Attacks and Security Proofs of EAX-Prime

Kazuhiko Minematsu[1(✉)], Stefan Lucks[2], Hiraku Morita[3], and Tetsu Iwata[3]

[1] NEC Corporation, Kawasaki-Shi, Japan
k-minematsu@ah.jp.nec.com
[2] Bauhaus-Universität Weimar, Weimar, Germany
stefan.lucks@uni-weimar.de
[3] Nagoya University, Nagoya, Japan
h_morita@echo.nuee.nagoya-u.ac.jp, iwata@cse.nagoya-u.ac.jp

Abstract. EAX′ (or EAX-prime) is an authenticated encryption (AE) specified by ANSI C12.22 as a standard security function for Smart Grid. EAX′ is based on EAX proposed by Bellare, Rogaway, and Wagner. While EAX has a proof of security based on the pseudorandomness of the internal blockcipher, no published security result is known for EAX′. This paper studies the security of EAX′ and shows that there is a sharp distinction in security of EAX′ depending on the input length. EAX′ encryption takes two inputs, called cleartext and plaintext, and we present various efficient attacks against EAX′ using single-block cleartext and plaintext. At the same time we prove that if cleartexts are always longer than one block, it is provably secure based on the pseudorandomness of the blockcipher.

Keywords: Authenticated encryption · EAX · EAX′ · Attack · Provable security

1 Introduction

ANSI C12.22 [3] specifies a blockcipher mode for authenticated encryption (AE) as the standard security function for Smart Grid. It is called EAX′ (or EAX-prime)[1]. As its name suggests, EAX′ is based on EAX proposed by Bellare, Rogaway, and Wagner at FSE 2004 [7]. Though EAX is already efficient with a small amount of precomputation, EAX′ aims at even reducing the amount of precomputation and memory, for making it suitable to the resource-constrained devices, typically smart meters. ANSI submitted EAX′ to NIST [13] and NIST called for the public comments on the proposal to approve EAX′. Following

A part of the result was presented at DIAC [12].

[1] The authors of [13] exchangeably use the three names, EAX′, EAX', and EAX-prime, to mean their proposal. To avoid any confusion by overlooking the tiny prime symbol or apostrophe, which could be misunderstood as claiming an attack on EAX, we prefer the longer name "EAX-prime" for the title. In the text we prefer the name EAX′.

S. Moriai (Ed.): FSE 2013, LNCS 8424, pp. 327–347, 2014.
DOI: 10.1007/978-3-662-43933-3_17,

ANSI C12.22, IEEE 1703 [6] and MC1222 [4] included EAX′. There is also an RFC [5] related to ANSI C12.22.

Though EAX′ is similar to EAX, to the best of our knowledge, its formal security analysis has not been published to date. In this paper, we investigate the security of EAX′ and show that there is a sharp distinction depending on the input length. The encryption algorithm of EAX′ takes two inputs, called cleartext and plaintext. In the standard AE terminology, the cleartext serves as a nonce, or a combination of nonce and associated data (the latter is also called header).

First, we show that if the lengths of cleartext and plaintext are not exceeding one block, there exist attacks against EAX′ for both privacy and authenticity. Specifically, we present

- *forgeries*, i.e., cleartext/ciphertext pairs with valid authentication tags,
- *chosen-plaintext distinguishers*, distinguishing the EAX′ encryption from a random encryption process, and
- *chosen-ciphertext plaintext recovery attacks*, decrypting ciphertexts by asking for the decryption of another ciphertext with a valid authentication tag.

Our attacks are simple and efficient as they require only one or two queries. The simplest one even produces a successful forgery without observing any valid plaintext/ciphertext pair. Our forgery and distinguishing attacks strictly require the target system to accept one-block cleartext and plaintext. The plaintext recovery attacks relax this condition, and given any ciphertext with one-block cleartext it works for any circumstance where ciphertext is decrypted without checking the cleartext length. This makes the possibility of attack even larger. Our attacks imply that, while the original EAX has a proof of security, the security of EAX′ has totally collapsed as a general-purpose AE.

Next, we show that if the cleartext is always longer than one block, it recovers the provable security based on the pseudorandomness of the blockcipher for both privacy and authenticity notions. The security proof is obtained by combining previous proof techniques of EAX by Bellare, Rogaway, and Wagner [7] with some non-trivial extensions, such as Iwata and Kurosawa's one used for proving the security of OMAC [9].

One may naturally wonder if our attacks are applicable to ANSI C12.22. Unfortunately we do not know if ANCI C12.22 protocols exclude one-block cleartexts or not, hence we have no clear answer. Still, considering the effect of our attacks, we conclude that EAX′ must be used with cleartext length check mechanisms at both ends of encryption and decryption.

2 Preliminaries

Basic Notations. Let $\mathbb{N} = \{0, 1, \dots\}$. Let $\{0,1\}^*$ be the set of all finite-length binary strings, including the empty string ε. The bit length of a binary string X is written as $|X|$, and let $|X|_n \stackrel{\text{def}}{=} \lceil |X|/n \rceil$. Here $|\varepsilon| = 0$. A concatenation of $X, Y \in \{0,1\}^*$ is written as $X \| Y$ or simply XY. A sequence of a zeros

(ones) is denoted by 0^a (1^a). For $k \geq 0$, let $\{0,1\}^{>k} \stackrel{\text{def}}{=} \bigcup_{i=k+1,\ldots}\{0,1\}^i$ and $(\{0,1\}^n)^{>k} \stackrel{\text{def}}{=} \bigcup_{j=k+1,\ldots}(\{0,1\}^n)^j$, and $(\{0,1\}^n)^+ \stackrel{\text{def}}{=} (\{0,1\}^n)^{>0}$. We also define $\{0,1\}^{\geq k}$, $(\{0,1\}^n)^{\geq k}$, $\{0,1\}^{<k}$, $(\{0,1\}^n)^{<k}$, $\{0,1\}^{\leq k}$, and $(\{0,1\}^n)^{\leq k}$ analogously. For $X, Y \in \{0,1\}^n$, $X+Y$ or $X-Y$ is considered as an addition or a subtraction modulo 2^n.

For $X \in \{0,1\}^*$, let $X[1]\|X[2]\|\ldots\|X[m] \stackrel{n}{\leftarrow} X$ denote the n-bit block partitioning of X, i.e., $X[1]\|X[2]\|\ldots\|X[m] = X$ where $m = |X|_n$, and $|X[i]| = n$ for $i < m$ and $|X[m]| \leq n$. For $X, Y \in \{0,1\}^*$, let $X \oplus_{\text{end}} Y$ be the XOR of X into the end of Y if $|X| \leq |Y|$, i.e. $X \oplus_{\text{end}} Y = (0^{|Y|-|X|}\|X) \oplus Y$. Otherwise $X \oplus_{\text{end}} Y = X \oplus (0^{|X|-|Y|}\|Y)$.

For a finite set $\mathcal{X}$, if X is uniformly chosen from $\mathcal{X}$ we write $X \stackrel{\$}{\leftarrow} \mathcal{X}$.

Random Function and Random Permutation. Let $\text{Func}(n,m)$ be the set of all functions $\{0,1\}^n \to \{0,1\}^m$. We may abbreviate $\text{Func}(n,n)$ to $\text{Func}(n)$. In addition, let $\text{Perm}(n)$ be the set of all permutations over $\{0,1\}^n$. A uniform random function (URF) having n-bit input and m-bit output is the set $\text{Func}(n,m)$ with uniform distribution over $\text{Func}(n,m)$. It is denoted by R, and the corresponding sampling is written as $\mathsf{R} \stackrel{\$}{\leftarrow} \text{Func}(n,m)$. An n-bit uniform random permutation (URP) is the set $\text{Perm}(n)$ with uniform distribution over $\text{Perm}(n)$. It is denoted by P, and the corresponding sampling is written as $\mathsf{P} \stackrel{\$}{\leftarrow} \text{Perm}(n)$.

Galois Field. Following [7], an n-bit string X may be viewed as an element of $\text{GF}(2^n)$ by taking X as a coefficient vector of the polynomial in $\text{GF}(2^n)$. We write $2X$ to denote the multiplication of 2 and X over $\text{GF}(2^n)$, where 2 denotes the generator of the field $\text{GF}(2^n)$. This operation is called *doubling*. We also write $4L$ to denote $2(2L)$. The doubling is efficiently implemented by one-bit shift with conditional XOR of a constant, see e.g. [9].

3 Specification of EAX-Prime

We describe the encryption and decryption algorithms of EAX′. We changed the original notations of EAX′ [3,13] following those of EAX [7]. This illustrates the similarities and the differences of EAX and EAX′ (See also the last part of this section).

EAX′ is a mode of operation based on an n-bit blockcipher, E. Here we typically assume $(n, E) = (128, \text{AES-128})$, however other choice is possible [13]. The key of E is written as K. Formally, the encryption function of EAX′ accepts a cleartext, $N \in \{0,1\}^*$ with $N \neq \varepsilon$, a plaintext, $M \in \{0,1\}^*$, and a secret key, K, to produce the ciphertext, $C \in \{0,1\}^*$, with $|C| = |M|$ and the tag $T \in \{0,1\}^{32}$. The decryption function, which we also call the verification function, accepts N, C, T, and K and generates the decrypted plaintext M if (N, C, T) is valid, or the flag $\perp$ if invalid. Cleartext N contains information that needs to be authenticated, but not encrypted. ANSI document requires that N must

be unique for all encryptions using the same key[2]. Hence N can be seen as a combination of a nonce and associated data in the standard terminology of AE (e.g., see [7]). The plaintext M can be the empty string ε, corresponding to the null string in [13], and in this case EAX′ works as a message authentication code for N.

