# Improved Authenticity Bound of EAX, and Refinements

**(full version of a paper presented at ProvSec 2013 [13])**

Kazuhiko Minematsu[1], Stefan Lucks[2], and Tetsu Iwata[3]

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

**Abstract.** EAX is a mode of operation for blockciphers to implement an authenticated encryption. The original paper of EAX proved that EAX is unforgeable up to $O(2^{n/2})$ data with one verification query. However, this generally guarantees a rather weak bound for the unforgeability under multiple verification queries, i.e., only $(2^{n/3})$ data is acceptable.
This paper provides an improvement over the previous security proof, by showing that EAX is unforgeable up to $O(2^{n/2})$ data with multiple verification queries. Our security proof is based on the techniques appeared in a paper of FSE 2013 by Minematsu et al. which studied the security of a variant of EAX called EAX-prime. We also provide some ideas to reduce the complexity of EAX while keeping our new security bound. In particular, EAX needs three blockcipher calls and keep them in memory as a pre-processing, and our proposals can effectively reduce three calls to one call. This would be useful when computational power and memory are constrained.

**Keywords:** Authenticated encryption, EAX, security bound.

## 1 Introduction

EAX [5] is a mode of operation for blockciphers proposed by Bellare, Rogaway and Wagner at FSE 1994. It implements an authenticated encryption with associated data, AEAD for short. EAX has been standardized by ISO/IEC [2] and included in some popular software libraries [1,8,9]. In FSE 2013, Minematsu, Lucks, Morita, and Iwata [14] investigated a variant of EAX defined by ANSI C12.22, called EAX-prime. They showed that EAX-prime is totally broken if the 'cleartext' part of the input is as short as a single block or shorter. At the same time, the authors proved EAX-prime is secure if cleartexts are required to be longer than a single block.

In this paper, we study the implications of [14] to the original EAX. Though the original EAX has already been proved to be secure, the security bound provided by [5], in particular the authenticity bound, does not show the standard birthday-type security when the adversary is allowed to make multiple verification queries. More formally, the original bound is $O(\sigma^2/2^n + 1/2^\tau)$ where $\sigma$ denotes the total input blocks, $n$ denotes the block size, and $\tau$ denotes the tag length, *if the number of verification queries is one*. From the well-known result of [3], this bound generally implies $O(q_v\sigma^2/2^n + q_v/2^\tau)$ when the number of verification queries is $q_v \geq 1$, hence the provable security is degraded, roughly from $2^{n/2}$ to $2^{n/3}$, assuming $q_v \approx \sigma$. We note that, since many systems in practice do accept multiple verification queries, the analysis for this case is relevant. Based on the idea of [14], we provide an improved authenticity bound for EAX, namely $O(\sigma^2/2^n + q_v/2^\tau)$, hence the security up to $2^{n/2}$ data. When $n = 128$, this means that the provable security is improved from 43 bits to 64 bits. In addition, we prove our new bound in a slight more general setting than the original specification, in the sense that the empty header is acceptable, which is plausible in practice.

We note that the technical difficulty in handling the multiple verification queries comes from the fact that the reject symbol returned from the decryption oracle may leak some information about the secret key, and hence this may have impact on the choice of the subsequent encryption and decryption queries. Furthermore, nonces used in encryption queries can be reused for

decryption queries, or vice versa. In this case, we may not have "fresh" randomness in order to show that the success probability of the last decryption query is small.

We also provide ideas to reduce the computation overhead of EAX, which we assume the primal goal of EAX-prime. In EAX, three blockcipher calls are required in advance to the actual processing, and this may make it less attractive to constrained devices. In our proposals, the overhead is reduced to one blockcipher call while keeping the security bound that we proved for the original EAX. This also achieves a more memory-efficient, faster operation than the original. In this respect our proposal can be seen as a provably-secure alternative to EAX-prime having no input-length restriction.

The main technical point in our proposals is the generation of five mask values, originally generated from three blockcipher calls. We propose three mask-generation methods, where the first one is based on the constant Galois field multiplication similar to [15], and the second and third ones are based on the word permutation and XOR. The underlying problem has a relationship to word-oriented LFSR [19] discussed by Chakraborty and Sarkar [7] and by Krovetz and Rogaway [12].

