(MCC)$; $MP = 0^n$;
  for $i = 1$ to m,
    if ($i - 1 > 0$ **and** $i - 1 \bmod (n+1) == 0$)
      $M_2 = E_K(M_2)$;
    endif
    $CC_i = C_i \oplus p_i(x)M_2$;
    $PP_i = E_K^{-1}(CC_i)$; $MP = MP \oplus PP_i$;
  endfor
  $MPP = MP \oplus EN$;
  $M_1 = E_K(MPP)$;
  for $i = 1$ to m,
    if ($i - 1 > 0$ **and** $i - 1 \bmod (n+1) = 0$)
      $M_1 = E_K(M_1)$;
    endif
    $P_i = PP_i \oplus p_i(x)M_1$;
  endfor
  return $P_1, P_2, \ldots, P_m$;
endif

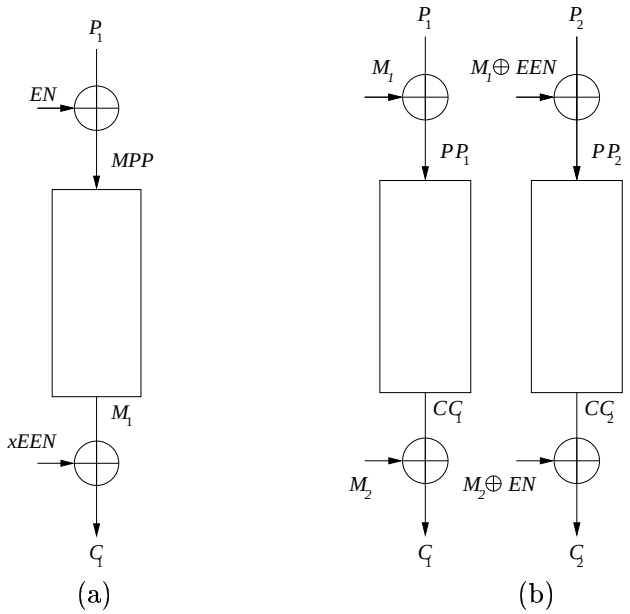

Figure 2: (a) Enciphering 1 plaintext block with MEM, (b) Enciphering 2 plaintext blocks with MEM.

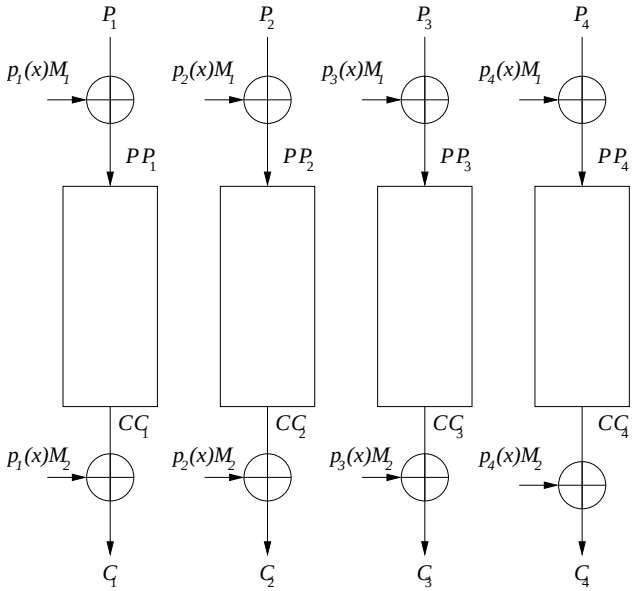

Figure 3: Enciphering 4 plaintext blocks with MEM. The boxes represents E_K, $M_1 = E_K(P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus E_K(N))$, $M_2 = E_K(CC_1 \oplus CC_2 \oplus CC_3 \oplus CC_4 \oplus E_K(xE_K(N)))$, N is the nonce.

Nonce: Encryption under MEM requires an n-bit nonce. The nonce need not be random, unpredictable or secret. Security is maintained even if the adversary has control over the nonce with the only restriction that the adversary cannot repeat nonces. The nonce is required for both encryption and decryption. Thus the nonce should be communicated along with the ciphertext or can be maintained as a shared counter between the two parties, or may be such that it can be understood from the context. Here we do not consider the nonce to be a part of the ciphertext.

Nonces have been previously used in many constructions as in OCB [11]. In [10], Rogaway formalizes nonce based encryptions. A nonce based scheme is viewed as a deterministic encryption scheme where the duty of the user is to supply unique IV's (the nonce) for each encryption and thus make the encryption process probabilistic/stateful.

Rogaway in [10] comments that the receiver may be stateless, i.e., the party encrypting a message is responsible for providing fresh nonces but the adversary may not be nonce respecting while it queries the decryption oracle. But, if the nonce is generated by means of a shared counter, then the receiver also becomes stateful and it becomes necessary for the receiver to maintain state in order to decrypt properly. Thus, restricting the adversary to be nonce respecting also while it queries the decryption oracle is meaningful in certain situations. Our security model considers such cases where the adversary is nonce respecting while it queries both the encryption and decryption oracles. However, an adversary is allowed to present the same nonce to the encryption and the decryption oracles.

The security of MEM depends on the adversary being symmetric nonce respecting. We now show that an adversary who is not nonce respecting can easily distinguish MEM from a random permutation. Let us consider the adversary provides two encryption queries $(N^s; P_1^s, P_2^s, P_3^s, P_4^s)$ and $(N^t; P_1^t, P_2^t, P_3^t, P_4^t)$. Where, $N^s = N^t$, $P_1^s = P_1^t$, $P_2^s = P_2^t$ and $P_3^s \oplus P_4^s = P_3^t \oplus P_4^t$. Note these queries are not nonce respecting. As a response to these queries he receives $C_1^s, C_2^s, C_3^s, C_4^s$ and $C_1^t, C_2^t, C_3^t, C_4^t$. He then computes the following:

$$X_1 = p_1(x)^{-1}(C_1^s \oplus C_1^t)$$

and

$$X_2 = p_2(x)^{-1}(C_2^s \oplus C_2^t).$$

If he finds $X_1 = X_2$, then he concludes that the output is from MEM. This works because for the queries specified above $PP_1^s = PP_1^t$ and also $CC_1^s = CC_1^t$, as the masks M_1^s and M_1^t would be equal. Also,

$$M_2^s = p_1(x)^{-1}(CC_1^s \oplus C_1^s) = p_2(x)^{-1}(CC_2^s \oplus C_2^s) \tag{1}$$

and

$$M_2^t = p_1(x)^{-1}(CC_1^t \oplus C_1^t) = p_2(x)^{-1}(CC_2^t \oplus C_2^t). \tag{2}$$

Thus, from Equations (1), (2) and from the facts that $CC_1^s = CC_1^t$ and $CC_2^s = CC_2^t$, we get

$$M_2^s \oplus M_2^t = p_1(x)^{-1}(C_1^s \oplus C_1^t) = p_2(x)^{-1}(C_2^s \oplus C_2^t) \tag{3}$$

Simplicity: The construction of MEM is simple. It uses only shifts and xors other than the block cipher calls. There is no need to keep the information regarding the length of the message in the ciphertext.

Efficiency: MEM requires $m + 4 + 2(\lceil m/(n+1) \rceil - 1)$ block-cipher calls for encrypting an m block message. The more general constructions of tweakable SPRPs like CMC, EME require at little more than $2m$ block-cipher calls for encrypting m message blocks. EME has a length limitation; the mode of operation EME* removes the length limitation and requires $2m + m/n + 1$ block-cipher calls. Comparing MEM to EME*, we see that MEM is almost twice as fast. On the other hand, EME* provides security in a more general model. They provide tweakable SPRP with unrestricted tweaks. MEM can be viewed as a tweakable SPRP with a restriction on the tweak, i.e., the tweak cannot be repeated.

Other than the block cipher calls the operations done by MEM are xors and shifts. Note that the finite field multiplications can also be realized by xors and shifts. MEM uses finite field multiplications of the type $(x^i \oplus x^{i+1})M \bmod \tau(x)$, where $\tau(x)$ is a irreducible polynomial and $0 \leq i \leq n-1$. Thus, to realize this multiplication we need to perform a multiplication of the form $x^i M \bmod \tau(x), 1 \leq i \leq n$. Now,

$$\begin{aligned} x^i M \bmod \tau(x) &= x(x^{i-1}M) \bmod \tau(x) \\ &= xS(x) \bmod (\tau(x)), \text{where } \; S(x) = x^{i-1}M \end{aligned}$$

The computation of the last step requires a shift and at most one xor.

MEM is almost fully parallelizable as the main m block-cipher calls can be done in parallel. The two iterated encryptions of the nonce and the mask calculations have to be done separately. Suppose κ processors are available, where each processor is capable of computing E_K. Then an m-block message can be completed in $2 + 2(\lceil m/(n+1) \rceil) + \lceil m/\kappa \rceil$ parallel encryption rounds.