For generality we assume that the tag length is specified by a predetermined parameter, $\tau \in \{1, \ldots, n\}$. The original definition employs $\tau = 32$. Let EAX′$[E, \tau]$ be EAX′ using n-bit blockcipher E with τ-bit tag. The corresponding encryption and decryption algorithms are written as EAX′-$\mathcal{E}_{K,\tau}$ and EAX′-$\mathcal{D}_{K,\tau}$. If τ is clear from the context we may write EAX′$[E]$ and EAX′-$\mathcal{E}_K$ and EAX′-$\mathcal{D}_K$. These algorithms and their components are shown in Fig. 1. The encryption algorithm of EAX′ is depicted in Fig. 2. In Fig. 1, α denotes an n-bit constant, $(1^{n-32}\|01^{15}\|01^{15})$. Note that CBC'$_K(0^n, M)$ is equivalent to the standard CBC-MAC using E_K with input M, denoted by $\mathrm{CBC}_K(M)$. In our description, we fixed an apparent error in line 72 of the original definition of EAX′.encrypt$_K$ in [3,13]. Some editorial errors of [13] were also pointed out by [1].

EAX′ and the Original EAX. The major differences between EAX′ and the original EAX are summarized as follows. For other minor differences, see Section 3 of [13]. For the definition of EAX, see [7].

1. Role of N. Inputs to EAX′-$\mathcal{E}_K$ consist of a cleartext N and a plaintext M, whereas those to the original EAX consist of a nonce N, a header (or associated data) H, and a plaintext M. EAX′ requires N to be unique, hence it works as a nonce. EAX′ does not explicitly define a header H; information corresponding to the header is included in the cleartext N.
2. Tweaking method for CMAC. For input M, CMAC [2] using E_K is defined as $\mathrm{CMAC}_K(M) = \mathrm{CBC}_K(\mathtt{pad}(M; D, Q))$. The original EAX uses the tweaked CMAC having an n-bit tweak t, defined as $\mathrm{CMAC}_K(t\|M)$, for $t \in \{0^n, 0^{n-1}1, 0^{n-2}10\}$, to process N, H, and C. For fast operation we need to precompute $E_K(t)$ for all t and store them to RAM. EAX′ employs a different way to tweak CMAC accepting two tweak values $(i = 0, 1)$ to generate $\mathrm{CMAC}'^{(0)}_K$ and $\mathrm{CMAC}'^{(1)}_K$ for processing N and C. For fast operation we can precompute $L = E_K(0^n)$. This reduces the precomputation time and RAM consumption from the original EAX.
3. Counter mode incrementation. The original EAX uses $\mathrm{CMAC}_K(0^n\|N)$ as an initial counter block for CTR mode, while that of EAX′ is $\mathrm{CMAC}'^{(0)}_K(N) \wedge \alpha$ to set some bits to zero. One can find a similar zeroing-out in the deterministic authenticated encryption called SIV [15]. As explained by [15], this contributes to a slight simpler operation.

[2] In ANSI C12.22, the uniqueness of N is guaranteed by including time information with a specific format.

Algorithm EAX′-$\mathcal{E}_{K,\tau}(N, M)$

1. $\underline{N} \leftarrow \mathrm{CMAC}'^{(0)}_K(N)$
2. $C \leftarrow \mathrm{CTR}'_K(\underline{N}, M)$
3. $\underline{T} \leftarrow \underline{N} \oplus \mathrm{CMAC}'^{(1)}_K(C)$
4. $T \leftarrow \mathrm{msb}_\tau(\underline{T})$
5. **return** (C, T)

Algorithm EAX′-$\mathcal{D}_{K,\tau}(N, C, T)$

1. $\underline{N} \leftarrow \mathrm{CMAC}'^{(0)}_K(N)$
2. $\underline{T} \leftarrow \underline{N} \oplus \mathrm{CMAC}'^{(1)}_K(C)$
3. $\widehat{T} \leftarrow \mathrm{msb}_\tau(\underline{T})$
4. **if** $\widehat{T} \neq T$ **return** $\perp$
5. $M \leftarrow \mathrm{CTR}'_K(\underline{N}, C)$
6. **return** M

Algorithm $\mathrm{CMAC}'^{(i)}_K(M)$ (for $i \in \{0, 1\}$)

1. $L \leftarrow E_K(0^n)$
2. $D \leftarrow 2L$, $Q \leftarrow 4L$
3. **if** $i = 0$ **then**
4. **return** $\mathrm{CBC}'_K(D, \mathtt{pad}(M; D, Q))$
5. **if** $i = 1$ **then**
6. **return** $\mathrm{CBC}'_K(Q, \mathtt{pad}(M; D, Q))$

Algorithm $\mathrm{CBC}'_K(I, M)$ (for $M \in (\{0,1\}^n)^+$)

1. $M[1]\|M[2]\|\ldots\|M[m] \xleftarrow{n} M$
2. $C[0] \leftarrow I$
3. **for** $i \leftarrow 1$ to m **do**
4. $\quad C[i] \leftarrow E_K(M[i] \oplus C[i-1])$
5. **return** $C[m]$

Algorithm $\mathrm{CTR}'_K(\underline{N}, M)$

1. $m \leftarrow |M|_n$
2. $\underline{N}^{\wedge} \leftarrow \underline{N} \wedge \alpha$
3. $S \leftarrow E_K(\underline{N}^{\wedge})\|\cdots\|E_K(\underline{N}^{\wedge} + m - 1)$
4. $C \leftarrow M \oplus \mathrm{msb}_{|M|}(S)$
5. **return** C

Algorithm $\mathtt{pad}(M; B_1, B_2)$

1. **if** $|M| \in \{n, 2n, 3n, \ldots, \}$
2. **then return** $M \oplus_{\mathrm{end}} B_1$
3. **else**
4. **return** $(M\|10^{n-1-(|M| \bmod n)}) \oplus_{\mathrm{end}} B_2$

Fig. 1. (Upper) The encryption and decryption algorithms of EAX′$[E, \tau]$, originally with $\tau = 32$. (Lower) Component algorithms of EAX′$[E, \tau]$. Here, $\alpha = (1^{n-32}\|01^{15}\|01^{15})$.

4 Attacks Based on One-Block Cleartext

4.1 Chosen-Message Forgeries

We first describe forgery attacks against EAX′$[E, \tau]$. Throughout the section D and Q denote $2L$ and $4L$ with $L = E_K(0^n)$. The adversary $\mathcal{A}$ we consider here can access both encryption and decryption (verification) oracles, namely EAX′-$\mathcal{E}_K$ and EAX′-$\mathcal{D}_K$. Suppose $\mathcal{A}$ (possibly adaptively) asks q queries to the encryption oracle, $(N_1, M_1), \ldots, (N_q, M_q)$, and receives $(C_1, T_1), \ldots, (C_q, T_q)$, and then asks (N, C, T) to the decryption oracle. We say $\mathcal{A}$ is successful if $\mathcal{A}$ receives a string other than $\perp$ and $(N, C, T) \neq (N_i, C_i, T_i)$ for any $1 \leq i \leq q$ (see also Sect. 5). Here we assume the nonce-respecting adversary [14]; it is allowed to query any (N_i, M_i) to the encryption oracle as long as N_i is unique.

Suppose $M \in \{0,1\}^{\leq n}$. Then $\mathtt{pad}(M; D, Q) = M \oplus_{\mathrm{end}} D = M \oplus D$ when $|M| = n$ and $\mathtt{pad}(M; D, Q) = M\|10^{n-1-|M|} \oplus_{\mathrm{end}} Q = M\|10^{n-1-|M|} \oplus Q$ when $0 \leq |M| < n$. Therefore, the definition of $\mathrm{CMAC}'^{(i)}_K$ in the previous section

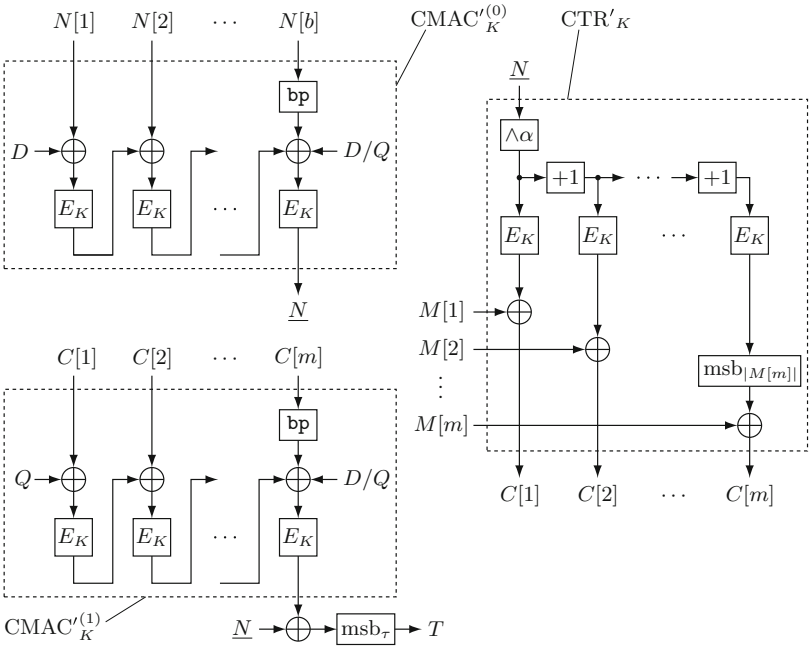

Fig. 2. The encryption algorithm of EAX′. In the figure, $|N|_n = b$ and $|M|_n = m$. $\mathtt{bp}(x) = x$ if $|x| = n$ and $\mathtt{bp}(x) = x\|10^{n-1-(|x| \bmod n)}$ if $|x| < n$.

conforms to that

$$\mathrm{CMAC}'^{(0)}_K(M) = \begin{cases} E_K(M) & \text{if } |\mathrm{M}| = \mathrm{n} \\ E_K(M\|10^{n-1-|M|} \oplus D \oplus Q) & \text{if } 0 \le |\mathrm{M}| < \mathrm{n} \end{cases}$$

$$\mathrm{CMAC}'^{(1)}_K(M) = \begin{cases} E_K(M \oplus D \oplus Q) & \text{if } |\mathrm{M}| = \mathrm{n} \\ E_K(M\|10^{n-1-|M|}) & \text{if } 0 \le |\mathrm{M}| < \mathrm{n} \end{cases}$$

The above observation immediately gives the following attacks:

Forgery attack 1 ($|N| = n$ and $|C| < n$)**.**

1. Prepare (N, C) such that $|N| = n$ and $|C| < n$ and $C\|10^{n-1-|C|} = N$.
2. Query (N, C, T) to the verification oracle, where $T = 0^\tau$.