## 2 Preliminaries

**Notations.** For a binary string $X$, $|X|$ denotes the bit length of $X$. For a positive integer $n$ we define $|X|_n \stackrel{\text{def}}{=} \max\{1, \lceil |X|/n \rceil\}$. The first $s$ bits of $X$ for $|X| \geq s$ is written as $\mathrm{msb}_s(X)$. Let $\varepsilon$ denote the empty string, which is a binary string of length 0. Thus we have $|\varepsilon| = 0$ and $|\varepsilon|_n = 1$. The set of all finite-length binary strings, including $\varepsilon$, is denoted by $\{0,1\}^*$. Let $\mathbb{N} = \{0, 1, \dots\}$. If $X \in \mathcal{X}$ is uniformly chosen from $\mathcal{X}$ we write $X \xleftarrow{\$} \mathcal{X}$. For $X, Y \in \{0,1\}^*$, their concatenation is denoted by $X\|Y$ or $XY$. A sequence of $a$ zeros (ones) is denoted by $0^a$ ($1^a$). Following [5], let $[i]_n$ denote a standard $n$-bit encoding of integer $i \geq 0$, e.g., $[2]_n$ denotes $0^{n-2}10$. Let $(\{0,1\}^n)^{>0}$ denote the set of strings of length $n, 2n, \dots$. We also define $\{0,1\}^{\geq k}$ and $(\{0,1\}^n)^{\geq k}$ analogously. For $X, Y \in \{0,1\}^n$, $X + Y$ or $X - Y$ is defined as an addition or a subtraction modulo $2^n$.

For $X \in \{0,1\}^*$, let $X[1]\|X[2]\|\dots\|X[m] \xleftarrow{n} X$ denote the partition into $n$-bit blocks, i.e., we have $m = |X|_n$ 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 the shorter variable into the end of the longer one: i.e. $X \oplus_{\text{end}} Y = (0^{|Y|-|X|}\|X) \oplus Y$ if $|Y| \geq |X|$ and otherwise $X \oplus_{\text{end}} Y = X \oplus (0^{|X|-|Y|}\|Y)$.

**Random Function.** The set of all functions $\{0,1\}^n \to \{0,1\}^m$ is denoted by $\mathrm{Func}(n,m)$. We will write $\mathrm{Func}(n)$ to mean $\mathrm{Func}(n,n)$. The set of all permutations over $\{0,1\}^n$ is denoted by $\mathrm{Perm}(n)$. Following [14], we define a uniform random function (URF) as a random function uniformly distributed over $\mathrm{Func}(n,m)$ for some $n$ and $m$. A URF is denoted by $\mathsf{R}$, assuming $n$ and $m$ are clear from the context. In a similar manner we define a uniform random permutation (URP) as a random permutation uniformly distributed over $\mathrm{Perm}(n)$ for some $n$. A URP is denoted by $\mathsf{P}$.

**Field with $2^n$ Points.** We may view $X \in \{0,1\}^n$ as a coefficient vector of the polynomial of $\mathrm{GF}(2^n)$, yielding a one-to-one mapping. By writing $2X$ we mean the multiplication of the generator of $\mathrm{GF}(2^n)$ and $X$ over $\mathrm{GF}(2^n)$. Here, $2(2L)$ is denoted by $4L$ or $2^2L$. The operation $2X$ is called *doubling*, and is efficiently implemented by one-bit shift with constant XOR, see e.g. [10].