Even though MEM has one layer of block cipher invocations which is almost fully parallelizable, MEM requires a total of 3 passes. Computing the mask M_1 requires a pass on the data as the mask M_1 contains the xor of all plaintext blocks. A second pass over the data is required in the encryption layer. Finally, computation of M_2 requires the XOR of all the CC_i's which constitutes the third pass. The first and the third passes are computationally inexpensive. However, for long messages, this can be a problem. The tweakable modes of operations CMC, EME and EME* also require three passes over the data.

An encryption scheme is called online if it can output a stream of ciphertext bits as a stream of plaintext bits arrive. Any SPRP construction, in general, is not online. This is because, an SPRP tries to incorporate the effect of the whole plaintext on each block of the ciphertext, hence the construction does not allow it to output a single block of ciphertext unless it has seen the total message. Due to this reason, MEM being a (nonce respecting) SPRP construction is also not online.

Some variations of MEM which do not work: To justify that our construction is non-trivial, here we provide some insecure variations of MEM.

In the current construction, calculation of M_1 and M_2 require the P_i's and CC_i's respectively. Suppose the masks are calculated as $M_1 = E_K(N)$ and $M_2 = E_K(E_K(N))$, where N is the nonce. This would bring down the number of passes to one. However, this leads to an insecure construction which can be easily distinguished from a random permutation. Suppose the adversary provides an encryption query $(N; P_1, P_2, P_3, P_4)$ and gets (C_1, C_2, C_3, C_4) as response. Next he provides a decryption query $(N; C_1, C_2', C_3', C_4')$. In response to his decryption query he obtains (P_1', P_2', P_3', P_4'). If the adversary finds $P_1' = P_1$, then he is sure that he is interacting with the mode of operation not a random permutation.

Table 2: Comparison of MEM with other SPRPs. CMC and MEM can handle messages whose length is a multiple of n.

Mode	Type of IV	Number of block-cipher calls	No of Keys	Length restriction	Parallelizable
CMC	Tweak	$2m+1$	1	Partial	No
EME	Tweak	$2m+2$	1	Yes	Yes
EME*	Tweak	$2m+\frac{m}{n}+1$	3	No	Yes
MEM	Nonce	$m+2\lceil\frac{m}{n+1}\rceil+2$	1	Partial	Yes

In the current construction a two level masking is done both before and after encryption. One level of masking will reduce one pass over the data but yields an insecure construction. Let us consider a construction where only M_1 is used before encryption and the second level of masking is not done. An adversary provides a decryption query $(N; C_1, C_2, C_3, C_4)$ and obtains $(N; P_1, P_2, P_3, P_4)$ as response. Next he provides an encryption query $(N; P_1, P_2', P_3', P_4')$, such that

$$P_1 \oplus P_2' \oplus P_3' \oplus P_4' = P_1 \oplus P_2 \oplus P_3 \oplus P_4.$$

As a response obtains (C_1', C_2', C_3', C_4'). If he finds $C_1' = C_1$, then he is sure he is interacting with the mode of operation.

Similarly, one can try to improve the efficiency by reducing operations. We have tried many other possibilities, all of which turned out to be insecure. This indicates that obtaining a correct construction is a non-trivial task, though we do not claim that our construction is the best possible.

Provable Security: MEM is provably secure. We state a theorem related to the security of MEM in Section 4 and provide the details of the proof in Section A. For MEM we prove that a computationally bounded symmetric nonce respecting adversary cannot distinguish the output of MEM and MEM^{-1} from the output of a random permutation and its inverse. Our proof follows the techniques used in [3, 4].

Comparison: Table 2 provides a comparison of the various features of MEM and other modes of operations which are SPRPs. Note that MEM is a nonce respecting SPRP while the other constructions are more general tweakable SPRPs. Thus, a direct comparison is perhaps not meaningful. On the other hand, the comparison shows that if we relax the security requirement, then a big improvement in efficiency is possible.

4 Security of MEM

4.1 Definitions and Notation

As mentioned earlier, an n-bit block cipher is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K} \neq \emptyset$ is called the key space and for any $K \in \mathcal{K}$, $E(K,.)$ is a permutation. We will usually write $E_K(\,)$ instead of $E(K,.)$.

An adversary A is a probabilistic algorithm which has access to some oracles and which outputs either 0 or 1. Oracles are written as superscripts. The notation $A^{\mathcal{O}_1,\mathcal{O}_2} \Rightarrow 1$ denotes the event that the adversary A, interacting with the oracles $\mathcal{O}_1, \mathcal{O}_2$, finally outputs the bit 1.

Let $\mathrm{Perm}(n)$ denote the set of all permutations on $\{0,1\}^n$. We require $E(,)$ to be a strong pseudorandom permutation. The advantage of an adversary in breaking the strong pseudorandomness of $E(,)$ is defined in the following manner.

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

Formally, a nonce based enciphering scheme is a function $\mathbf{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{M}$, where $\mathcal{K} \neq \emptyset$ and $\mathcal{N} \neq \emptyset$ are the key space and the nonce space respectively and $\mathcal{M} = \cup_{i \geq 1}\{0,1\}^{ni}$, where n is the length of a message block. We shall often write $\mathbf{E}_K^N(.)$ instead of $\mathbf{E}(K, N, .)$. The inverse of an enciphering scheme is $\mathbf{D} = \mathbf{E}^{-1}$ where $X = \mathbf{D}_K^N(Y)$ if and only if $\mathbf{E}_K^N(X) = Y$.

Let $\mathrm{Perm}^{\mathcal{N}}(\mathcal{M})$ denote the set of all functions $\boldsymbol{\pi} : \mathcal{N} \times \mathcal{M} \to \mathcal{M}$ where $\boldsymbol{\pi}(\mathcal{N}, .)$ is a length preserving permutation. Such a $\boldsymbol{\pi} \in \mathrm{Perm}^{\mathcal{N}}(\mathcal{M})$ is called a nonce indexed permutation. For a nonce based enciphering scheme $\mathbf{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{M}$, we define the advantage an adversary A has in distinguishing $\mathbf{E}$ and its inverse from a random nonce indexed permutation and its inverse in the following manner.

$$\mathbf{Adv}_{\mathbf{E}}^{\pm\mathrm{nprp}}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{N}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right].$$

Symmetric Nonce Respecting Adversary. We assume that the adversary is "symmetric nonce respecting" which means that the adversary does not query the encryption or decryption oracle with the same nonce more than once. Note that an adversary is allowed to present a particular nonce to both the encryption and decryption oracles. A usual nonce respecting adversary, never repeats a nonce to only the encryption oracle.

Further, an adversary never queries its deciphering oracle with (N, C) if it got C in response to an encipher query (N, M) for some M. Similarly, the adversary never queries its enciphering oracle with (N, M) if it got M as a response to a decipher query of (N, C) for some C. These queries are called *pointless* as the adversary knows what it would get as response for such queries.

Following [4], we define the query complexity σ_n of an adversary as follows. A string X contributes $\max(|X|/n, 1)$ to the query complexity. A tuple of strings $(X_1, X_2, \ldots)$ contributes the sum of the contributions from all oracle queries plus the contribution from the adversary's output. Suppose an adversary makes q queries where the number of n-bit blocks in the ith query is ℓ_i. Then, $\sigma_n = 1 + \sum_{i=1}^{q}(1 + \ell_i) \geq 2q$. Let ρ be a list of resources used by the adversary A and suppose $\mathbf{Adv}_{\pi}^{\pm\mathrm{xxx}}(A)$ has been defined where π is either a block cipher or a nonce based encipering scheme. $\mathbf{Adv}_{\pi}^{\pm\mathrm{xxx}}(\rho)$ denotes the maximal value of $\mathbf{Adv}_{\pi}^{\pm\mathrm{xxx}}(A)$ over all adversaries A using resources at most ρ. Usual resources of interest are the running time t of the adversary, the number of oracle queries q made by the adversary and the query complexity σ_n $(n \geq 1)$.

The notation MEM$[E]$ denotes a nonce based enciphering scheme, where the n-bit block cipher E is used in the manner specified by MEM. Our purpose is to show that MEM$[E]$ is secure if E is secure. The notation MEM[Perm(n)] denotes a nonce based enciphering scheme obtained by plugging in a random permutation from Perm(n) into the structure of MEM. For an adversary attacking MEM[Perm(n)], we do not put any bound on the running time of the adversary, though we still put a bound on the query complexity σ_n. We show the information theoretic security of MEM[Perm(n)] by obtaining an upper bound on $\mathbf{Adv}_{\mathrm{MEM[Perm}(n)]}^{\pm\mathrm{nprp}}(\sigma_n)$. The upper bound is

obtained in terms of n and σ_n. For a fixed block cipher E, we bound $\mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[E]}(\sigma_n, t)$ in terms of $\mathbf{Adv}^{\pm\text{prp}}_{E}(\sigma_n, t')$, where $t' = t + O(\sigma_n)$. In this process, we need to consider an adversary's advantage in distinguishing a nonce based enciphering scheme $\mathbf{E}$ from an oracle which simply returns random bit strings. This advantage is defined in the following manner.

$$\mathbf{Adv}^{\pm\text{nrnd}}_{\mathbf{E}}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.)\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1] - \Pr[A^{\$(.,.)\$(.,.)} \Rightarrow 1]$$

where $\$(.,.)$ returns random bits of length $|M|$.