This attack always succeeds as the "valid" tag for (N, C) is $\mathrm{msb}_\tau(E_K(N) \oplus E_K(C\|10^{n-1-|C|})) = 0^\tau$.

Forgery attack 2 ($|N| < n$ and $|C| = n$)**.**

1. Prepare (N, C) such that $|N| < n$, $|C| = n$, and $N\|10^{n-1-|N|} = C$.
2. Query (N, C, T) to the verification oracle, where $T = 0^\tau$.

The attack is again successful as the valid tag for (N, C) is $\mathrm{msb}_\tau(E_K(D \oplus Q \oplus N\|10^{n-1-|N|}) \oplus E_K(Q \oplus D \oplus C)) = 0^\tau$. These attacks use only one forgery attempt and no encryption query. By using one encryption query the forgery attack is possible even when $|N| = n$ and $|C| = n$:

Forgery attack 3 ($|N| = |M| = n$)**.**

1. Query (N, M) with $|N| = |M| = n$ and $N \neq 0^n$ to the encryption oracle.
2. Obtain (C, T) (where $|C| = n$) from the oracle and see if $C \neq 0^n$ (quit if $C = 0^n$).
3. Query $(\widetilde{N}, \widetilde{C}, \widetilde{T})$ to the verification oracle, where $|\widetilde{N}| < n$, $\widetilde{N}\|10^{n-1-|\widetilde{N}|} = C$, $|\widetilde{C}| < n$, $\widetilde{C}\|10^{n-1-|\widetilde{C}|} = N$, and $\widetilde{T} = T$.

The above attack is almost always successful; unless $C = 0^n$ we have $T = \mathrm{msb}_\tau(E_K(N) \oplus E_K(Q \oplus D \oplus C))$ and the valid tag for $(\widetilde{N}, \widetilde{C})$ is

$$\begin{aligned}&\mathrm{msb}_\tau(E_K(D \oplus Q \oplus \widetilde{N}\|10^{n-1-|\widetilde{N}|}) \oplus E_K(Q \oplus Q \oplus \widetilde{C}\|10^{n-1-|\widetilde{C}|}))\\ &= \mathrm{msb}_\tau(E_K(D \oplus Q \oplus C) \oplus E_K(N)),\end{aligned}$$

thus equals to T. The converse of Forgery attack 3 is also possible for $|N| < n$ and $|M| < n$:

Forgery attack 4 ($|N| < n$ and $|M| < n$)**.**

1. Query (N, M) with $|N| < n$ and $|M| < n$ to the encryption oracle.
2. Obtain (C, T) (where $|C| = |M| < n$) from the oracle.
3. Query $(\widetilde{N}, \widetilde{C}, \widetilde{T})$ to the verification oracle, where $|\widetilde{N}| = |\widetilde{C}| = n$, $\widetilde{N} = C\|10^{n-1-|C|}$, $\widetilde{C} = N\|10^{n-1-|N|}$, and $\widetilde{T} = T$.

We have $T = \mathrm{msb}_\tau(E_K(D \oplus Q \oplus N\|10^{n-1-|N|}) \oplus E_K(Q \oplus Q \oplus C\|10^{n-1-|C|}))$ and the valid tag for $(\widetilde{N}, \widetilde{C})$ is

$$\begin{aligned}&\mathrm{msb}_\tau(E_K(D \oplus D \oplus \widetilde{N}) \oplus E_K(Q \oplus D \oplus \widetilde{C}))\\ &= \mathrm{msb}_\tau(E_K(C\|10^{n-1-|C|}) \oplus E_K(Q \oplus D \oplus N\|10^{n-1-|N|})) = T.\end{aligned}$$

Partially Selective Forgeries. A forgery is *selective* instead of *existential*, if the adversary can determine the content of the message to be forged. Since EAX′ provides *authenticated encryption with associated data* (AEAD), the content of the message consists of both the confidential plaintext M and the non-confidential associated data (or cleartext) N. While the above attacks do not allow to choose M, the adversary can arbitrarily choose N (restricted to $|N| \leq n$ and, for $|N| = n$, $N \neq 0^n$). In this sense, the forgery attacks above are *partially selective*.

4.2 Chosen-Plaintext Distinguishers

The forgery attacks above are based on the idea of generating (N, C) that makes the tag $T = 0^\tau$. To distinguish EAX′-$\mathcal{E}_K$ from a random encryption process,

which produces $(|M| + \tau)$-bit random sequence on receiving (N, M), one can similarly make (N, M) so that EAX′-$\mathcal{E}_K$ will generate (C, T) with $T = 0^\tau$.

Distinguishing attack 1 ($|N| = n$ and $|M| = 0$)**.**

1. Query (N, M) to the encryption oracle, where $N = 10^{n-1}$ and $M = \varepsilon$.
2. Obtain (C, T) from the oracle with $C = \varepsilon$.
3. If $T = 0^\tau$ then return 1, otherwise return 0.

As EAX′-$\mathcal{E}_K$ returns $T = 0^\tau$ with probability 1 while the same event occurs with probability $1/2^\tau$ with a random encryption process, this enables us to easily distinguish T from random with the distinguishing advantage almost 1, using only one encryption query.

Distinguishing attack 2 ($|N| = n$, $1 \le |M| < n$, and fixed i for $1 \le i \le n-1$)**.**

1. Fix $M \in \{0,1\}^i$, and query (N, M) to the encryption oracle with $N = M \| 10^{n-1-|M|}$.
2. Obtain (C, T) from the oracle.
3. If $C = M$ and $T = 0^\tau$ then return 1, otherwise return 0.

In this case, we have $C = M$ with probability $1/2^i$ for both EAX′-$\mathcal{E}_K$ and a random encryption process. Given the event $C = M$, we have

$$T = \mathrm{msb}_\tau(E_K(N) \oplus E_K(C \| 10^{n-1-|C|})) = 0^\tau$$

with probability 1 for EAX′-$\mathcal{E}_K$, while $T = 0^\tau$ occurs with probability $1/2^\tau$ for the random encryption process. Thus, with probability $1/2^i$ the distinguisher succeeds with a high probability, which is non-negligible when i is small.

4.3 Chosen-Ciphertext Plaintext Recovery Attacks

Consider a triple (N^*, C^*, T^*) of cleartext N^*, ciphertext C^* and tag T^*. The corresponding plaintext M^* is unknown. The adversary can ask a decryption oracle, for the decryption of any (N, C, T) under its choice, except for $(N, C, T) = (N^*, C^*, T^*)$ (otherwise, finding M^* would be trivial). The adversary receives either $\perp$ (if verification fails) or the decryption M of C. This is the setting in a *chosen ciphertext attack*. Below, we focus on *plaintext recovery attacks*, where the adversary actually finds (a part of) M^*. We describe two attacks: the first for $|N^*| = n$, the second for $|N^*| < n$.

Plaintext recovery attack 1 ($|N^*| = n$)**.**

1. Obtain (N^*, C^*, T^*) for unknown plaintext M^*.
2. Prepare C with $|C| < n$ and $C \| 10^{n-1-|C|} = N^*$ and $T = 0^\tau$.
3. Query (N^*, C, T) to the decryption oracle. Let M be the answer.
4. Compute the keystream $KS = C \oplus M \in \{0,1\}^{|C|}$.

Since the decryption of (N^*, C^*, T^*) uses the same keystream KS, we now can compute the first $|C|$ bits of M^*, or the full M^* if $|M^*| \le |C|$. It succeeds for the same reason as Forgery attack 1 (unless $N^* = 0^n$, in which case there is no C in Step 2, or $C^* \| 10^{n-1-|C^*|} = N^*$ and $T^* = 0^\tau$, in which case the decryption query in Step 3 makes the attack trivial).

Plaintext recovery attack 2 ($|N^*| < n$)**.**

1. Obtain (N^*, C^*, T^*) for unknown plaintext M^*.
2. Prepare C with $|C| = n$ and $N^* \| 10^{n-1-|N^*|} = C$ and $T = 0^\tau$.
3. Query (N^*, C, T) to the decryption oracle. Let M be the answer.
4. Compute the keystream $KS = C \oplus M \in \{0,1\}^n$.

Unless $N^* \| 10^{n-1-|N^*|} = C^*$ and $T^* = 0^\tau$, the attack succeeds for the same reason as Forgery attack 2.

4.4 Remarks

The Source of Attacks. Not to mention, our attacks cannot be applied on the original EAX having the proof of security. Our attacks exploit the wrong tweaking method of CMAC in EAX′. While the tweaking method in the original EAX provides a set of computationally independent PRFs, the tweaking method of EAX′ fails to do this. For instance ${\mathrm{CMAC}'}_K^{(0)}(M) = {\mathrm{CMAC}'}_K^{(1)}(M')$ holds with probability 1 for any (M, M') such that $|M| = n$ and $|M'| < n$ and $M' \| 10^{n-1-|M'|} = M$, which is unlikely to occur if ${\mathrm{CMAC}'}_K^{(0)}$ and ${\mathrm{CMAC}'}_K^{(1)}$ were computationally independent. The SIV-like counter incrementation also increases the collision probability of counter blocks, however this only leads to a small degradation in security, as mentioned by [3], hence our attacks do not rely on this fact.