## 3 Provable Security of EAX

### 3.1 Specification of EAX

We first define the authenticated encryption, AE in short (or more formally, AE with associated data (AEAD)). The encryption function of an AE scheme accepts the nonce $N$, the header (also

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

1. $\underline{N} \leftarrow \mathrm{CMAC}_K^{(0)}(N)$
2. $\underline{H} \leftarrow \mathrm{CMAC}_K^{(1)}(H)$
3. $C \leftarrow \mathrm{CTR}_K(\underline{N}, M)$
4. $\underline{T} \leftarrow \underline{N} \oplus \underline{H} \oplus \mathrm{CMAC}_K^{(2)}(C)$
5. $T \leftarrow \mathrm{msb}_\tau(\underline{T})$
6. **return** $(C, T)$

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

1. $\underline{N} \leftarrow \mathrm{CMAC}_K^{(0)}(N)$
2. $\underline{H} \leftarrow \mathrm{CMAC}_K^{(1)}(H)$
3. $\underline{T} \leftarrow \underline{N} \oplus \underline{H} \oplus \mathrm{CMAC}_K^{(2)}(C)$
4. $\widehat{T} \leftarrow \mathrm{msb}_\tau(\underline{T})$
5. **if** $\widehat{T} \neq T$ **return** $\perp$
6. **else** $M \leftarrow \mathrm{CTR}_K(\underline{N}, C)$
7. **return** $M$

**Algorithm** $\mathrm{CMAC}_K^{(i)}(M)$ (for $i \in \{0, 1, 2\}$)

1. $L \leftarrow E_K([0]_n)$, $L' \leftarrow E_K([1]_n)$, $L'' \leftarrow E_K([2]_n)$
2. $D \leftarrow 2L$, $Q \leftarrow 4L$
3. **if** $M = \varepsilon$ **return** $E_K([i]_n \oplus D)$
4. **else**
5. **if** $i = 0$ **return** $\mathrm{CBC}_K(L, \mathtt{pad}(M; D, Q))$
6. **if** $i = 1$ **return** $\mathrm{CBC}_K(L', \mathtt{pad}(M; D, Q))$
7. **if** $i = 2$ **return** $\mathrm{CBC}_K(L'', \mathtt{pad}(M; D, Q))$

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

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

**Algorithm** $\mathrm{CBC}_K(I, M)$

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

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

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

**Fig. 1.** (Upper) The encryption and decryption algorithms of $\mathrm{EAX}[E, \tau]$. Here $H$ and $M$ can be the empty string, $\varepsilon$, while $H \neq \varepsilon$ was originally required in [5]. (Lower) Component algorithms of $\mathrm{EAX}[E, \tau]$. For $\mathrm{CBC}_K$, $|M| \in \{n, 2n, \dots\}$.

called associated data) $H$, and the plaintext $M$ and generates the ciphertext $C$ and the tag $T$. The decryption (verification) function accepts $N$, $H$, $C$, and $T$, and generates the decrypted plaintext $M$ if $(N, C, T)$ is valid, or the flag $\perp$ if invalid.

The specification EAX is shown in Fig. 1. EAX is based on an $n$-bit blockcipher, $E$, where the key of $E$ is written as $K$. EAX taking a blockcipher $E$ and using the $\tau$-bit tag for $\tau \leq n$ is denoted by $\mathrm{EAX}[E, \tau]$. The encryption and decryption functions are written as EAX-$\mathcal{E}_{K,\tau}$ and EAX-$\mathcal{D}_{K,\tau}$, or EAX-$\mathcal{E}_K$ and EAX-$\mathcal{D}_K$ if $\tau$ is clear from the context.

In $\mathrm{EAX}[E, \tau]$, we assume that $N, H, M \in \{0,1\}^*$ with $N \neq \varepsilon$. In the original specification, $N$ and $H$ are assumed to be non-empty (see Section 6 of [4]). However, this paper slightly generalizes the setting, allowing $H$ to be empty[4]. The plaintext $M$ can be empty, and in that case the corresponding $C$ is also empty. The ciphertext $C$ has the same length as the corresponding plaintext, $M$, and the tag $T$ is $\tau$ bits.

In [5] the definition of $\mathrm{CMAC}_K^{(i)}(M)$ is simpler than ours, i.e. it is defined as $\mathrm{CMAC}_K^{(i)}(M) \stackrel{\mathrm{def}}{=} \mathrm{CMAC}_K([i]_n \| M)$. Here, $\mathrm{CMAC}_K(M)$ denotes the original CMAC defined as $\mathrm{CMAC}_K(M) = \mathrm{CBC}_K(\mathtt{pad}(M; D, Q))$. Our definition is equivalent and we employ it to emphasize the three redundant $E_K$ calls, $L$, $L'$, and $L''$, and make explicit the computation of $\mathrm{CMAC}_K^{(i)}(\varepsilon)$ with them.