We will use the notation $\mathbf{E}_\pi$ as a shorthand for MEM[Perm(n)] and $\mathbf{D}_\pi$ will denote the inverse of $\mathbf{E}_\pi$. Thus, the notation $A^{\mathbf{E}_\pi, \mathbf{D}_\pi}$ will denote an adversary interacting with the oracles $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$.

4.2 Statement of Result

The following theorem specifies the security of MEM.

Theorem 1 *Fix n, σ_n to be positive integers and an n-bit block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then*

$$\mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[\text{Perm}(n)]}(\sigma_n) \leq \frac{9.5\sigma_n^2}{2^n} \tag{4}$$

$$\mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[E]}(\sigma_n, t) \leq \frac{9.5\sigma_n^2}{2^n} + \mathbf{Adv}^{\pm\text{prp}}_{E}(\sigma_n, t') \tag{5}$$

where $t' = t + O(\sigma_n)$.

The above result and its proof is similar to previous work (see for example [3, 4, 11]). As mentioned in [3], Equation (5) embodies a standard way to pass from the information theoretic setting to the complexity theoretic setting. Let $\mathbf{E}(.,.,.)$ denote MEM[E]. For any adversary A, we have the following.

$$\begin{aligned}
\mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[E]}(A) &= \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \text{Perm}^{\mathcal{N}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right] \\
&= \left(\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1\right]\right) \\
&\quad + \left(\Pr\left[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1\right]\right. \\
&\quad \left. - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \text{Perm}^{\mathcal{N}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\right) \\
&= X + \mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[\text{Perm}(n)]}(\sigma_n)
\end{aligned}$$

where $X = \left(\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathbf{E}_K(.,.),\mathbf{E}_K^{-1}(.,.)} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1\right]\right)$. The quantity X represents an adversary's advantage in distinguishing MEM[E] from MEM[π], where π is a randomly chosen permutation from Perm(n). Clearly, such an adversary A can also distinguish E from a random permutation and hence $X \leq \mathbf{Adv}^{\pm\text{prp}}_{E}(A)$. This argument shows how (4) is obtained from (5). The basic idea of proving (4) is as follows.

$$\mathbf{Adv}^{\pm\text{nprp}}_{\text{MEM}[\text{Perm}(n)]}(\sigma_n) = \left(\Pr\left[\pi \xleftarrow{\$} \text{Perm}(n) : A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1\right]\right.$$

$$
\begin{aligned}
& - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{N}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\Big) \\
= & \left(\Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1\right] - \Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right]\right) \\
& + \left(\Pr\left[A^{\$(.,.),\$(.,.)} \Rightarrow 1\right] - \Pr\left[\boldsymbol{\pi} \xleftarrow{\$} \mathrm{Perm}^{\mathcal{N}}(\mathcal{M}) : A^{\boldsymbol{\pi}(.,.),\boldsymbol{\pi}^{-1}(.,.)} \Rightarrow 1\right]\right) \\
\leq & \ \mathbf{Adv}^{\pm\mathrm{nrnd}}_{\mathrm{MEM}[\mathrm{Perm}(n)]}(\sigma_n) + \binom{q}{2}\frac{1}{2^n}
\end{aligned}
$$

where q is the number of queries made by the adversary. For a proof of the last inequality see [4].

The main task of the proof now reduces to obtaining an upper bound on $\mathbf{Adv}^{\pm\mathrm{nrnd}}_{\mathrm{MEM}[\mathrm{Perm}(n)]}(\sigma_n)$. This proof is provided in Section A. The proof uses the standard technique of sequence of games between an adversary and the mode of operation MEM. The proof is similar to the corresponding proofs of CMC [3] and EME [4]. By a sequence of games we show that if in response to any valid query of the adversary random strings of appropriate lengths are returned then the internal computations of MEM can be performed consistently under the assumption that the block cipher and its inverse are random permutations. The crux of the proof lies in showing that there would seldom be any collisions in the range and domain sets of the block cipher if the adversary queries MEM with valid queries and MEM responds to them by producing random strings. In a later part we remove the randomness associated with the adversary and the game runs on a fixed transcript consisting of the queries and their responses. We show that in such a situation also the internal computations of MEM can be performed consistently.

We later prove (see (15)) that for any adversary with query complexity σ_n

$$
\mathbf{Adv}^{\pm\mathrm{nrnd}}_{\mathrm{MEM}[\mathrm{Perm}(n)]}(\sigma_n) \leq \frac{9\sigma_n^2}{2^n} + \frac{q}{2 \times 2^n}
$$

where q is the number of queries made by the adversary. Using this, we obtain

$$
\begin{aligned}
\mathbf{Adv}^{\pm\mathrm{nprp}}_{\mathrm{MEM}[\mathrm{Perm}(n)]}(\sigma_n) &\leq \frac{9\sigma_n^2}{2^n} + \frac{q}{2 \times 2^n} + \binom{q}{2}\frac{1}{2^n} \\
&\leq \frac{9\sigma_n^2 + 0.5q^2}{2^n} \\
&\leq \frac{9.5\sigma_n^2}{2^n}.
\end{aligned}
$$

Why nonce helps the proof? CMC, EME are secure with unrestricted tweaks. But they use two levels of encryption. Intuitively, the security demands that the blocks that get into the last level of block-cipher must themselves be random. Making this possible with a single level of encryption appear to be otherwise impossible without use of a non repeating tweak (the nonce). In more concrete terms, the first comment on Table 3 in Section A requires nonces to be non-repeating to rule out certain types of collisions.

5 Conclusion

In this paper, we have presented a new mode of operation of a block cipher called MEM. MEM is a nonce based enciphering scheme which can be proved to be secure against symmetric nonce

respecting adversaries. Such an adversary can query both the encryption and decryption oracles but is not allowed to repeat a nonce to either of these. MEM is almost twice as fast as the more general tweakable modes of operation CMC [3] and EME [4]. If the block size of the underlying block cipher is n, then MEM can handle messages whose lengths are multiples of n. In a future work, we plan to construct MEM*, which will be capable of handling arbitrary length messages.

References

[1] M. Bellare and P. Rogaway, " Encode-then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography", Advances in Cryptology - Asiacrypt '00, LNCS, vol. 1976, pp. 317-330, Springer, 2000.

[2] S. Halevi, "EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data". INDOCRYPT 2004, pp. 315-327, Springer 2004

[3] S. Halevi and P. Rogaway, "A tweakable enciphering mode", Advances in Cryptology - CRYPTO '03, LNCS, vol. 2729, pp. 482-499, Springer, 2003.

[4] S. Halevi and P. Rogaway, "A parallelizable enciphering mode", Topics in Cryptology, CT-RSA 2004, LNCS, vol. 2964, pp. 292-304, Springer, 2004

[5] C. S. Jutla: Encryption modes with almost free message integrity. EUROCRYPT 2001: 529-544.

[6] M. Liskov, R. L. Rivest and D. Wagner: Tweakable Block Ciphers. CRYPTO 2002: 31-46.

[7] M. Luby and C. Rackoff, "How to construct pseudo-random permutations and pseudo-random functions", *SIAM Journal of Computing*, vol. 17, pp. 373-386, 1988.

[8] David A. McGrew and John Viega: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. Proceedings of Indocrypt 2004, 343-355.

[9] Moni Naor and Omer Reingold, "On the construction of pseudo-random permutations: Luby-Rackoff revisited," J. of Cryptology, vol 12, pp. 29-66, 1999.

[10] P. Rogaway, " Nonce-Based Symmetric Encryption", Fast Software Encryption (FSE) 2004, LNCS 3017, pp. 348-359, Springer, 2004.

[11] P. Rogaway, M. Bellare and J. Black: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Conference on Computer and Communication Security 2001: 196-205.

A Upper Bound on $\mathbf{Adv}^{\pm\text{nrnd}}_{\text{MEM}[\text{Perm}(n)]}(\sigma_n)$

We model the adversary's interaction with the oracles $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$ as a game. In the usual game, which we call MEM1, the adversary submits queries to $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$ and gets appropriate answers. Starting from this game, we modify it in successive steps to obtain a game where the adversary is provided random bit strings of appropriate lengths. This results in a sequence of games: MEM1, RAND1, RAND2, RAND3 and NON.

By an abuse of notation, we will use A^{MEM1} to denote an adversary A's interaction with the oracles while playing game MEM1. We will use similar notations for the other games.

A.1 The Game Sequence

Game MEM1: We describe the attack scenario of the adversary A through a probabilistic game which we call MEM1. In MEM1, the adversary interacts with $\mathbf{E}_\pi$ and $\mathbf{D}_\pi$ where π is a randomly chosen permutation from Perm(n). Instead of initially choosing π, we build up π in the following manner.

Initially π is assumed to be undefined everywhere. When $\pi(X)$ is needed, but the value of π is not yet defined at X, then a random value is chosen among the available range values. Similarly when $\pi^{-1}(Y)$ is required and there is no X yet defined for which $\pi(X) = Y$, we choose a random value for $\pi^{-1}(Y)$ from the available domain values.