Applicability to ANSI C12.22 Protocols. All our attacks require $|N| \leq n$. The forgery and distinguishing attacks also require $|M|, |C| \leq n$, and the plaintext recovery attacks actually require at most the first n bits of the ciphertext. In addition, the forgery and plaintext recovery attacks could not be prevented by restricting the input length at encryption: one must implement the input length check at decryption as well.

One can find some examples that have $|M| = n$ or $|M| = 0$ (i.e. the authentication of N) with $n = 128$ in communication examples of ANSI C12.22 (Annex G of [3]) or test vectors[3] of EAX′ (Section V of [13]). At the same time, we do not know[4] whether $|N| > n$ holds for ANSI C12.22 protocols, even though the specification [13] does not, at least explicitly, regulate the length of cleartext. The reference code of EAX′ given by [3,6] has no restriction on input lengths, and we verified our attacks with that code.

A natural question arises from the above observation: whether EAX′ is provably secure under the restriction $|N| > n$. In the next section we provide a positive answer to this question.

[3] One can find test vectors with n-bit cleartexts in [13]. However, they seem to contain an editorial error; the cleartext may mean the plaintext and vice versa.

[4] In [13], "Justification" of Issue 6 (in page 3) states that "The CMAC′ computations here always involve CBC of at least two blocks". This looks odd since M or C can be null (as stated by ANSI) and CMAC′ taking the empty string certainly operates on the single-block CBC, but it may be a hint that $|N| > n$ would hold for any legitimate ANSI C12.22 messages.

5 Provable Security for More-Than-One-Block Cleartext

Now we are going to prove that EAX′ provides the provable security when the cleartext N is always more than n bits for both encryption and decryption. Combined with the attacks described in the previous section, the result of this section draws a sharp distinction on the security between the case $|N| > n$ and the case $|N| \leq n$.

Security Notions. Following [7,14], we introduce two security notions, privacy and authenticity, to model the security of EAX′. For c oracles, $O_1, O_2, \ldots, O_c$, we write $\mathcal{A}^{O_1,O_2,\ldots,O_c}$ to represent the adversary $\mathcal{A}$ accessing these c oracles in an arbitrarily order. If F and G are oracles having the same input and output domains, we say they are compatible.

A CPA-adversary $\mathcal{A}$ against EAX′$[E,\tau]$ accesses EAX′-$\mathcal{E}_{K,\tau}$. The encryption queries made by $\mathcal{A}$ are denoted by $(N_1, M_1), \ldots, (N_q, M_q)$. We define $\mathcal{A}$'s parameter list as (q, σ_N, σ_M), where $\sigma_N \stackrel{\text{def}}{=} \sum_{i=1}^{q} |N_i|_n$ and $\sigma_M \stackrel{\text{def}}{=} \sum_{i=1}^{q} |M_i|_n$ if all $|M_i|_n > 0$. For convention, if $|M_i| = 0$ for some $i \leq q$, $\sigma_M \stackrel{\text{def}}{=} (\sum_{i=1}^{q} |M_i|_n) + 1$. We also define random-bit oracle, \$, which takes $(N, M) \in \{0,1\}^* \times \{0,1\}^*$ and returns $(C, T) \stackrel{\$}{\leftarrow} \{0,1\}^{|M|} \times \{0,1\}^\tau$. The privacy notion for CPA-adversary $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}'[E,\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathrm{EAX}'\text{-}\mathcal{E}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1]. \tag{1}$$

We assume $\mathcal{A}$ in the privacy notion is nonce-respecting, i.e., all N_is are distinct. Similarly, a CCA-adversary $\mathcal{A}$ against EAX′$[E,\tau]$ accesses EAX′-$\mathcal{E}_{K,\tau}$ and EAX′-$\mathcal{D}_{K,\tau}$. The encryption and decryption queries made by $\mathcal{A}$ are denoted by $(N_1, M_1), \ldots, (N_q, M_q)$ and $(\widetilde{N}_1, \widetilde{C}_1, \widetilde{T}_1), \ldots, (\widetilde{N}_{q_v}, \widetilde{C}_{q_v}, \widetilde{T}_{q_v})$. We define $\mathcal{A}$'s parameter list as $(q, q_v, \sigma_N, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{C}})$, where $\sigma_{\widetilde{N}} \stackrel{\text{def}}{=} \sum_{i=1}^{q_v} |\widetilde{N}_i|_n$, $\sigma_{\widetilde{C}} \stackrel{\text{def}}{=} \sum_{i=1}^{q_v} |\widetilde{C}_i|_n$ when all $|\widetilde{C}_i|_n > 0$ and $\sigma_{\widetilde{C}} \stackrel{\text{def}}{=} (\sum_{i=1}^{q_v} |\widetilde{C}_i|_n) + 1$ otherwise. The definitions of σ_N and σ_M are the same as above. The authenticity notion for a CCA-adversary $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}'[E,\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathrm{EAX}'\text{-}\mathcal{E}_K, \mathrm{EAX}'\text{-}\mathcal{D}_K} \text{ forges}], \tag{2}$$

where $\mathcal{A}$ forges if EAX′-$\mathcal{D}_K$ returns a bit string (other than $\perp$) for a query $(\widetilde{N}_i, \widetilde{C}_i, \widetilde{T}_i)$ for some $1 \leq i \leq q_v$ such that $(\widetilde{N}_i, \widetilde{C}_i, \widetilde{T}_i) \neq (N_j, C_j, T_j)$ for all $1 \leq j \leq q$. We assume $\mathcal{A}$ in the authenticity notion is always nonce-respecting with respect to encryption queries; using the same N for encryption and decryption queries is allowed, and the same N can be repeated within decryption queries, i.e. N_i is different from N_j for any $j \neq i$ but $\widetilde{N}_i$ may be equal to N_j or $\widetilde{N}_{i'}$ for some j and $i' \neq i$.

Bounds. We denote EAX′ with an n-bit URP being used as a blockcipher by EAX′$[\mathrm{Perm}(n), \tau]$ and the corresponding encryption and decryption functions by EAX′-$\mathcal{E}_{\mathsf{P}}$ and EAX′-$\mathcal{D}_{\mathsf{P}}$. Similarly, the subscript K in the component algorithms is substituted with P, e.g. $\mathrm{CMAC}'^{(i)}_{\mathsf{P}}$. We here provide the security bounds for

$\mathrm{EAX}'[\mathrm{Perm}(n), \tau]$; the computational counterpart for $\mathrm{EAX}'[E, \tau]$ is trivial. The security bound for the privacy notion is as follows.

Theorem 1. *Let $\mathcal{A}$ be the CPA-adversary against $EAX'[Perm(n), \tau]$ who does not query cleartexts of n bits or shorter and has parameter list (q, σ_N, σ_M). Let $\sigma_{\mathrm{priv}} = \sigma_N + \sigma_M$. Then we have*

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}'[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq \frac{18\sigma^2_{\mathrm{priv}}}{2^n}.$$

The security bound for the authenticity notion is as follows.

Theorem 2. *Let $\mathcal{A}$ be the CCA-adversary against $EAX'[Perm(n), \tau]$ who does not query cleartexts of n bits or shorter for both encryption and decryption oracles, and has parameter list $(q, q_v, \sigma_N, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{C}})$. Let $\sigma_{\mathrm{auth}} = \sigma_N + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{C}}$. Then we have*

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}'[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq \frac{18\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q_v}{2^\tau}.$$

6 Proofs of Theorem 1 and Theorem 2

6.1 Overview

The proofs of Theorems 1 and 2 are bit long, hence we first provide the overview. The basic strategy follows from the proof of the original EAX [7] with some extensions taken from OMAC proofs [9,10]. We first break down the algorithm of $\mathrm{EAX}'[\mathrm{Perm}(n), \tau]$ into a pair of functions, which we call OMAC-extension, $\text{OMAC-e}[\mathsf{P}] = (\text{OMAC-e}[\mathsf{P}]^{(0)}, \text{OMAC-e}[\mathsf{P}]^{(1)})$, where $\text{OMAC-e}[\mathsf{P}]^{(0)} : \{0,1\}^{>n} \times \mathbb{N} \to (\{0,1\}^n)^{>0}$ and $\text{OMAC-e}[\mathsf{P}]^{(1)} : \{0,1\}^* \to \{0,1\}^n$. It uses an n-bit random permutation P and an additional independent and random value, $U \in \{0,1\}^n$. Intuitively, $\text{OMAC-e}[\mathsf{P}]^{(0)}$ is a function that takes (N, d), where $d = |M|_n$ ($d = |C|_n$) for encryption (decryption), and produces $\underline{N} \oplus U$ and the d-block keystream before truncation, i.e., S of Fig. 1 (See also Fig. 2). Similarly, $\text{OMAC-e}[\mathsf{P}]^{(1)}$ takes a ciphertext, C, and produces $\mathrm{CMAC}'^{(1)}_{\mathsf{P}}(C) \oplus U$. Since $(\underline{N} \oplus U) \oplus (\mathrm{CMAC}'^{(1)}_{\mathsf{P}}(C) \oplus U) = \underline{N} \oplus \mathrm{CMAC}'^{(1)}_{\mathsf{P}}(C)$, such a function pair can perfectly simulate $\mathrm{EAX}'[\mathrm{Perm}(n), \tau]$. We introduce U to make the remaining analysis less involved. Then, the bound evaluation for $\mathrm{EAX}'[\mathrm{Perm}(n), \tau]$ is mostly reduced to that of the indistinguishability between OMAC-e[P] and a random function pair $\mathbb{RND} = (\mathbb{RND}^{(0)}, \mathbb{RND}^{(1)})$. Here $\mathbb{RND}^{(0)}$ takes (N, d) and samples $Y \xleftarrow{\$} (\{0,1\}^n)^{d_{\max}+1}$ if N is new, and outputs the first $(d+1)$ blocks of Y, where $d_{\max}$ is the maximum possible value of d implied by the game we consider. Similarly $\mathbb{RND}^{(1)}$ takes $C \in \{0,1\}^*$ and outputs $Y' \xleftarrow{\$} \{0,1\}^n$ if C is new. To bound the indistinguishability between OMAC-e[P] and $\mathbb{RND}$, we further break down OMAC-e[P] into a set of ten small functions, $\mathbf{Q} = \{\mathbf{Q}_i\}_{i=1,\ldots,10}$, following the proof of OMAC [9]. Using two random values in addition to U, these functions are built so that they behave close to a set of independent URFs or URPs, and

at the same time have the capability to perfectly simulate OMAC-e[P] (hence EAX′[Perm(n)]). The indistinguishability of $\mathbf{Q}$ from the set of URPs/URFs is relatively easy to derive, and as a result the following analysis becomes much easier.