### 3.2 Security Notions

The security of AE can be defined by two notions, privacy and authenticity [5,16]. In defining them, let $\mathcal{A}^{O_1, O_2, \dots, O_c}$ denote the adversary $\mathcal{A}$ accessing $c$ oracles, $O_1, \dots, O_c$, in an arbitrarily

[4] This setting allows $(N, H, M)$ with $H = \varepsilon$ and $M = \varepsilon$ as a valid, though artificial, input to the encryption function.

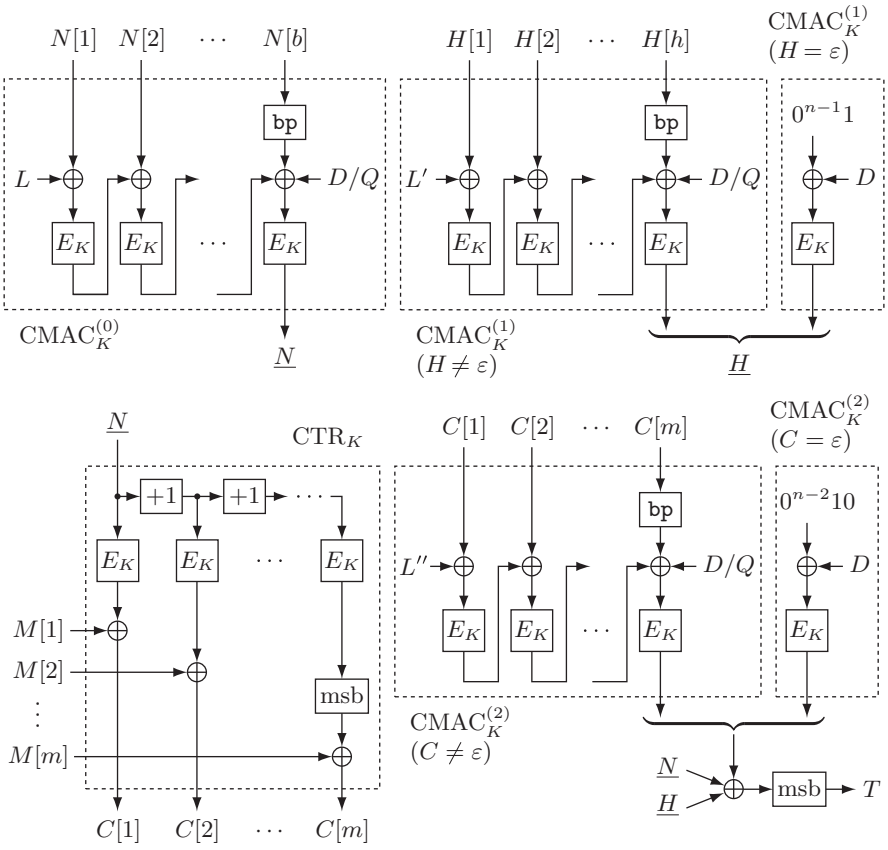


**Fig. 2.** The encryption algorithm of EAX. The specification is extended to accept $H = \varepsilon$. In the figure, $\mathtt{bp}(x) = x$ if $|x| = n$ and $\mathtt{bp}(x) = x\|10^{n-1-(|x| \bmod n)}$ if $|x| < n$.

order. If $O_i$ and $O_j$ are oracles having the same input and output domains, we say they are compatible. Let $\mathsf{AE}[\tau]$ be an AE compatible with EAX having $\tau$-bit tag. The encryption and decryption algorithms are $\mathsf{AE}\text{-}\mathcal{E}_\tau$ and $\mathsf{AE}\text{-}\mathcal{D}_\tau$. If $\mathcal{A}$ is a CPA-adversary against $\mathsf{AE}[\tau]$, it accesses $\mathsf{AE}\text{-}\mathcal{E}_\tau$. The encryption queries made by $\mathcal{A}$ are written as $(N_1, H_1, M_1), \ldots, (N_q, H_q, M_q)$, where the number of queries, $q$, is a parameter of $\mathcal{A}$. We also consider $\sigma_{\mathsf{X}} \stackrel{\text{def}}{=} \sum_{i=1}^{q} |\mathsf{X}_i|_n$ for $\mathsf{X} \in \{N, H, M\}$, and assume a parameter list $(q, \sigma_N, \sigma_H, \sigma_M)$ to define the resource of $\mathcal{A}$.