The domain and range of π are maintained in two sets *Domain* and *Range*, and $\overline{Domain}$ and $\overline{Range}$ are the complements of *Domain* and *Range* relative to $\{0,1\}^n$. The game MEM1 is shown in Figure 4. The figure shows the subroutines Choose-π, Choose-π^{-1}, the initialization steps and how the game responds to a encipher/decipher query of the adversary. The i^{th} query of the adversary depends on its previous queries, the responses to those queries and on some coins of the adversary.

The game MEM1 accurately represents the attack scenario, and by our choice of notation, we can write

$$\Pr[A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1] = \Pr[A^{\text{MEM1}} \Rightarrow 1]. \tag{6}$$

Game RAND1: We modify MEM1 by deleting the boxed entries in MEM1 and call the modified game as RAND1. By deleting the boxed entries it cannot be guaranteed that π would be a permutation as though we do the consistency checks but do not reset the values of Y (in Choose-π) and X (in Choose-π^{-1}). Thus, the games MEM1 and RAND1 are identical apart from what happens when the bad flag is set. So,

$$\Pr[A^{\text{MEM1}} \Rightarrow 1] - \Pr[A^{\text{RAND1}} \Rightarrow 1] \leq \Pr[A^{\text{RAND1}}\text{sets } \mathit{bad}] \tag{7}$$

Game RAND2: We make certain changes to the game RAND1 which are invisible to the adversary. In RAND1 as the permutation π is not maintained, thus the subroutine Choose-π and Choose-π^{-1} are no more needed. Instead we add a new subroutine called Check-Domain-Range(X, Y). In Check-Domain-Range(X, Y), X is inserted into *Domain* and Y is inserted into *Range*, also, if $X \in Domain$ or $Y \in Range$ then the bad flag is set.

In this game, for an encryption query having two or more blocks, instead of choosing CC_i's and $M_{2,k}$ randomly and then setting values of C_i's we randomly choose C_i's and $M_{2,k}$ and next we set values of CC_i's. Similarly for a decryption query we choose P_i's and $M_{1,k}$ and set values of PP_i's.

For an encryption query of length 1, C_1 is chosen randomly and the value of $M_{1,0}$ is set using the value of C_1 and $xEEN$. The game RAND2 is shown in Figure 5.

RAND1 and RAND2 are similar from the point of view of the adversary. Thus,

$$\Pr[A^{\text{RAND1}} \Rightarrow 1] = \Pr[A^{\text{RAND2}} \Rightarrow 1] \tag{8}$$

also,

$$\Pr[A^{\text{RAND1}}\text{sets } \mathit{bad}] = \Pr[A^{\text{RAND2}}\text{sets } \mathit{bad}] \tag{9}$$

In RAND2 the adversary is supplied with random bits as response to queries to both the encrypt and the decrypt oracles. Hence,

$$\Pr[A^{\text{RAND2}} \Rightarrow 1] = \Pr[A^{\$(.,.),\$(.,.)} \Rightarrow 1] \tag{10}$$

Now, from Equation (6), (7), (8), (9) and (10) we get

$$\begin{aligned}
\mathbf{Adv}^{\pm\text{nrnd}}_{\text{MEM}[\text{Perm}(n)]}(A) &= \Pr[A^{\mathbf{E}_\pi,\mathbf{D}_\pi} \Rightarrow 1] - \Pr[A^{\$(.,.),\$(.,.)} \Rightarrow 1] \\
&= \Pr[A^{\text{MEM1}} \Rightarrow 1] - \Pr[A^{\text{RAND2}} \Rightarrow 1] \\
&= \Pr[A^{\text{MEM1}} \Rightarrow 1] - \Pr[A^{\text{RAND1}} \Rightarrow 1] \\
&\leq \Pr[A^{\text{RAND1}}\text{sets } \mathit{bad}] \\
&= \Pr[A^{\text{RAND2}}\text{sets } \mathit{bad}] \qquad (11)
\end{aligned}$$

Our task is thus to bound $\Pr[A^{\text{RAND2}}\text{sets } \mathit{bad}]$.

Game RAND3**:** Here we do two subtle changes on the game RAND2. Here instead of the *Domain* and *Range* sets we use multisets $\mathcal{D}$ and $\mathcal{R}$ respectively. In the game RAND3 on either an encryption or a decryption query by the adversary a random string is given as output. Next, the internal variables are adjusted in the first phase of the finalization step. The *bad* flag is set at the second phase of the finalization step by checking whether a value occurs in either $\mathcal{R}$ or $\mathcal{D}$ more than once. The game RAND3 is shown in Figure 6. RAND3 sets *bad* in exactly the same conditions in which RAND2 sets *bad*, hence

$$\Pr[A^{\text{RAND2}}\text{sets } \mathit{bad}] = \Pr[A^{\text{RAND3}}\text{sets } \mathit{bad}]. \qquad (12)$$

Game NON**:** In this game we get rid of the adversary and its interactions. We want to bound $\Pr[A^{\text{RAND3}}\text{sets } \mathit{bad}]$, we assume that the adversary is deterministic and so the probability is over the random choices of $P \xleftarrow{\$} \{0,1\}^{nm_s}$ and $C \xleftarrow{\$} \{0,1\}^{nm_s}$ and the other random choices made during the finalization steps but not on the distribution of the queries made by the adversary.

In the previous games the adversary specified the plaintext along with the nonce, when it was an encipher query and the ciphertext along with a nonce if it was a decipher query. Let us consider that the adversary specifies the ciphertext also along with the plain text and the nonce in case of an encipher query. Similarly, for a decipher query, the adversary specifies the plaintext also along with the ciphertext and the nonce. We make a stronger claim that for any set of values returned to the adversary and for any set of queries (none pointless and all nonce respecting) associated to them the game RAND3 rarely sets bad. But this claim is not quite correct. For example, let the r^{th} query be an encipher query of $N^r; P_1^r, P_2^r, P_3^r, \ldots, P_{m_r}^r$ for which the adversary got an output of $C_1^r, C_2^r, \ldots, C_{m_r}^r$. Let the t^{th} $(t > r)$ query be a decipher query $N^t; C_1^t, C_1^t, \ldots, C_{m_t}^t$ for which the adversary got the response $P_1^t, P_2^t, \ldots, P_{m_t}^t$. Let $N^t = N^r$ and

$$P_1^t \oplus P_2^t \oplus \cdots \oplus P_{m_t}^t = P_1^r \oplus P_2^r \oplus \cdots \oplus P_{m_r}^r.$$

Note that neither of these two queries are pointless and both are nonce respecting. But in such a situation MPP^r and MPP^t will take the same values as

$$MPP^r = P_1^r \oplus P_2^r \oplus \cdots \oplus P_{m_r}^r \oplus EN^r = P_1^t \oplus P_2^t \oplus P_3^t \oplus \cdots \oplus P_{m_t}^t \oplus EN^t = MPP^t.$$

As both MPP^r and MPP^t are added to $\mathcal{D}$, hence in the finalization step the *bad* flag will be surely set. A similar collision will occur in the values taken by MCC^r and MCC^t if the xor of the ciphertext blocks returned in an encipher query is same as the xor of the ciphertext blocks of a previous decryption query with the same nonce. A similar situation occurs in [4], where it is called an immediate collision. Following [4] we call such collisions *immediate collisions*. The probability of

an immediate collision in the game RAND3 is at most $\frac{q}{2}\frac{1}{2^n}$. This is because an immediate collision can occur in at most $q/2$ pairs of queries. As the adversary is symmetric nonce respecting there can be at most two queries (one encrypt and the other decrypt) with the same nonce.

Now we modify game RAND3 into a new game NON (noninteractive). NON depends on a fixed transcript $\mathsf{tr} = (\mathbf{ty}, \mathbf{N}, \mathbf{P}, \mathbf{C})$ with $\mathbf{ty} = (ty^1, ty^2, \ldots, ty^q)$, $\mathbf{N} = (\mathsf{N}^1, \mathsf{N}^2, \ldots \mathsf{N}^q)$, and $\mathbf{P} = (\mathsf{P}^1, \mathsf{P}^2, \ldots \mathsf{P}^q)$, and $\mathbf{C} = (\mathsf{C}^1, \mathsf{C}^2, \ldots \mathsf{C}^q)$ where $ty^s = \{\mathsf{Enc}, \mathsf{Dec}\}$, $\mathsf{N}^s = \{0,1\}^n$ and $\mathsf{P}^s = \mathsf{P}^s_1, \mathsf{P}^s_2, \ldots, \mathsf{P}^s_{ms}$, and $\mathsf{C}^s = \mathsf{C}^s_1, \mathsf{C}^s_2, \ldots, \mathsf{C}^s_{ms}$. If this fixed transcript does not contain any pointless query, and each query is nonce respecting and queries do not specify an immediate collision, then the transcript is called *allowed*.

Now fix a transcript tr which maximizes the probability of *bad* being set. This transcript tr is hardwired into the game NON. The game NON is shown in Figure 7. We would like to point out that from a syntactic point of view, *the game* NON *is the same as the game* RAND3 after the **Finalization** step. It is only the interpretation of the variables which are different in the two games.