6.2 Proof

Setup. Without loss of generality and for simplicity this section assumes that the space of valid cleartexts of EAX′ is $\{0,1\}^{>n}$, rather than restricting the adversary's strategy.

For convenience we introduce the following notions. Let $F_K : \mathcal{X} \to \mathcal{Y}$ and $G_{K'} : \mathcal{X} \to \mathcal{Y}$ be two keyed functions with $K \in \mathcal{K}$ and $K' \in \mathcal{K}'$, and let $\mathcal{A}$ be the CPA-adversary. We define

$$\mathtt{Adv}^{\mathtt{cpa}}_{F,G}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{G_{K'}} \Rightarrow 1]. \quad (3)$$

Note that this definition can be naturally extended when $G_{K'}$ is substituted with the random-bit oracle compatible with F_K. Moreover, when F_K and $G_{K'}$ are compatible with EAX′-$\mathcal{E}_K$, we define $\mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{F,G}(\mathcal{A})$ as the same function as $\mathtt{Adv}^{\mathtt{cpa}}_{F,G}(\mathcal{A})$ but CPA-adversary $\mathcal{A}$ is restricted to be nonce-respecting. Let $\mathbf{F} = (F^e_K, F^d_K)$ and $\mathbf{G} = (G^e_{K'}, G^d_{K'})$ be the pairs of functions that are compatible with (EAX′-$\mathcal{E}_K$, EAX′-$\mathcal{D}_K$). We define

$$\mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{\mathbf{F},\mathbf{G}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F^e_K, F^d_K} \Rightarrow 1] - \Pr[K' \xleftarrow{\$} \mathcal{K}' : \mathcal{A}^{G^e_{K'}, G^d_{K'}} \Rightarrow 1], \quad (4)$$

where the underlying $\mathcal{A}$ is assumed to be nonce-respecting for encryption queries. Note that we have $\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}'[E,\tau]}(\mathcal{A}) = \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathrm{EAX}'\text{-}\mathcal{E}_K,\$}(\mathcal{A})$ for any nonce-respecting CPA-adversary $\mathcal{A}$.

Step 1: OMAC-extension. For $x \in \{0,1\}^{\le n}$, let $\mathtt{bp}(x) = x$ if $|x| = n$ and $\mathtt{bp}(x) = x \| 10^{n-1-(|x| \bmod n)}$ if $|x| < n$. If $x = \varepsilon$ then $\mathtt{bp}(x) = 10^{n-1}$. We first define OMAC-extension using an n-bit URP, denoted by OMAC-e[P] : $\{0,1\} \times \{0,1\}^* \times \mathbb{N} \to (\{0,1\}^n)^{>0}$. The definition is given in Fig. 3. See also Fig. 4. Actually it consists of two functions, written as

$$\text{OMAC-e[P]}^{(0)} : \{0,1\}^{>n} \times \mathbb{N} \to (\{0,1\}^n)^{>0}, \text{ and} \quad (5)$$

$$\text{OMAC-e[P]}^{(1)} : \{0,1\}^* \to \{0,1\}^n, \quad (6)$$

where the first argument to OMAC-e[P], $t \in \{0,1\}$, specifies which function to be used, i.e., OMAC-e[P]$(0,X,d) =$ OMAC-e[P]$^{(0)}(X,d)$ and OMAC-e[P]$(1,X,d) =$ OMAC-e[P]$^{(1)}(X)$ (d is discarded). Here $|\text{OMAC-e[P]}^{(0)}(X,d)| = (d+1)n$. For simplicity we assume the input domain of OMAC-e[P] is a set of $(t,X,d) \in \{0,1\} \times \{0,1\}^* \times \mathbb{N}$ that is acceptable for OMAC-e[P]$^{(t)}$. More formally, when $t = 0$ we assume $|X| > n$ and $d \in \mathbb{N}$, and when $t = 1$ we assume d is fixed (say 0). As described in Sect. 6.1, OMAC-e[P] enables us to simulate EAX′-$\mathcal{E}_{\mathsf{P}}$ and EAX′-$\mathcal{D}_{\mathsf{P}}$; note that the simulator only needs to compute the sum of two outputs from

$\mathrm{CMAC}'^{(0)}_{\mathsf{P}}$ and $\mathrm{CMAC}'^{(1)}_{\mathsf{P}}$, and not to compute the output itself. For instance, if we want to perform EAX′-$\mathcal{E}_{\mathsf{P}}$ for $N = (N[1]\|N[2])$ and $M = (M[1]\|M[2])$ with $|N[1]| = |N[2]| = |M[1]| = n$ and $|M[2]| = n - 2$, then the procedure is (1) $Y\|S[1]S[2] \leftarrow$ OMAC-e[P]$(0, N, 2)$, (2) $C \leftarrow \mathrm{msb}_{2n-2}(S[1]S[2]) \oplus M$, (3) $Y' \leftarrow$ OMAC-e[P]$(1, C, 0)$, where the last argument is arbitrary, (4) $T \leftarrow \mathrm{msb}_{\tau}(Y \oplus Y')$, and (5) output (C, T). The following proposition is easy to check.

Proposition 1. *There exist deterministic procedures, $f_e(\cdot)$ and $f_d(\cdot)$, that use OMAC-e[P] as a black box and perfectly simulate EAX′-$\mathcal{E}_{\mathsf{P}}$ and EAX′-$\mathcal{D}_{\mathsf{P}}$. That is, we have*[5] *EAX′-$\mathcal{E}_{\mathsf{P}} \equiv f_e($OMAC-e[P]$)$ and EAX′-$\mathcal{D}_{\mathsf{P}} \equiv f_d($OMAC-e[P]$)$.*

A keyed function F compatible with OMAC-e[P] is said to have OMAC-e profile, and we denote $F(t, X, d)$ by $F^{(t)}(X, d)$. Suppose an adversary querying F of OMAC-e profile has q queries $(t_1, X_1, d_1), \ldots, (t_q, X_q, d_q)$ and corresponding answers are $Y_1, \ldots, Y_q$. Such an adversary is called to be with parameter list $(q, \sigma_{\mathrm{in}}, \sigma_{\mathrm{out}})$ where $\sigma_{\mathrm{in}} \stackrel{\mathrm{def}}{=} \sum_{i=1,\ldots,q} |X_i|_n$ and $\sigma_{\mathrm{out}} \stackrel{\mathrm{def}}{=} \sum_{i=1,\ldots,q;t_i=0} |Y_i|_n$.

Algorithm OMAC-e[P]:
Initialization
00 $L \leftarrow \mathsf{P}(0^n)$, $U \stackrel{\$}{\leftarrow} \{0,1\}^n$
On query $(t, X, d) \in \{0,1\} \times \{0,1\}^* \times \mathbb{N}$
10 $X[1]\|X[2]\|\ldots\|X[m] \stackrel{n}{\leftarrow} X$
11 **if** $|X| \bmod n \neq 0$ **or** $X = \varepsilon$ **then** $w \leftarrow 1$, **else** $w \leftarrow 0$ (note: $w \leftarrow w(X)$)
12 **if** $t = 0$ (note: $m \geq 2$ holds for valid queries)
13 $Y[1] \leftarrow \mathsf{P}(2L \oplus X[1])$
14 **for** $i = 1$ **to** $m - 2$ **do** $Y[i+1] \leftarrow \mathsf{P}(Y[i] \oplus X[i+1])$
15 $V \leftarrow \mathsf{P}(Y[m-1] \oplus \mathsf{bp}(X[m]) \oplus 2^{w+1}L)$
15 $Y \leftarrow V \oplus U$
16 **if** $d = 0$ **return** Y
17 **else** $V^{\wedge} \leftarrow V \wedge \alpha$
18 **for** $j = 0$ **to** $d - 1$ **do** $S[j+1] \leftarrow \mathsf{P}(V^{\wedge} + j)$
19 **return** $Y\|S[1]S[2]\ldots S[d]$
20 **if** $t = 1$
21 **if** $|X| \leq n$ **then** $Y' \leftarrow \mathsf{P}(\mathsf{bp}(X) \oplus 4L \oplus 2^{w+1}L) \oplus U$; **return** Y'
22 **else** $Y'[1] \leftarrow \mathsf{P}(4L \oplus X[1])$
23 **for** $i = 1$ **to** $m - 2$ **do** $Y'[i+1] \leftarrow \mathsf{P}(Y'[i] \oplus X[i+1])$
24 $Y' \leftarrow \mathsf{P}(Y'[m-1] \oplus \mathsf{bp}(X[m]) \oplus 2^{w+1}L) \oplus U$
25 **return** Y'

Fig. 3. OMAC-extension using an n-bit URP, P.

[5] Here $F \equiv G$ means the equivalence of the output probability distribution functions, i.e. $\Pr[F(x_1) = y_1, \ldots, F(x_q) = y_q] = \Pr[G(x_1) = y_1, \ldots, G(x_q) = y_q])$ for any fixed possible $x_1, \ldots, x_q$ and $y_1, \ldots, y_q$. The probabilities are defined over F and G's randomness.