Let \$ denote the random-bit oracle, which takes $(N, H, M)$ and returns $(C, T) \stackrel{\$}{\leftarrow} \{0,1\}^{|M|} \times \{0,1\}^\tau$. Then the privacy of $\mathsf{AE}$ for CPA-adversary $\mathcal{A}$ is defined as

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathsf{AE}\text{-}\mathcal{E}_\tau} \Rightarrow 1] - \Pr[\mathcal{A}^{\$} \Rightarrow 1].$$

Here, $\mathcal{A}$ is nonce-respecting, i.e., all $N_i$s chosen by $\mathcal{A}$ are distinct.

To define the authenticity, we assume a CCA-adversary $\mathcal{A}$ against $\mathsf{AE}[\tau]$. It accesses $\mathsf{AE}\text{-}\mathcal{E}_\tau$ and $\mathsf{AE}\text{-}\mathcal{D}_\tau$. The set of encryption queries is denoted by $(N_1, H_1, M_1), \ldots, (N_q, H_q, M_q)$, and the set of decryption queries is denoted by $(\widetilde{N}_1, \widetilde{H}_1, \widetilde{C}_1, \widetilde{T}_1), \ldots, (\widetilde{N}_{q_v}, \widetilde{H}_{q_v}, \widetilde{C}_{q_v}, \widetilde{T}_{q_v})$. We assume a parameter list $(q, q_v, \sigma_N, \sigma_H, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{H}}, \sigma_{\widetilde{C}})$ to define the attack resource, where $\sigma_{\mathsf{Y}} = \sum_{i=1}^{q_v} |\mathsf{Y}_i|_n$ for $\mathsf{Y} \in \{\widetilde{N}, \widetilde{H}, \widetilde{C}\}$, in addition to $\sigma_N$, $\sigma_H$, and $\sigma_M$. The authenticity of $\mathsf{AE}$ is defined as

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathsf{AE}[\tau]}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr[K \stackrel{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{\mathsf{AE}\text{-}\mathcal{E}_\tau, \mathsf{AE}\text{-}\mathcal{D}_\tau} \text{ forges }],$$

where $\mathcal{A}$ forges if it receives a bit string (not $\perp$) from $\mathsf{AE}\text{-}\mathcal{D}_\tau$ for a non-trivial query $(\widetilde{N}_i, \widetilde{H}_i, \widetilde{C}_i, \widetilde{T}_i)$ for some $1 \leq i \leq q_v$. Here $(\widetilde{N}_i, \widetilde{H}_i, \widetilde{C}_i, \widetilde{T}_i)$ is non-trivial if any encryption query-response pair

$(N_j, H_j, M_j, C_j, T_j)$ obtained before satisfies $(\widetilde{N}_i, \widetilde{H}_i, \widetilde{C}_i, \widetilde{T}_i) \neq (N_j, H_j, C_j, T_j)$. We remark that CCA-adversary is always nonce-respecting with respect to encryption queries. This implies that, we can have $N_i = \widetilde{N}_j$ or $\widetilde{N}_i = \widetilde{N}_j$ for some $i$ and $j$. In the security proofs we use the following notion. Let $F_K$ and $G_{K'}$ be two compatible keyed functions with $K \in \mathcal{K}$ and $K' \in \mathcal{K}'$. Then

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

Note that this definition can be naturally extended when $G_{K'}$ is substituted with the random-bit oracle compatible to $F_K$. In addition when $F_K$ and $G_{K'}$ are compatible with $\mathsf{AE}\text{-}\mathcal{E}_\tau$, 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. Also, 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 $(\mathsf{AE}\text{-}\mathcal{E}_\tau, \mathsf{AE}\text{-}\mathcal{D}_\tau)$. We define