If IC denotes the event that an immediate collision occurs in the game RAND3 then we have

$$\Pr[A^{\text{RAND3}}\text{sets } bad] = \Pr[A^{\text{RAND3}}\text{sets } bad|IC]\Pr[IC] + \Pr[A^{\text{NON}}\text{sets } bad].$$

An immediate collision will always set the *bad* flag. Hence, $\Pr[A^{\text{RAND3}}\text{sets } bad|IC] = 1$ and so

$$\Pr[A^{\text{RAND3}}\text{sets } bad] \leq \Pr[A^{\text{NON}}\text{sets } bad] + \frac{q}{2}\frac{1}{2^n}. \tag{13}$$

A.2 Analysis of Game NON

In the analysis we consider the sets $\mathcal{D}$ and $\mathcal{R}$ to consist of the formal variables instead of their values. For example, whenever we set $\mathcal{D} \leftarrow \mathcal{D} \cup \{X\}$ for some variable X we think of it as setting $\mathcal{D} \leftarrow \mathcal{D} \cup \{\text{“}X\text{”}\}$ where “X” is the name of that formal variable. Thus, our goal would be to bound the probability that two formal variables in sets $\mathcal{D}$ and $\mathcal{R}$ takes the same value. The formal variables in both $\mathcal{D}$ and $\mathcal{R}$ do not depend on the random choices made in the game NON. Let $\mathsf{NewN} = \{1 \leq s \leq q : N^s \text{ is new}\}$. The sets $\mathcal{D}$ and $\mathcal{R}$ can be written as

$$\begin{aligned}
\mathcal{D} &= \{MPP^s, MCC^s : 1 \leq s \leq q\} \cup \{PP^s_i : 1 \leq i \leq m_s, m_s > 1, 1 \leq s \leq q\} \\
&\quad \cup \{N^s, xEN^s : s \in \mathsf{NewN}\} \cup \{M^s_{1,i}, M^s_{2,i} : 0 \leq i \leq \lceil m_s/(n+1) \rceil - 1, m_s > 2, 1 \leq s \leq q\} \\
&\quad \cup \{M^s_{2,0} : m_s = 1, 1 \leq s \leq q\}. \\
\mathcal{R} &= \{MCC^s : m_s = 1, 1 \leq s \leq q\} \cup \{CC^s_i : 1 \leq i \leq m_s, m_s > 1, 1 \leq s \leq q\} \\
&\quad \cup \{EN^s, EEN^s : s \in \mathsf{NewN}\} \cup \{M^s_{1,i}, M^s_{2,i} : 0 \leq i \leq \lceil m_s/(n+1) \rceil, m_s > 2, 1 \leq s \leq q\} \\
&\quad \cup \{M^s_{1,0} : m_s = 1, 1 \leq s \leq q\}.
\end{aligned}$$

We make two claims which will lead us to Theorem 1.

Claim 1: For any two distinct variables $X, X' \in \mathcal{D}$, $\Pr[X = X'] = \Pr[X \oplus X' = 0] \leq \frac{1}{2^n}$.

Claim 2: For any two distinct variables $X, X' \in \mathcal{R}$, $\Pr[X = X'] = \Pr[X \oplus X' = 0] \leq \frac{1}{2^n}$.

The proofs of Claims 1 and 2 are similar. Below we present a sketch of the proof of Claim 1.

Proof of Claim 1: The proof of Claim 1 is by exhaustive case analysis. A list of the variables which are included in $\mathcal{D}$ is given in Table 3. For different conditions of the game, i.e., for different

Table 3: The variables in $\mathcal{D}$.

MPP^s	=	$\mathsf{P}_1^s \oplus \mathsf{P}_2^s \oplus \ldots \mathsf{P}_{m_s}^s \oplus EN^s$
MCC^s	=	$\mathsf{C}_1^s \oplus xEEN^s$ when $t^s = \mathsf{Dec}$ and $m_s = 1$ $\mathsf{C}_1^s \oplus \mathsf{C}_2^s \oplus EN^s \oplus EEN^s$ when $m_s = 2$ $\mathsf{C}_1^s \oplus \mathsf{C}_2^s \oplus \ldots \oplus \mathsf{C}_{m_s} \oplus EEN^s$, when $m_s > 2$
PP_i^s	=	$p_i(x)M_{1,k}^s \oplus \mathsf{P}_i^s$ when $m_s > 2$ $M_{1,0}^s \oplus \mathsf{P}_i^s$ when $m_s = 2$ and $i = 1$ $M_{1,0}^s \oplus EEN^s \oplus \mathsf{P}_i^s$ when $m_s = 2$ and $i = 2$
$M_{1,i}^s$	=	$\mathsf{C}_1^s \oplus xEEN^s$ when $t^s = \mathsf{Enc}$ and $m_s = 1$ Randomly selected, otherwise
$M_{2,i}^s$	=	$\mathsf{P}_1^s \oplus EEN^s$ when $t^s = \mathsf{Dec}$ and $m_s = 1$ Randomly selected, otherwise
N^s		
xEN^s		

query lengths and depending on whether it is an encryption or decryption query the representation of some variables changes as shown in Table 3. Also, note that not all variables are defined for all queries.

The case analysis involves showing that the probability of collision between any two distinct variables in $\mathcal{D}$ (as shown in Table 3) is at most $1/2^n$. To show this, it is sufficient to show that if X_1, X_2 are two formal variables in $\mathcal{D}$ then the expression $X_1 \oplus X_2$ is either always non-zero; or can be written as the XOR of two strings, such that at least one of them is a randomly chosen n-bit string. So, $\Pr[X_1 \oplus X_2] \leq 1/2^n$. The total number of possible choices of X_1 and X_2 in $\mathcal{D}$ is fairly large, and verifying the above statement for most of these cases is quite routine. Below, we comments on certain types of collisions.

Comments on Table 3:

1. As NON runs on an allowed transcript so there cannot be more than one query of the same kind (encryption or decryption) with the same nonce. But there can be an encryption query and a decryption query with the same nonce. In game NON, for each nonce it is checked whether a previous query was made with the same nonce, if there are no such previous queries then a random value is assigned to EN otherwise if the current nonce N^r is equal to some N^s, $s < r$, the the value of EN^s is assigned to EN^r and N^r is not put into $\mathcal{D}$. Thus the nonces present in $\mathcal{D}$ are all distinct, thus $\Pr[N^s \oplus N^t = 0] = 0$.

2. The quantities EN^s, EN^t and EEN^s, EEN^t are randomly chosen quantities. Due to the reason explained in point 1 above, we will have $EN^s \neq EN^t$ and $EEN^s \neq EEN^t$ for $s \neq t$. This prevents the following collisions: MCC^s and MPP^t, xEN^s and xEN^t, N^s and xEN^t.

3. As discussed earlier the game NON runs on *allowed transcripts*. This prevents the following collisions: MPP^s and MPP^t, MCC^s and MPP^t. Also, as NON runs on allowed transcripts a collision between MCC^s ($t^s = \mathsf{Dec}$ and $m_s = 1$) and $M_{1,i}$ ($t^s = \mathsf{Enc}$), $m_s = 1$) is not possible.

4. Consider the collision between PP_i^s and PP_j^s where $m_s > 2$.

$$PP_i^s \oplus PP_j^s = p_i(x)M_{1,k_1}^s \oplus \mathsf{P}_i^s \oplus p_j(x)M_{1,k_2}^s \oplus \mathsf{P}_j^s.$$

Let $i = (n+1) \times n_1 + r_1$ and $j = (n+1) \times n_2 + r_2$, where $r_1 = i \bmod (n+1)$ and $r_2 = j \bmod (n+1)$.

- If $n_1 = n_2$ then as per the specification of the mode $M_{1,k_1} = M_{1,k_2} = M$ (say) and hence $p_i(x)M_{1,k_1}^s \oplus p_j(x)M_{1,k_2}^s = M(p_i(x) \oplus p_j(x))$. Recall from Section 2, that by the construction of $p_i(x)$ and $p_j(x)$, for this case, we have $p_i(x) \oplus p_j(x) \not\equiv 0 \bmod \tau(x)$. Hence, $M(p_i(x) \oplus p_j(x))$ is a random n-bit string.
- If $n_1 \neq n_2$ then $M_{1,k_1} \neq M_{1,k_2}$. As M_{1,k_1}, M_{1,k_2} are selected randomly, we have $p_i(x)M_{1,k_1}^s \oplus p_j(x)M_{1,k_2}^s$ to be a random string.
- Hence, in both cases, we have $\Pr[PP_i^s \oplus PP_j^s = 0] \leq 1/2^n$.