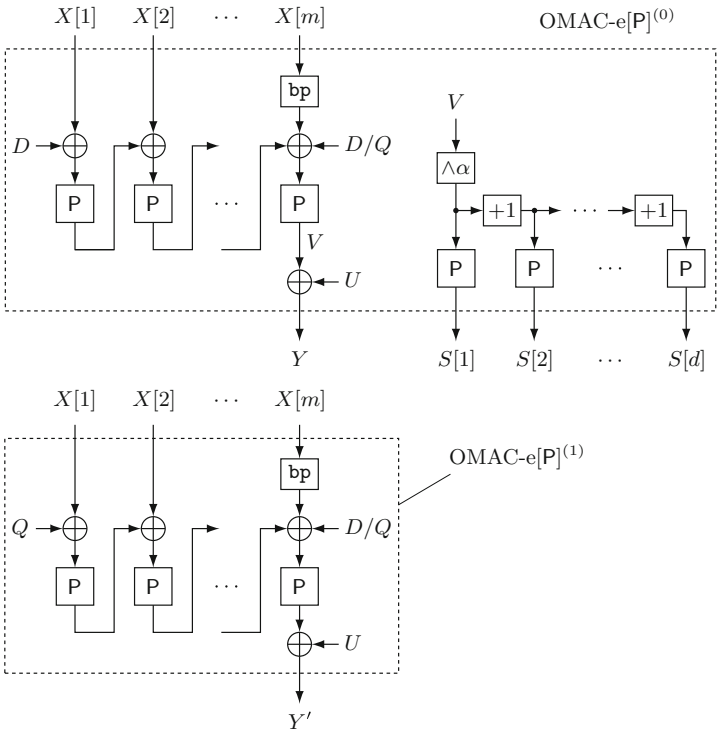

Fig. 4. Component functions of OMAC-extension. Here D and Q denote $2L$ and $4L$ with $L = \mathsf{P}(0^n)$, and U is uniformly random over n bits.

To further analyze OMAC-e[P], we introduce a set of ten functions, $\mathbf{Q} = \{\mathbf{Q}_i\}_{i=1,\ldots,10}$.

Definition 1. *Let* $\mathbf{Q}_i : \{0,1\}^n \to \{0,1\}^n$ *for* $i = 1,2,3,4,7,8,9$ *and let* $\mathbf{Q}_j : \{0,1\}^n \times \mathbb{N} \to (\{0,1\}^n)^{>0}$ *for* $j = 5,6$*, and let* $\mathbf{Q}_{10} : \{0,1\}^n \setminus \{0^n\} \to \{0,1\}^n$*. These functions are defined as*

$$\mathbf{Q}_1(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus x) \oplus \mathtt{Rnd}_1, \qquad \mathbf{Q}_2(x) \stackrel{\text{def}}{=} \mathsf{P}(4L \oplus x) \oplus \mathtt{Rnd}_2,$$

$$\mathbf{Q}_3(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_1 \oplus x) \oplus \mathtt{Rnd}_1, \qquad \mathbf{Q}_4(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_2 \oplus x) \oplus \mathtt{Rnd}_2,$$

$$\mathbf{Q}_5(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P},U}(\mathsf{P}(2L \oplus \mathtt{Rnd}_1 \oplus x), d), \qquad \mathbf{Q}_6(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P},U}(\mathsf{P}(4L \oplus \mathtt{Rnd}_1 \oplus x), d)$$

$$\mathbf{Q}_7(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus \mathtt{Rnd}_2 \oplus x) \oplus U, \qquad \mathbf{Q}_8(x) \stackrel{\text{def}}{=} \mathsf{P}(4L \oplus \mathtt{Rnd}_2 \oplus x) \oplus U,$$

$$\mathbf{Q}_9(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus 4L \oplus x) \oplus U, \qquad \mathbf{Q}_{10}(x) \stackrel{\text{def}}{=} \mathsf{P}(x) \oplus U,$$

where P *is an* n*-bit URP, and* $L = \mathsf{P}(0^n)$*, and* $\mathtt{Rnd}_1$ *and* $\mathtt{Rnd}_2$ *are independent* n*-bit random sequences, and* U *is another random* n*-bit value. Here,* $G_{\mathsf{P},U}(v,d)$ *is* $v \oplus U$ *if* $d = 0$ *and* $(v \oplus U \| \mathsf{P}(v \wedge \alpha) \| \mathsf{P}((v \wedge \alpha) + 1) \| \ldots \| \mathsf{P}((v \wedge \alpha) + (d-1)))$ *if* $d > 0$*. The sampling procedures for* $\mathsf{P}, \mathtt{Rnd}_1, \mathtt{Rnd}_2$*, and* U *are shared by all* $\mathbf{Q}_i$*s.*

We also treat $\mathbf{Q}$ as a tweakable function with tweak $t \in \{1, \ldots, 10\}$ by writing $\mathbf{Q}(t, x, d) = \mathbf{Q}_t(x, d)$ when $t \in \{5, 6\}$ and otherwise $\mathbf{Q}(t, x, d) = \mathbf{Q}_t(x)$. We can easily see that OMAC-e[P] can be simulated with black-box access to $\mathbf{Q}$, just the same as Q functions appeared in the proof of OMAC [9] that simulate OMAC.

We next define $\widetilde{\mathbf{Q}} = \{\widetilde{\mathbf{Q}}_i\}_{i=1,\ldots,10}$. For all $i = 1, \ldots, 10$, $\widetilde{\mathbf{Q}}_i$ is compatible with $\mathbf{Q}_i$.

Definition 2. *Let* $\mathsf{P}_1, \ldots, \mathsf{P}_4$ *be four independent* n*-bit URPs. Let* $\mathsf{R}_7, \ldots, \mathsf{R}_{10}$ *be four independent* n*-bit URFs, and let* R_5 *and* R_6 *be two independent URFs with* n*-bit input and* $(d_{\max} + 1)n$*-bit output. Using them we define*

$$\widetilde{\mathbf{Q}}_1(x) \stackrel{\text{def}}{=} \mathsf{P}_1(x), \qquad \widetilde{\mathbf{Q}}_2(x) \stackrel{\text{def}}{=} \mathsf{P}_2(x),$$
$$\widetilde{\mathbf{Q}}_3(x) \stackrel{\text{def}}{=} \mathsf{P}_3(x), \qquad \widetilde{\mathbf{Q}}_4(x) \stackrel{\text{def}}{=} \mathsf{P}_4(x),$$
$$\widetilde{\mathbf{Q}}_5(x, d) \stackrel{\text{def}}{=} \mathsf{R}_5^{d+1}(x), \qquad \widetilde{\mathbf{Q}}_6(x, d) \stackrel{\text{def}}{=} \mathsf{R}_6^{d+1}(x)$$
$$\widetilde{\mathbf{Q}}_7(x) \stackrel{\text{def}}{=} \mathsf{R}_7(x), \qquad \widetilde{\mathbf{Q}}_8(x) \stackrel{\text{def}}{=} \mathsf{R}_8(x),$$
$$\widetilde{\mathbf{Q}}_9(x) \stackrel{\text{def}}{=} \mathsf{R}_9(x), \qquad \widetilde{\mathbf{Q}}_{10}(x) \stackrel{\text{def}}{=} \mathsf{R}_{10}(x),$$

where $\mathsf{R}_i^{d+1}(x) = \mathrm{msb}_{n(d+1)}(\mathsf{R}_i(x))$ *for* $i = 5, 6$*. Here* $d_{\max}$ *is the maximum possible value of queried* d*, which will be determined by the underlying game and the adversary's parameter.*

We say a function compatible with $\mathbf{Q}$ is said to have $\mathbf{Q}$ profile. An adversary querying a function of $\mathbf{Q}$ profile is characterized by the number of queries, q, and the total sum of output n-bit blocks for $t \in \{5, 6\}$, σ_{out}. The next lemma shows the CPA-advantage in distinguishing $\mathbf{Q}$ and $\widetilde{\mathbf{Q}}$.

Lemma 1. *Let* $\mathcal{A}$ *be the adversary querying a function of* $\mathbf{Q}$ *profile with parameter list* (q, σ_{out})*. Then we have* $\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{A}) \le (3.5q^2 + 10\sigma_{\text{out}}q + 2.5\sigma_{\text{out}}^2)/2^n$.

The proof is given in the full-version.

Step 2: Modified CBC-MAC. For any n-bit (keyed) permutations, G and G', let $\mathrm{CBC}_{G,G'} : (\{0,1\}^n)^{>0} \to \{0,1\}^n$ be defined as

$$\mathrm{CBC}_{G,G'}(X[1]\|\ldots\|X[m]) = \begin{cases} G(X[1]) & \text{if m} = 1 \\ \mathrm{CBC}_{G'}(G(X[1])\|X[2]\|\ldots\|X[m]) & \text{if m} \ \ge 2, \end{cases}$$

where $\mathrm{CBC}_{G'}$ is the standard CBC-MAC using G'. We then define a function compatible with OMAC-e[P], denoted by $\mathbb{CBC}$. For any $X \in \{0,1\}^*$, let $w(X) = 1$ if $|X|$ mod n $\neq 0$ or $X = \varepsilon$ and otherwise $w(X) = 0$. For $|X| > n$, $\mathbb{CBC}^{(0)}(X, d)$ is computed as follows.

1. $X[1]\|X[2]\|\ldots\|X[m] \xleftarrow{n} X$ and $w \leftarrow w(X)$
2. $Z \leftarrow \mathrm{CBC}_{\mathsf{P}_1,\mathsf{P}_3}(X[1]\|\ldots\|X[m-1])$
3. Output $Y\|S[1]\|\ldots\|S[d] \leftarrow \mathsf{R}_{5+w}^{d+1}(Z \oplus \mathtt{bp}(X[m]))$

Here, if $d = 0$ the output is Y. Similarly, for $X \in \{0,1\}^*$, $\mathbb{CBC}^{(1)}(X)$ is computed as follows.