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

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

### 3.3 Security Bounds

**Original Bounds.** We denote EAX using an $n$-bit URP as a blockcipher by $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ and the corresponding encryption and decryption functions by $\mathrm{EAX}\text{-}\mathcal{E}_\mathsf{P}$ and $\mathrm{EAX}\text{-}\mathcal{D}_\mathsf{P}$. Similarly, the subscript $K$ in the component algorithms is substituted with $\mathsf{P}$, e.g. $\mathrm{CMAC}^{(i)}_\mathsf{P}$. We focus on the security bounds for $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ as the computational counterparts for $\mathrm{EAX}[E, \tau]$ are trivial.

In [5], Bellare et al. introduced *data complexity* denoted by $\sigma$, which is slightly different from our parameters[5]. The provided bounds are as follows. Note that these theorems assume $H \neq \varepsilon$.

**Theorem 1 ([5]).** *Fix $\tau \in \{1, \ldots, n\}$. Let $\mathcal{A}$ be the CPA-adversary against $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ with data complexity $\sigma$. Then the privacy is bounded as $\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq 9.5\sigma^2/2^n$.*

**Theorem 2 ([5]).** *Fix $\tau \in \{1, \ldots, n\}$. Let $\mathcal{A}$ be the CCA-adversary against $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ with data complexity $\sigma$ and $q_v = 1$. Then the authenticity is bounded as $\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq 11\sigma^2/2^n + 1/2^\tau$.*

**Our Bounds.** The privacy bound of Theorem 1 is the standard birthday bound security. The bound is tight in the sense that there is an adversary that meets the stated security bound up to a constant factor. However, the authenticity bound of Theorem 2 is not satisfactory as it requires $q_v = 1$. There is a known result [3] proving that, if authenticity bound of a scheme for one verification query is $\epsilon$, authenticity bound for $c$ verification queries is bounded by $c\epsilon$, for any $c > 1$. Applying this result to Theorem 2, we have $\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq 11q_v\sigma^2/2^n + q_v/2^\tau$ for $q_v \geq 1$, implying that the security is guaranteed up to $2^{n/3}$ data when $q_v \approx \sigma$. Now we show an improved authenticity bound for EAX that provides security up to $2^{n/2}$ data even for $q_v \geq 1$, with an extended specification allowing $H = \varepsilon$.

**Theorem 3.** *Fix $\tau \in \{1, \ldots, n\}$. Let $\mathcal{A}$ be the CCA-adversary against $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ with parameter list $(q, q_v, \sigma_N, \sigma_H, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{H}}, \sigma_{\widetilde{C}})$. Let $\sigma_{\mathrm{auth}} = \sigma_N + \sigma_H + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{H}} + \sigma_{\widetilde{C}}$. Then we have*

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq \frac{18.5\sigma^2_{\mathrm{auth}} + 4.5}{2^n} + \frac{q_v}{2^\tau}.$$

[5] According to [5], $\sigma$ is defined as "the sum of the lengths of all strings encoded in the adversary's oracle queries, plus the total number of all of these strings".

Note that $\sigma_{\mathrm{auth}}$ is largely the same as the plain $\sigma$ of Theorems 1 and 2. Theorem 3 shows that EAX preserves birthday-type security in the authenticity notion for any $q_v \geq 1$, rather than for $q_v = 1$, only.

As we extended the specification to allow $H = \varepsilon$, a corresponding privacy bound should also be given in principle. For completeness we show the privacy bound in this extended specification.

**Theorem 4.** *Fix $\tau \in \{1, \ldots, n\}$. Let $\mathcal{A}$ be the CPA-adversary against* $\mathrm{EAX}[\mathrm{Perm}(n), \tau]$ *who has parameter list $(q, \sigma_N, \sigma_H, \sigma_M)$. Let $\sigma_{\mathrm{priv}} = \sigma_N + \sigma_H + \sigma_M$. Then we have* $\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \leq (18.5\sigma^2_{\mathrm{priv}} + 4.5)/2^n$.