This completes the proof of Claim 1. Now,

$$|\mathcal{D}| = |\mathcal{R}| \leq 2q + \sum_{s=1}^{q} m_s + 2|\mathsf{NewN}| + 2\sum_{s=1}^{q}(\lceil m_s/(n-1)\rceil - 1) + q.$$

As, $\sum_{s=1}^{q}\lceil m_s/(n-1)\rceil \leq \sum_{s=1}^{q}$, $|\mathsf{NewN}| \leq q$, and as per our definition of query complexity $\sigma_n = q + \sum_{s=1}^{q} m_s + 1$. Hence, we have $|\mathcal{D}| = |\mathcal{R}| \leq 3\sigma_n$. Hence, from Claim 1 and Claim 2 we get

$$\begin{aligned}\Pr[\text{NON sets } \textit{bad}] &= \Pr[\text{There is a collision in } \mathcal{D}] + \Pr[\text{There is a collision in } \mathcal{R}] \\ &\leq 2 \times \frac{1}{2^n}\binom{3\sigma_n}{2} \qquad (14)\end{aligned}$$

From Equations (11), (12), (13) and (14), we get

$$\mathbf{Adv}^{\pm\text{nrnd}}_{\text{MEM}[\text{Perm}(n)]}(A) \leq 2 \times \frac{1}{2^n}\binom{2\sigma_n}{2} + \frac{q}{2}\frac{1}{2^n} \leq \frac{9\sigma_n^2}{2^n} + \frac{q}{2 \times 2^n} \qquad (15)$$

Since A was an arbitrary adversary with query complexity σ_n, the upper bound in (15) also holds for $\mathbf{Adv}^{\pm\text{nrnd}}_{\text{MEM}[\text{Perm}(n)]}(\sigma_n)$.

Figure 4: Game MEM1

Subroutine Choose-$\pi(X)$

$Y \xleftarrow{\$} \{0,1\}^n$; **if** $Y \in Range$ **then** $bad \leftarrow$ true; $\boxed{Y \xleftarrow{\$} \overline{Range}}$; **endif**;

if $X \in Domain$ **then** $bad \leftarrow$ true; $\boxed{Y \leftarrow \pi(X)}$; **endif**

$\pi(X) \leftarrow Y$; $Domain \leftarrow Domain \cup \{X\}$; $Range \leftarrow Range \cup \{Y\}$; **return**$(Y)$;

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

$X \xleftarrow{\$} \{0,1\}^n$; **if** $X \in Domain$, $bad \leftarrow$ true; $\boxed{X \xleftarrow{\$} \overline{Domain}}$; **endif**;

if $Y \in Range$ **then** $bad \leftarrow$ true; $\boxed{X \leftarrow \pi^{-1}(Y)}$; **endif**;

$\pi(X) \leftarrow Y$; $Domain \leftarrow Domain \cup \{X\}$; $Range \leftarrow Range \cup \{Y\}$; **return**$(X)$;

Initialization:

for all $X \in \{0,1\}^n$ $\pi(X) \leftarrow$ undef **endfor**

$bad \leftarrow$ false

Respond to the s^{th} adversary query as follows:

Encipher query

$\overline{\mathsf{Enc}(N^s; P_1^s, \ldots, P_{m_s}^s)}$

if $N^s = N^r$ for some $r < s$ **then**
$EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
else
$EN^s \leftarrow$ Choose-$\pi(N^s)$; $EEN^s \leftarrow$ Choose-$\pi(xEN^s)$;
endif
$MP^s \leftarrow P_1^s \oplus P_2^s \oplus \ldots \oplus P_{m_s}^s$;
if $m_s = 1$ **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0}^s \leftarrow$ Choose-$\pi(MPP^s)$
$C_1^s \leftarrow M_{1,0}^s \oplus xEEN^s$; **return** C_1^s;
endif

if $m_s = 2$ **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0} \leftarrow$ Choose-$\pi(MPP^s)$;
$PP_1^s \leftarrow M_{10}^s \oplus P_1^s$; $PP_2^s \leftarrow M_{10}^s \oplus EEN^s \oplus P_2$;
$CC_1^s \leftarrow$ Choose-$\pi(PP_1^s)$;$CC_2^s \leftarrow$ Choose-$\pi(PP_2^s)$;
$MC^s \leftarrow CC_1^s \oplus CC_2^s$; $MCC^s \leftarrow MC^s \oplus EEN^s$;
$M_{2,0}^s \leftarrow$ Choose-$\pi(MCC^s)$;
$C_1^s \leftarrow M_{2,0} \oplus CC_1$; $C_2^s \leftarrow M_{2,0}^s \oplus EN^s \oplus CC_2^s$;
returnC_1^s, C_2^s;
endif

if $m^s > 2$ **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0} \leftarrow$Choose-$\pi(MPP^s)$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s
if $count > n+1$ **then**
$count \leftarrow 0$; $k \leftarrow k+1$; $M_{1,k}^s =$ Choose-$\pi(M_{1,k-1})$;
endif
$PP_i^s \leftarrow p_i(x) M_{1,k}^s \oplus P_i^s$; $CC_i^s \leftarrow$ Choose-$\pi(PP_i^s)$;
$count \leftarrow count + 1$;
endfor
$MC^s \leftarrow CC_1^s \oplus CC_2^s \oplus \ldots \oplus CC_{m_s}^s$;
$MCC^s \leftarrow MC^s \oplus EEN^s$; $M_{2,0}^s =$Choose-$\pi(MCC^s)$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s,
if $count > n+1$ **then**
$count \leftarrow 0$; $k \leftarrow k+1$; $M_{2,k}^s =$ Choose-$\pi(M_{2,k-1})$;
endif
$C_i^s \leftarrow p_i(x) M_{2,k} \oplus CC_i^s$;
$count \leftarrow count + 1$;
endfor;
return $C_1^s, C_2^s, \ldots, C_{m_s}^s$
endif

Decipher query

$\overline{\mathsf{Dec}(N^s; C_1^s, \ldots, C_{m_s}^s)}$

if $N^s = N^r$ for some $r < s$ then
$EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
else
$EN^s \leftarrow$ Choose-$\pi(N^s)$; $EEN^s \leftarrow$ Choose-$\pi(xEN^s)$;
endif
$MC^s \leftarrow C_1^s \oplus C_2^s \oplus \ldots \oplus C_{m_s}^s$;
if $m = 1$, **then**
$MCC^s \leftarrow MC^s \oplus xEEN^s$;
$M_{2,0}^s \leftarrow$ Choose-$\pi^{-1}(MCC^s)$;
$P_1^s \leftarrow M_{2,0}^s \oplus EN^s$; **return** P_1^s;
endif

if $m = 2$, **then**
$MCC^s \leftarrow MC^s \oplus EN^s \oplus EEN^s$;
$M_{2,0}^s \leftarrow$ Choose-$\pi(MCC^s)$;
$CC_1^s \leftarrow M_{2,0}^s \oplus C_1^s$; $CC_2^s \leftarrow M_{2,0}^s \oplus EN^s \oplus C_2^s$;
$PP_1^s \leftarrow$ Choose-$\pi^{-1}(CC_1^s)$; $PP_2^s \leftarrow$ Choose-$\pi^{-1}(CC_2^s)$;
$MP^s \leftarrow PP_1^s \oplus PP_2^s \oplus EEN^s$; $MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0}^s \leftarrow$ Choose-$\pi(MPP^s)$;
$P_1^s \leftarrow PP_1^s \oplus M_{1,0}^s$; $P_2^s \leftarrow PP_2^s \oplus M_{1,0}^s \oplus EN^s$;
return P_1^s, P_2^s;
endif

if $m_s > 2$, **then**
$MCC^s \leftarrow MC^s \oplus EEN^s$;
$M_{2,0}^s \leftarrow$ Choose-$\pi(MCC^s)$; $count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s
if $count > n+1$ **then**
$count \leftarrow 0$; $k \leftarrow k+1$; $M_{2,k}^s =$ Choose-$\pi(M_{1,k-1})$;
endif
$CC_i^s = C_i^s \oplus p_i(x) M_{2,k}^s$;
$PP_i \leftarrow$ Choose-$\pi^{-1}(CC_i^s)$;
$count \leftarrow count + 1$;
endfor;
$MP^s \leftarrow PP_1^s \oplus PP_2^s \oplus \ldots \oplus PP_{m_s}^s$;
$MPP^s \leftarrow MP^s \oplus EN^s$; $M_{1,0}^s \leftarrow$ Choose-$\pi(MPP^s)$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s,
if $count > n+1$ **then**
$count \leftarrow 0$; $k \leftarrow k+1$; $M_{1,k}^s =$ Choose-$\pi(M_{1,k-1})$;
endif
$P_i^s \leftarrow M_1^s p_i(x) \oplus PP_i^s$;
$count \leftarrow count + 1$;
endfor
return $P_1^s, P_2^s, \ldots, P_{m_s}^s$;
endif

Figure 5: Game RAND2

Subroutine Check-Domain-Range(X, Y)
  if $X \in Domain$ **then** $bad \leftarrow$ true; **endif**
  if $Y \in Range$ **then** $bad \leftarrow$ true; **endif**
  $Domain \leftarrow Domain \cup \{X\}$; $Range \leftarrow Range \cup \{Y\}$;

Initialization
  $Domain \leftarrow Range \leftarrow \emptyset$; $bad \leftarrow$ false;