1. $X[1]\|X[2]\|\ldots\|X[m] \xleftarrow{n} X$ and $w \leftarrow w(X)$
2. If $|X| \le n$ output $Y' \leftarrow \mathsf{R}_{9+w}(\mathtt{bp}(X))$,
3. Otherwise $Z' \leftarrow \mathrm{CBC}_{\mathsf{P}_2,\mathsf{P}_4}(X[1]\|\ldots\|X[m-1])$, and output $Y' \leftarrow \mathsf{R}_{7+w}(Z' \oplus \mathtt{bp}(X[m]))$.

The pseudo-code of $\mathbb{CBC}$ (combining $\mathbb{CBC}^{(0)}$ and $\mathbb{CBC}^{(1)}$) is presented in Fig. 5. Here, $\mathsf{R}^i_j(X)$ for $j = 5, 6$ denotes $\mathrm{msb}_{ni}(\mathsf{R}_j(X))$. One can simulate OMAC-e[P] via black-box accesses to $\mathbf{Q}$, including the final mask by U. For example, to simulate OMAC-e[P]$(0, N, 2)$ for $|N| = 3n$, we first perform a partition, $N[1]\|N[2]\|N[3] \xleftarrow{n} N$, and then proceed as (1) $Y[1] \leftarrow \mathbf{Q}_1(N[1])$, (2) $Y[2] \leftarrow \mathbf{Q}_3(N[2] \oplus Y[1])$, and (3) $Y[3]\|S[1]S[2] \leftarrow \mathbf{Q}_5(N[3] \oplus Y[2])$. If $|N[3]| = n-2$ then $\mathbf{Q}_5(N[3] \oplus Y[2])$ is replaced with $\mathbf{Q}_6(N[3]\|10 \oplus Y[2])$. For more examples, OMAC-e[P]$(1, C, 0)$ for $|C| = n$ can be simulated via calling $\mathbf{Q}_9(C)$. For $|C| < n$, OMAC-e[P]$(1, C, 0)$ can be simulated via calling $\mathbf{Q}_{10}(\mathtt{bp}(C)) = \mathbf{Q}_{10}(C\|10\ldots0)$. Formally, we have the following proposition.

Proposition 2. *There exists a procedure $h(\cdot)$ that uses $\mathbf{Q}$ as a black box and perfectly simulates OMAC-e[P], i.e. $h(\mathbf{Q}) \equiv$ OMAC-e[P]. Moreover, we have $h(\widetilde{\mathbf{Q}}) \equiv \mathbb{CBC}$ for this $h(\cdot)$.*

Let $\mathbb{RND}^{(0)}$ and $\mathbb{RND}^{(1)}$ be the independent random functions compatible with OMAC-e[P]$^{(0)}$ and OMAC-e[P]$^{(1)}$. Here, $\mathbb{RND}^{(0)}$ takes $(N, d) \in \{0,1\}^{>n} \times \mathbb{N}$ and samples $Y \xleftarrow{\$} (\{0,1\}^n)^{d_{\max}+1}$ if N is new, and outputs $\mathrm{msb}_{n(d+1)}(Y)$, where $d_{\max}$ is the same as $\mathbb{CBC}$. Similarly $\mathbb{RND}^{(1)}$ takes $C \in \{0,1\}^*$ and outputs $Y' \xleftarrow{\$} \{0,1\}^n$ if C is new. We define $\mathbb{RND}$ as a function consisting of $\mathbb{RND}^{(0)}$ and $\mathbb{RND}^{(1)}$ and taking $t = 0, 1$ as a tweak. Then, we have the following lemma. The proof is given in the full-version.

Lemma 2. *Let $\mathcal{A}$ be an adversary querying a function of OMAC-e profile with parameter list $(q, \sigma_{in}, \sigma_{out})$. Then, $\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{A}) \le 2\sigma_{\mathrm{in}}^2/2^n$.*

Step 3: Derivation of PRIV Bound. Combining the above lemmas and propositions, our PRIV bound is derived. Let $\mathcal{A}$ be the CPA-adversary against AE with parameter list (q, σ_N, σ_M). Then there exist adversary $\mathcal{B}$ querying to a function of OMAC-e profile with $2q$ queries, $\sigma_{\mathrm{in}} = \sigma_N + \sigma_M$ input blocks, and $\sigma_{\mathrm{out}} = \sigma_M + 2q$ output blocks, and adversary $\mathcal{C}$ querying to a set of ten functions with $\mathbf{Q}$ profile, using $\sigma_N + \sigma_M$ queries and $\sigma_M + q$ output n-bit blocks for queries with $t = 5, 6$, such that

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}'[\mathrm{Perm}(n)]}(\mathcal{A}) = \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\mathrm{EAX}'\text{-}\mathcal{E}_{\mathsf{P}},\$}(\mathcal{A}) = \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\text{OMAC-e[P]}),\$}(\mathcal{A}) \tag{7}$$

$$\le \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\text{OMAC-e[P]}),f_e(\mathbb{CBC})}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\mathbb{CBC}),f_e(\mathbb{RND})}(\mathcal{A}) + \underbrace{\mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\mathbb{RND}),\$}(\mathcal{A})}_{=0} \tag{8}$$

$$\le \mathtt{Adv}^{\mathtt{cpa}}_{\text{OMAC-e[P]},\mathbb{CBC}}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) \tag{9}$$

$$= \mathbf{Adv}^{\mathbf{cpa}}_{h(\mathbf{Q}),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) \tag{10}$$

$$\leq \mathbf{Adv}^{\mathbf{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{2(\sigma_N+\sigma_M)^2}{2^n} \tag{11}$$

$$\leq \frac{3.5(\sigma_N+\sigma_M)^2 + 10(\sigma_M+q)(\sigma_N+\sigma_M) + 2.5(\sigma_M+q)^2}{2^n} + \frac{2(\sigma_N+\sigma_M)^2}{2^n} \tag{12}$$

$$\leq \frac{18(\sigma_N+\sigma_M)^2}{2^n} = \frac{18\sigma^2_{\mathrm{priv}}}{2^n}, \tag{13}$$

as $q \leq \sigma_N$. Here, the second equality in Eq. (7) follows from Proposition 1, Eq. (10) follows from Proposition 2, Eq. (11) follows from Lemma 2, and Eq. (12) follows from Lemma 1. In addition, $\mathbf{Adv}^{\mathbf{cpa\text{-}nr}}_{f_e(\mathbb{RND}),}(\mathcal{A}) = 0$ holds because when $\mathcal{A}$ queries (N, M) to $f_e(\mathbb{RND})$ the output is a subsequence of $\mathbb{RND}^{(0)}(N, |M|_n)$ with the first n bits XORed by the output of $\mathbb{RND}^{(1)}$ (whose input is a part of $\mathbb{RND}^{(0)}(N, |M|_n)$). As N is always fresh, the output is always random. This concludes the proof of Theorem 1.

Step 4: Derivation of AUTH Bound. The AUTH bound is derived in a similar way. Let $\mathbb{EAX}'$ be the AE algorithm compatible with EAX′[Perm(n)] using $f_e(\mathbb{RND})$ and $f_d(\mathbb{RND})$ for the encryption and decryption algorithms. We let $\mathcal{A}$ be the CCA-adversary against AE with parameter list $(q, q_v, \sigma_N, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{C}})$. Then we have the following bound.

$$\mathbf{Adv}^{\mathbf{auth}}_{\mathbb{EAX}'}(\mathcal{A}) \leq q_v/2^\tau. \tag{14}$$

The proof of Eq. (14) is given in the full-version. Then, there exist adversary $\mathcal{B}$ querying to a function of OMAC-e profile with $2(q+q_v)$ queries with $\sigma_{\mathrm{in}} = \sigma_N + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{C}}$ and $\sigma_{\mathrm{out}} = \sigma_M + 2q + \sigma_{\widetilde{C}} + 2q_v$, and adversary $\mathcal{C}$ querying to a function of $\mathbf{Q}$ profile with $\sigma_N + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{C}}$ queries and $\sigma_M + q + \sigma_{\widetilde{C}} + q_v$ output blocks for queries with $t = 5, 6$, such that

$$\mathbf{Adv}^{\mathbf{auth}}_{\mathrm{EAX}'[\mathrm{Perm}(n)]}(\mathcal{A})$$

$$\leq \mathbf{Adv}^{\mathbf{cca\text{-}nr}}_{(\mathrm{EAX}'\text{-}\mathcal{E}_\mathsf{P},\mathrm{EAX}'\text{-}\mathcal{D}_\mathsf{P}),(f_e(\mathbb{RND}),f_d(\mathbb{RND}))}(\mathcal{A}) + \mathbf{Adv}^{\mathbf{auth}}_{\mathbb{EAX}'}(\mathcal{A}) \tag{15}$$

$$\leq \mathbf{Adv}^{\mathbf{cca\text{-}nr}}_{(f_e(\mathrm{OMAC\text{-}e[P]}),f_d(\mathrm{OMAC\text{-}e[P]})),(f_e(\mathbb{RND}),f_d(\mathbb{RND}))}(\mathcal{A}) + \frac{q_v}{2^\tau} \tag{16}$$

$$\leq \mathbf{Adv}^{\mathbf{cpa}}_{\mathrm{OMAC\text{-}e[P]},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \tag{17}$$

$$\leq \mathbf{Adv}^{\mathbf{cpa}}_{\mathrm{OMAC\text{-}e[P]},\mathbb{CBC}}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \tag{18}$$

$$= \mathbf{Adv}^{\mathbf{cpa}}_{h(\mathbf{Q}),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \tag{19}$$

$$\leq \mathbf{Adv}^{\mathbf{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{2(\sigma_N+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{C}})^2}{2^n} + \frac{q_v}{2^\tau} \tag{20}$$

$$\leq \frac{3.5(\sigma_N+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{C}})^2+10(\sigma_M+q+\sigma_{\widetilde{C}}+q_v)(\sigma_N+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{C}})}{2^n}$$

$$+\frac{2.5(\sigma_M+q+\sigma_{\widetilde{C}}+q_v)^2}{2^n}+\frac{2(\sigma_N+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{C}})^2}{2^n}+\frac{q_v}{2^\tau} \quad (21)$$

$$\leq \frac{18\sigma_{\mathrm{auth}}^2}{2^n}+\frac{q_v}{2^\tau}, \quad (22)$$

since $q \leq \sigma_N$ and $q_v \leq \sigma_{\widetilde{N}}$. Here, Eq. (16) follows from Proposition 1 and Eqs. (14), (19) follows from Proposition 2, Eq. (20) follows from Lemma 2, and Eq. (21) follows from Lemma 1. This concludes the proof of Theorem 2.