The proofs of the above theorems are provided in Section 5.1.

## 4 Refinements of EAX

EAX needs three blockcipher calls in advance to the actual processing, namely $L = E_K([0]_n)$, $L' = E_K([1]_n)$, and $L'' = E_K([2]_n)$. They are used as masks for the initial block of CMAC. In addition, CMAC itself needs two masks for the last block, namely $2L$ and $4L$, hence five mask values in total. To achieve the fastest operation, these mask values, at least the first three ones, must be kept in memory while processing. This fact implies that EAX is not ultimately optimized, in particular for short messages, when the amount of pre-processing is critical. This is possible due to some practical reasons, e.g., a huge number of keys, or frequent key changes. In addition retaining many mask values in memory may not be desirable for constrained devices, such as low-end micro-controllers or tiny hardware.

We propose a refinement of EAX, which we call $\mathrm{EAX}^+$, to minimize these drawbacks. We note that EAX seems to have a design philosophy for keeping the algorithm of CMAC intact, and our proposal does not follow this design philosophy in return for the efficiency gain.

Specifically, $\mathrm{EAX}^+$ changes the definitions of five mask values so that they are simple functions of $L = E_K([0]_n)$. $\mathrm{EAX}^+$ also sets some initial counter bits off to suppress carry bit propagation. This is the technique used by SIV [17] and EAX-prime to simplify the implementation of the counter mode. These changes affect the definitions of two internal components, $\mathrm{CMAC}^{(i)}$ and CTR. $\mathrm{EAX}^+$ uses $\mathrm{CMAC}^{+(i)}$ and $\mathrm{CTR}^+$, as shown in Fig. 3, instead of $\mathrm{CMAC}^{(i)}$ and CTR of Fig. 1. For simplicity Fig. 3 assumes $n = 128$ for fixing the constant adjusting the initial counter, however other values of $n$ are possible. In $\mathrm{CMAC}^{+(i)}$, the five mask values are denoted by $A(0)$, $A(1)$, $A(2)$, $D$, and $Q$, and they are functions of $L = E_K([0]_n)$ denoted by $g_{A(0)}$, $g_{A(1)}$, and so on. In the following, we give three concrete masking schemes.

**Scheme 1: Use GF Doubling.** The first scheme, which we call $\mathrm{EAX}^+_1$, uses the following masks. Here, $3L$ denotes $2L \oplus L$.

$$A(0) = 3L, \qquad A(1) = 2 \cdot 3L, \qquad A(2) = 2^2 \cdot 3L,$$
$$D = 2L, \qquad Q = 2^2 L$$

This keeps the definitions of CMAC masks for the last blocks ($D$ and $Q$). Note that we have $A(0) = 2L \oplus L$, $A(1) = 2^2 L \oplus 2L$, and $A(2) = 2^3 L \oplus 2^2 L$. Any mask is efficiently computed by holding $\mathsf{X} = L$ and $\mathsf{Y} = 2^2 L$, as we have $A(0) = 2\mathsf{X} \oplus \mathsf{X}$, $A(1) = \mathsf{Y} \oplus 2\mathsf{X}$, $A(2) = 2\mathsf{Y} \oplus \mathsf{Y}$, $D = 2\mathsf{X}$, and $Q = \mathsf{Y}$. Each mask computation requires at most one doubling and one XOR.

**Scheme 2: Use Sum of Four Quarters of $L$.** The second scheme, which we call $\mathrm{EAX}^+_2$, assumes that $n$ is divisible by 4, and uses operations over $\mathrm{GF}((2^{n/4})^4)$. Let $L = (L_1, L_2, L_3, L_4)$, where $L_i \in \mathrm{GF}(2^{n/4})$. The masks are as follows.