Encipher query Enc($N^s; P_1^s, \ldots, P_{m_s}^s$)
  if $N^s = N^r$ for some $r < s$ **then**
    $EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
  else
    $EN^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($N^s, EN^s$);
    $EEN^s \leftarrow \{0,1\}^n$; Check-Domain-Range($xEN^s, EEN^s$);
  endif
  $MP^s \leftarrow P_1^s \oplus P_2^s \oplus \ldots \oplus P_{m_s}^s$;

  if $m_s = 1$ **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $C_1^s \xleftarrow{\$} \{0,1\}^n$;
    $M_{1,0}^s \leftarrow C_1^s \oplus xEEN$;
    Check-Domain-Range($MPP^s, M_{1,0}^s$);
    return C_1^s;
  endif

  if $m_s = 2$ **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($MPP^s, M_{1,0}^s$);
    $PP_1^s \leftarrow M_{1,0}^s \oplus P_1^s$; $PP_2^s \leftarrow M_{1,0}^s \oplus EEN^s \oplus P_2^s$;
    $C_1^s \xleftarrow{\$} \{0,1\}^n$; $C_2^s \xleftarrow{\$} \{0,1\}^n$;
    $MC^s \leftarrow C_1^s \oplus C_2^s \oplus EN^s$; $MCC^s \leftarrow MC^s \oplus EEN^s$;
    $M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($MCC^s, M_{2,0}^s$);
    $CC_1^s \leftarrow C_1^s \oplus M_{2,0}^s$; $CC_2^s \leftarrow C_2^s \oplus M_{2,0}^s \oplus EN^s$
    Check-Domain-Range(PP_1^s, CC_1^s);
    Check-Domain-Range(PP_2^s, CC_2^s);
    return C_1^s, C_2^s;
  endif

  if $m_s > 2$ **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($MPP^s, M_{1,0}^s$);
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M_{1,k}^s \xleftarrow{\$} \{0,1\}^n$;
        Check-Domain-Range($M_{1,k-1}^s, M_{1,k}^s$);
      endif
      $PP_i^s \leftarrow p_i(x) M_{1,k}^s \oplus P_i^s$; $C_i^s \xleftarrow{\$} \{0,1\}^n$;
      $count \leftarrow count + 1$;
    endfor
    $MC^s \leftarrow C_1^s \oplus C_2^s \oplus \ldots \oplus C_{m_s}^s$;
    $MCC^s \leftarrow MC^s \oplus EEN^s$; $M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$;
    Check-Domain-Range($MCC^s, M_{2,0}^s$);
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s,
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M_{2,k}^s \xleftarrow{\$} \{0,1\}^n$;
        Check-Domain-Range($M_{2,k-1}^s, M_{2,k}^s$);
      endif
      $CC_i^s \leftarrow p_i(x) M_2 \oplus C_i^s$;
      Check-Domain-Range(PP_i^s, CC_i^s)
      $count \leftarrow count + 1$;
    endfor;
    return $C_1^s, C_2^s, \ldots, C_{m_s}^s$
  endif

Decipher query Enc($N^s; C_1^s, \ldots, C_{m_s}^s$)
  if $N^s = N^r$ for some $r < s$ **then**
    $EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
  else
    $EN^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($N^s, EN^s$);
    $EEN^s \leftarrow \{0,1\}^n$; Check-Domain-Range($xEN^s, EEN^s$);
  endif
  $MC^s \leftarrow C_1^s \oplus C_2^s \oplus \ldots \oplus C_{m_s}^s$;
  if $m_s = 1$ **then**
    $MCC^s \leftarrow MC^s \oplus EEN^s$;
    $P_1^s \xleftarrow{\$} \{0,1\}^n$
    $M_{2,0}^s = P_1^s \oplus EN^s$;
    Check-Domain-Range($M_{2,0}^s, MCC^s$);
    return P_1;
  endif

  if $m_s = 2$ **then**
    $MCC^s \leftarrow MC^s \oplus EN^s \oplus EEN^s$;
    $M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$;
    Check-Domain-Range($MCC^s, M_{2,0}^s$);
    $CC_1^s \leftarrow M_{2,0}^s \oplus C_1^s$; $CC_2^s \leftarrow M_{2,0}^s \oplus EN^s \oplus P_2$;
    $P_1^s \xleftarrow{\$} \{0,1\}^n$; $P_2^s \xleftarrow{\$} \{0,1\}^n$;
    $MP^s \leftarrow P_1^s \oplus P_2^s$; $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$; Check-Domain-Range($MPP^s, M_{1,0}^s$);
    $PP_1^s \leftarrow P_1^s \oplus M_{1,0}^s$; $PP_2^s \leftarrow P_2^s \oplus M_{1,0}^s \oplus EEN^s$
    Check-Domain-Range(PP_1^s, CC_1^s);
    Check-Domain-Range(PP_2^s, CC_2^s);
    return P_1^s, P_2^s;
  endif

  if $m_s > 2$ **then**
    $M_{2,0} \xleftarrow{\$} \{0,1\}^n$
    Check-Domain-Range($MCC^s, M_{2,0}^s$);
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M_{2,k}^s \xleftarrow{\$} \{0,1\}^n$;
        Check-Domain-Range($M_{2,k-1}^s, M_{2,k}^s$);
      endif
      $CC_i^s \leftarrow p_i(x) M_{2,k}^s \oplus C_i^s$; $P_i^s \xleftarrow{\$} \{0,1\}^n$;
      $count \leftarrow count + 1$;
    endfor
    $MP^s \leftarrow P_1^s \oplus P_2^s \oplus \ldots \oplus P_{m_s}^s$;
    $MPP^s \leftarrow MP^s \oplus EN^s$; $M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$;
    Check-Domain-Range($MPP^s, M_{1,0}^s$);
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s,
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M_{1,k}^s \xleftarrow{\$} \{0,1\}^n$;
        Check-Domain-Range($M_{1,k-1}^s, M_{1,k}^s$);
      endif
      $PP_i^s \leftarrow p_i(x) M_1^s \oplus P_i^s$;
      Check-Domain-Range(PP_i^s, CC_i^s)
      $count \leftarrow count + 1$;
    endfor;
    return $P_1^s, P_2^s, \ldots, C_{m_s}^s$
  endif

Figure 6: Game RAND3

Respond to the s^{th} adversary query as follows:

AN ENCIPHER QUERY $\mathsf{Enc}(N^s; P_1^s, \ldots, P_{m_s}^s)$

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

$C^s = C_1^s, C_2^s, \ldots, C_{m_s}^s \xleftarrow{\$} \{0,1\}^{n\,m_s}$;

return C^s;

A DECIPHER QUERY $\mathsf{Dec}(N^s; C_1^s, \ldots, C_{m_s}^s)$

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

$P^s = P_1^s, P_2^s, \ldots, P_{m_s}^s \xleftarrow{\$} \{0,1\}^{n\,m_s}$;

return P^s;

Finalization:

FIRST PHASE

$\mathcal{D} \leftarrow \emptyset; \mathcal{R} \leftarrow \emptyset$;
for $s \leftarrow 1$ to q,
if $N^s = N^r$ for some $r < s$ **then**
$EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
else
$EN^s \xleftarrow{\$} \{0,1\}^n$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{N^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{EN^s\}$;
$EEN^s \xleftarrow{\$} \{0,1\}^n$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{xEN^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{EEN^s\}$;
endif

if $ty^s = \mathsf{Enc}$
$MP^s \leftarrow P_1^s \oplus P_2^s \oplus \ldots \oplus P_{m_s}^s$;
if $m_s = 1$, **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0}^s \leftarrow C_1^s \oplus xEEN^s$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,0}^s\}$;
endif

if $m_s = 2$ **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0} \xleftarrow{\$} \{0,1\}^n$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,0}^s\}$;
$PP_1^s \leftarrow M_{1,0}^s \oplus P_1^s$; $PP_2^s \leftarrow M_{1,0}^s \oplus EEN^s \oplus P_2$;
$MC^s \leftarrow C_1^s \oplus C_2^s \oplus EN^s$; $MCC^s \leftarrow MC^s \oplus EEN^s$;
$M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,0}^s\}$;
$CC_1^s \leftarrow C_1^s \oplus M_{2,0}^s$; $CC_2^s \leftarrow C_2^s \oplus M_{2,0}^s \oplus EN^s$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{PP_1^s, PP_2^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC_1^s, CC_2^s\}$;
endif

if $m_s > 2$ **then**
$MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0} \xleftarrow{\$} \{0,1\}^n$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,0}^s\}$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s
if $count > n + 1$ **then**
$count \leftarrow 0$; $k \leftarrow k + 1$; $M_{1,k}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{M_{1,k-1}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,k}^s\}$;
endif
$PP_i^s \leftarrow p_i(x) M_{1,k}^s \oplus P_i^s$; $count \leftarrow count + 1$;
endfor
$MC^s \leftarrow C_1^s \oplus C_2^s \oplus \ldots \oplus C_{m_s}^s$;
$MCC^s \leftarrow MC^s \oplus EEN^s$; $M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,0}^s\}$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s,
if $count > n + 1$ **then**
$count \leftarrow 0$; $k \leftarrow k + 1$; $M_{2,k}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{M_{2,k-1}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,k}^s\}$;
endif
$CC_i^s \leftarrow p_i(x) M_2 \oplus C_i^s$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{PP_i^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC_i^s\}$;
$count \leftarrow count + 1$;
endfor;
endif
endfor