7 Fixing the Flaw

There would be ways to fix the flaw of EAX′ to make it as a secure general-purpose AE accepting cleartexts of any length. Below, we provide some of them, naming it to EAX″. The concept here is not to touch the inside of EAX′, instead using it as a black box. We only propose the fixes for encryption, as the corresponding decryptions are fairly straightforward.

Method 1: EAX″$_1$-$\mathcal{E}_K(N,M) \stackrel{\mathrm{def}}{=}$ EAX′-$\mathcal{E}_K(0^n\|N,M)$.

Algorithm $\mathbb{CBC}$ (given $d_{\max}$):
Initialization
00 **for** $i=1$ **to** 4 **do** $\mathsf{P}_i \xleftarrow{\$} \mathrm{Perm}(n)$
01 $\mathsf{R}_5 \xleftarrow{\$} \mathrm{Func}(n,d_{\max})$, $\mathsf{R}_6 \xleftarrow{\$} \mathrm{Func}(n,d_{\max})$
02 **for** $j=7$ **to** 10 **do** $\mathsf{R}_j \xleftarrow{\$} \mathrm{Func}(n)$ (note: R_{10}'s actual input is in $\{0,1\}^n \setminus \{0^n\}$)
On query $(t,X,d) \in \{0,1\} \times \{0,1\}^* \times \mathbb{N}$
10 $X[1]\|X[2]\|\ldots\|X[m] \xleftarrow{n} X$
11 **if** $|X| \bmod n \neq 0$ **or** $X=\varepsilon$ **then** $w \leftarrow 1$, **else** $w \leftarrow 0$ (note: $w \leftarrow w(X)$)
12 **if** $t=0$ (note: $m \geq 2$ holds for valid queries)
13 $Y[1] \leftarrow \mathsf{P}_1(X[1])$
14 **for** $i=1$ **to** $m-2$ **do** $Y[i+1] \leftarrow \mathsf{P}_3(Y[i] \oplus X[i+1])$
15 **if** $d=0$ **then** $Y \leftarrow \mathsf{R}^1_{5+w}(Y[m-1] \oplus \mathsf{bp}(X[m]))$; **return** Y
16 **else** $Y\|S[1]\|S[2]\|\ldots\|S[d] \leftarrow \mathsf{R}^{d+1}_{5+w}(Y[m-1] \oplus \mathsf{bp}(X[m]))$
17 **return** $Y\|S[1]\|S[2]\|\ldots\|S[d]$
18 **if** $t=1$
19 **if** $|X| \leq n$ **then** $Y' \leftarrow \mathsf{R}_{9+w}(\mathsf{bp}(X))$; **return** Y'
20 **else** $Y'[1] \leftarrow \mathsf{P}_2(X[1])$
21 **for** $i=1$ **to** $m-2$ **do** $Y'[i+1] \leftarrow \mathsf{P}_4(Y'[i] \oplus X[i+1])$
22 $Y' \leftarrow \mathsf{R}_{7+w}(Y'[m-1] \oplus \mathsf{bp}(X[m]))$
23 **return** Y'

Fig. 5. $\mathbb{CBC}$ using four n-bit URPs, four n-bit URFs, and two n-bit input, $(d_{\max}+1)n$-bit output URFs.

Method 2: Use two keys for E, K and K', and let

$$\mathrm{EAX}''_2\text{-}\mathcal{E}_{K,K'}(N,M) \stackrel{\text{def}}{=} \begin{cases} \mathrm{EAX}'\text{-}\mathcal{E}_K(N,M) & \text{if } |\mathrm{N}| > \mathrm{n}, \\ \mathrm{EAX}'\text{-}\mathcal{E}_{K'}(0^n \| N, M) & \text{if } |\mathrm{N}| \leq \mathrm{n}, \end{cases}$$

where K and K' are independent or $K' = K \oplus \mathtt{cst}$ for a non-zero constant $\mathtt{cst}$. The choice of $\mathtt{cst}$ must be done with care to avoid related-key attacks. For instance, letting $\mathtt{cst} = 1^{|K|}$ seems natural while this is problematic with DES due to the complementary property of the key schedule. One option is to use a random-looking constant, say the first few digits of π.

Method 3: Use a key for E, K, and an independent n-bit key, L, and let

$$\mathrm{EAX}''_3\text{-}\mathcal{E}_{K,L}(N,M) \stackrel{\text{def}}{=} \begin{cases} \mathrm{EAX}'\text{-}\mathcal{E}_K(N,M) & \text{if } |\mathrm{N}| > \mathrm{n}, \\ \mathrm{EAX}'\text{-}\mathcal{E}^{\oplus}_{K,L}(0^n \| N, M) & \text{if } |\mathrm{N}| \leq \mathrm{n}, \end{cases}$$

where $\mathrm{EAX}'\text{-}\mathcal{E}^{\oplus}_{K,L}$ is EAX′ encryption with blockcipher $\widetilde{E}_{K,L}$ defined as $\widetilde{E}_{K,L}(X) = E_K(X \oplus L)$.

The security bounds of the above methods are easily derived from the results of Theorems 1 and 2. For the latter option of Method 2 we also need a very restricted form of related-key security of E, and for Method 3 we need the theory of tweakable blockcipher [11]. Each method has its own pros and cons: Method 1 is the simplest but needs additional blockcipher calls irrespective of $|N|$. Methods 2 and 3 keep the original operation for $|N| > n$, but need additional key or a stronger security requirement on E. We also warn that Method 3 allows a partial key recovery attack with birthday complexity.

8 Concluding Remarks

Practical Implications. Attacks as those described in the current paper are often turned down by non-cryptographers as "only theoretical" or "don't apply in practice". Indeed, none of our attacks is applicable if the cleartext size exceeds n bits. But even if ANSI C12.22 prohibited any cleartexts of size $n = 128$ bits or shorter, including EAX′ in the standard would be like an unexploded bomb – waiting to go off any time in the future. Remember that EAX′ is intended for Smart Grid, i.e., for the use in dedicated industrial systems such as electrical meters, controllers and appliances. It hardly seems reasonable to assume that *every* device will *always* carefully check cleartexts and plaintexts for validity and plausibility. Also, vendors may be tempted to implement their own nonstandard extensions avoiding "unnecessarily long" texts.

For a non-cryptographer, assuming a "decryption oracle" may seem strange – if there were such an oracle, why bother with message recovery attacks at all? However, experience shows that such theoretical attacks are often practically exploitable. For example, some error messages return the input that caused the error: "Syntax error in 'xyzgarble'." Even if the error message does not transmit

the entire fake plaintext, any error message telling the attacker whether the fake message followed some syntactic conventions or not is potentially useful for the attacker. See [8] for an early example.

Also note that our forgery attacks allow a malicious attacker to create a large number of messages with given single-block cleartexts and random single-block plaintexts, that appear to come from a trusted source, because the authentication succeeded. What the actual devices will do when presented with apparently valid random commands is a source of great speculation.

Our Recommendation. Whenever possible, avoid adopting EAX′ in new applications. If EAX′ cannot be avoided, then this has to be carefully implemented to exclude one-block cleartexts. We note that specifying the minimum data length in standard documents does not necessarily prevent the adversary from using short cleartexts. Therefore, the cleartext length checking mechanisms are needed at both ends of encryption and decryption. Instead, one can safely use EAX″ which allows the re-use of EAX′ implementations. Other provably secure authenticated encryptions, including the original EAX, are also safe options.

Acknowledgments. The authors thank the anonymous FSE 2013 reviewers for helpful comments. This paper is based on the collaboration started at Dagstuhl Seminar 12031, Symmetric Cryptography. We thank participants of the seminar for useful comments, and discussions with Greg Rose were invaluable for writing Sect. 8. We also thank Mihir Bellare and Jeffrey Walton for feedback. The work by Tetsu Iwata was supported by MEXT KAKENHI, Grant-in-Aid for Young Scientists (A), 22680001.

References

1. Comment for EAX Cipher Mode (by Toshiba Corporation). http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/EAX%27/Toshiba_Report2NIST_rev051.pdf
2. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B (2005)
3. American National Standard Protocol Specification For Interfacing to Data Communication Networks. ANSI C12.22-2008 (2008)
4. Measurement Canada, Specification for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables. MC1222, 2009 (2009)
5. ANSI C12.22, IEEE 1703, and MC12.22 Transport Over IP. RFC 6142 (2011)
6. IEEE Standard for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables. IEEE 1703–2012 (2012)
7. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: Roy, Meier: [16], pp. 389–407
8. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
9. Iwata, T., Kurosawa, K.: OMAC: one-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)

10. Iwata, T., Kurosawa, K.: Stronger security bounds for OMAC, TMAC, and XCBC. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 402–415. Springer, Heidelberg (2003)
11. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. J. Cryptology **24**(3), 588–613 (2011)
12. Minematsu, K., Lucks, S., Morita, H., Iwata, T.: Cryptanalysis of EAX-Prime. DIAC - Directions in Authenticated Ciphers (2012). http://hyperelliptic.org/DIAC/
13. Moise, A., Beroset, E., Phinney, T., Burns, M.: EAX' cipher mode. NIST Submission, May 2011. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax-prime/eax-prime-spec.pdf
14. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, Meier: [16], pp. 348–359
15. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
16. Roy, B., Meier, W. (eds.): FSE 2004. LNCS, vol. 3017. Springer, Heidelberg (2004)