if $ty^s = \mathsf{Dec}$
$MC^s \leftarrow C_1^s \oplus C_2^s \oplus \ldots \oplus C_{m_s}^s$;
if $m_s = 1$ **then**
$MCC^s \leftarrow MC^s \oplus xEEN^s$;
$M_{2,0}^s = P_1^s \oplus EN^s$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{M_{2,0}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{MCC^s\}$;
endif

if $m_s = 2$ **then**
$MCC^s \leftarrow MC^s \oplus EN^s \oplus EEN^s$;
$M_{2,0}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,0}^s\}$;
$CC_1^s \leftarrow M_{2,0}^s \oplus C_1^s$; $CC_2^s \leftarrow M_{2,0}^s \oplus EN^s \oplus P_2$;
$MP^s \leftarrow P_1^s \oplus P_2^s$; $MPP^s \leftarrow MP^s \oplus EN^s$;
$M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,0}^s\}$;
$PP_1^s \leftarrow P_1^s \oplus M_{1,0}^s$; $PP_2^s \leftarrow P_2^s \oplus M_{1,0}^s \oplus EEN^s$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{PP_1^s, PP_2^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC_1^s, CC_2^s\}$;
endif

if $m_s > 2$ **then**
$MCC^s \leftarrow MC^s \oplus EEN^s$;
$M_{2,0} \xleftarrow{\$} \{0,1\}^n$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,0}^s\}$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s
if $count > n + 1$ **then**
$count \leftarrow 0$; $k \leftarrow k + 1$; $M_{2,k}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{M_{2,k-1}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{2,k}^s\}$;
endif
$CC_i^s \leftarrow p_i(x) M_{2,k}^s \oplus C_i^s$; $count \leftarrow count + 1$;
endfor
$MP^s \leftarrow P_1^s \oplus P_2^s \oplus \ldots \oplus P_{m_s}^s$;
$MPP^s \leftarrow MP^s \oplus EN^s$; $M_{1,0}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,0}^s\}$;
$count \leftarrow 0$; $k \leftarrow 0$;
for $i = 1$ to m_s,
if $count > n + 1$ **then**
$count \leftarrow 0$; $k \leftarrow k + 1$; $M_{1,k}^s \xleftarrow{\$} \{0,1\}^n$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{M_{1,k-1}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_{1,k}^s\}$;
endif
$PP_i^s \leftarrow p_i(x) M_1^s \oplus P_i^s$;
$\mathcal{D} \leftarrow \mathcal{D} \cup \{PP_i^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC_i^s\}$;
$count \leftarrow count + 1$;
endfor;
endif
endfor

SECOND PHASE

if (some value occurs more than once in $\mathcal{D}$) **then**
$bad \leftarrow$ true
endif
if (some value occurs more than once in $\mathcal{R}$) **then**
$bad \leftarrow$ true
endif

Figure 7: Game NON

$\mathcal{D} \leftarrow \emptyset; \mathcal{R} \leftarrow \emptyset;$
for $s \leftarrow 1$ to q,
  if $\mathsf{N}^s = \mathsf{N}^r$ for some $r < s$ **then**
    $EN^s \leftarrow EN^r$; $EEN^s \leftarrow EEN^r$
  else
    $EN^s \xleftarrow{\$} \{0,1\}^n$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathsf{N}^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{EN^s\}$;
    $EEN^s \xleftarrow{\$} \{0,1\}^n$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{xEN^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{EEN^s\}$;
  endif

if $ty^s = \mathsf{Enc}$
  $MP^s \leftarrow \mathsf{P}^s_1 \oplus \mathsf{P}^s_2 \oplus \ldots \oplus \mathsf{P}^s_{m_s}$;
  if $m_s = 1$, **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0} \leftarrow \mathsf{C}^s_1 \oplus xEEN^s$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,0}\}$;
  endif

  if $m_s = 2$ **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0} \xleftarrow{\$} \{0,1\}^n$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,0}\}$;
    $PP^s_1 \leftarrow M^s_{1,0} \oplus \mathsf{P}^s_1$; $PP^s_2 \leftarrow M^s_{1,0} \oplus EEN^s \oplus \mathsf{P}_2$;
    $MC^s \leftarrow \mathsf{C}^s_1 \oplus \mathsf{C}^s_2 \oplus EN^s$; $MCC^s \leftarrow MC^s \oplus EEN^s$;
    $M^s_{2,0} \xleftarrow{\$} \{0,1\}^n$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{2,0}\}$;
    $CC^s_1 \leftarrow \mathsf{C}^s_1 \oplus M^s_{2,0}$; $CC^s_2 \leftarrow \mathsf{C}^s_2 \oplus M^s_{2,0} \oplus EN^s$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{PP^s_1, PP^s_2\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC^s_1, CC^s_2\}$;
  endif

  if $m_s > 2$ **then**
    $MPP^s \leftarrow MP^s \oplus EN^s$;
    $M_{1,0} \xleftarrow{\$} \{0,1\}^n$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,0}\}$;
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M^s_{1,k} \xleftarrow{\$} \{0,1\}^n$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{M^s_{1,k-1}\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,k}\}$;
      endif
      $PP^s_i \leftarrow p_i(x) M^s_{1,k} \oplus \mathsf{P}^s_i$; $count \leftarrow count + 1$;
    endfor
    $MC^s \leftarrow \mathsf{C}^s_1 \oplus \mathsf{C}^s_2 \oplus \ldots \oplus \mathsf{C}^s_{m_s}$;
    $MCC^s \leftarrow MC^s \oplus EEN^s$; $M^s_{2,0} \xleftarrow{\$} \{0,1\}^n$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{2,0}\}$;
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s,
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M^s_{2,k} \xleftarrow{\$} \{0,1\}^n$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{M^s_{2,k-1}\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{2,k}\}$;
      endif
      $CC^s_i \leftarrow p_i(x) M_2 \oplus \mathsf{C}^s_i$;
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{PP^s_i\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC^s_i\}$;
      $count \leftarrow count + 1$;
    endfor;
  endif
endfor

if $ty^s = \mathsf{Dec}$
  $MC^s \leftarrow \mathsf{C}^s_1 \oplus \mathsf{C}^s_2 \oplus \ldots \oplus \mathsf{C}^s_{m_s}$;
  if $m_s = 1$ **then**
    $MCC^s \leftarrow MC^s \oplus xEEN^s$;
    $M^s_{2,0} = \mathsf{P}^s_1 \oplus EN^s$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{M^s_{2,0}\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{MCC^s\}$;
  endif

  if $m_s = 2$ **then**
    $MCC^s \leftarrow MC^s \oplus EN^s \oplus EEN^s$;
    $M^s_{2,0} \xleftarrow{\$} \{0,1\}^n$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{2,0}\}$;
    $CC^s_1 \leftarrow M^s_{2,0} \oplus \mathsf{C}^s_1$; $CC^s_2 \leftarrow M^s_{2,0} \oplus EN^s \oplus \mathsf{P}_2$;
    $MP^s \leftarrow \mathsf{P}^s_1 \oplus \mathsf{P}^s_2$; $MPP^s \leftarrow MP^s \oplus EN^s$;;
    $M^s_{1,0} \xleftarrow{\$} \{0,1\}^n$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,0}\}$;
    $PP^s_1 \leftarrow \mathsf{P}^s_1 \oplus M^s_{1,0}$; $PP^s_2 \leftarrow \mathsf{P}^s_2 \oplus M^s_{1,0} \oplus EEN^s$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{PP^s_1, PP^s_2\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC^s_1, CC^s_2\}$;
  endif

  if $m_s > 2$ **then**
    $MCC^s \leftarrow MC^s \oplus EEN^s$;
    $M_{2,0} \xleftarrow{\$} \{0,1\}^n$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MCC^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{20}\}$;
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M^s_{2,k} \xleftarrow{\$} \{0,1\}^n$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{M^s_{2,k-1}\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{2,k}\}$;
      endif
      $CC^s_i \leftarrow p_i(x) M^s_{2,k} \oplus \mathsf{C}^s_i$; $count \leftarrow count + 1$;
    endfor
    $MP^s \leftarrow \mathsf{P}^s_1 \oplus P^s_2 \oplus \ldots \oplus P^s_{m_s}$;
    $MPP^s \leftarrow MP^s \oplus EN^s$; $M^s_{1,0} \xleftarrow{\$} \{0,1\}^n$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{MPP^s\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,0}\}$;
    $count \leftarrow 0$; $k \leftarrow 0$;
    for $i = 1$ to m_s,
      if $count > n + 1$ **then**
        $count \leftarrow 0$; $k \leftarrow k + 1$; $M^s_{1,k} \xleftarrow{\$} \{0,1\}^n$;
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{M^s_{1,k-1}\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{M^s_{1,k}\}$;
      endif
      $PP^s_i \leftarrow p_i(x) M^s_{1,k} \oplus \mathsf{P}^s_i$;
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{PP^s_i\}$; $\mathcal{R} \leftarrow \mathcal{R} \cup \{CC^s_i\}$;
      $count \leftarrow count + 1$;
    endfor;
  endif
endfor

if (some value occurs more than once in $\mathcal{D}$) **then**
  $bad \leftarrow$ true
endif
if (some value occurs more than once in $\mathcal{R}$) **then**
  $bad \leftarrow$ true
endif