$$A(0) = (L_1, L_2, L_3, L_4), \qquad A(1) = (L_*, L_1, L_2, L_3), \qquad A(2) = (L_4, L_*, L_1, L_2),$$
$$D = (L_3, L_4, L_*, L_1), \qquad Q = (L_2, L_3, L_4, L_*),$$

| **Algorithm** $\mathrm{CMAC}^{+(i)}_K(M)$ | **Algorithm** $\mathrm{CTR}^{+}_K(\underline{N}, M)$ |
|---|---|
| 1. $L \leftarrow E_K([0]_n)$<br>2. **for** $i = 0, 1, 2$ **do** $A(i) \leftarrow g_{A(i)}(L)$<br>3. $D \leftarrow g_D(L)$, $Q \leftarrow g_Q(L)$<br>4. **if** $M = \varepsilon$ **return** $E_K([i]_n \oplus D)$<br>5. **else return**<br>$\mathrm{CBC}_K(A(i), \mathtt{pad}(M; D, Q))$ | 1. $m \leftarrow \lvert M\rvert_n$<br>2. $\underline{N}^{\wedge} \leftarrow \underline{N} \wedge (1^{n-64} \| 01^{31} \| 01^{31})$<br>3. $S \leftarrow E_K(\underline{N}^{\wedge}) \| E_K(\underline{N}^{\wedge} + 1) \|$<br>$\cdots \| E_K(\underline{N}^{\wedge} + m - 1)$<br>4. $C \leftarrow M \oplus \mathrm{msb}_{\lvert M\rvert}(S)$<br>5. **return** $C$ |

**Fig. 3.** Our refinement of EAX, $\mathrm{EAX}^{+}$. Here, $\mathrm{CMAC}^{+(i)}$ for $i \in \{0, 1, 2\}$ and $\mathrm{CTR}^{+}$ are used instead of $\mathrm{CMAC}^{(i)}$ and CTR, and other functions are not changed. The definitions of $g_A(i)$, $g_D$, and $g_Q$ are written in Section 4, yielding the three versions.

where $L_* = L_1 \oplus L_2 \oplus L_3 \oplus L_4$. The scheme is efficient, in particular for software, since it is merely a combination of $n/4$-bit word permutations and XORs. Specifically, any mask can be efficiently computed by holding $L$ and $L_*$, which are $5n/4$ bits in total.

**Scheme 3: Use Two Sums.** The third scheme, which we call $\mathrm{EAX}^{+}_3$, is another instance using word permutation and XOR. The masks are;

$$A(0) = (L_1, L_2, L_3, L_4), \quad A(1) = (L_2, L_\sharp, L_4, L_\flat), \quad A(2) = (L_\sharp, L_1, L_\flat, L_3),$$
$$D = (L_3, L_\flat, L_2, L_1), \quad Q = (L_4, L_3, L_\sharp, L_2),$$

where $L_\sharp = L_1 \oplus L_2$ and $L_\flat = L_3 \oplus L_4$. The mask generation is a simple word permutation by holding $L$, $L_\sharp$, and $L_\flat$. Even if we only hold $L$, each mask is computed by at most 2 XORs of words and a permutation, and the number of word XORs required for generating all masks from $L$ is 6.

The following theorem shows the security of these schemes.

**Theorem 5.** *For $j \in \{1, 2, 3\}$, let $\mathrm{EAX}^{+}_j[\mathrm{Perm}(n), \tau]$ be $\mathrm{EAX}^{+}_j$ using $n$-bit URP. For any $j$ we have*

$$\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}^{+}_j[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \le \frac{15\sigma^2_{\mathrm{priv}}}{2^n}, \text{ and}$$

$$\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}^{+}_j[\mathrm{Perm}(n),\tau]}(\mathcal{A}) \le \frac{15\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q_v}{2^\tau}.$$

The core part of the proof of Theorem 5 is given in Section 5.2. We can build variants of these schemes by applying a permutation $\mathbb{P}$ that commutes with respect to XOR, i.e. $\mathbb{P}(x) \oplus \mathbb{P}(y) = \mathbb{P}(x \oplus y)$, to all masks. An example is a regular matrix over $\mathrm{GF}(2^{n/a})^a$ for $a$ being a factor of $n$, and a variant using such masks will have the same bounds as Theorem 5.

We note that [14] suggested some variants of EAX-prime that are provably secure without input-length restriction. However the proposals of [14] focuses on the black-box usage of EAX-prime. As a result the proposals of [14] are not as efficient as ours, or require a stronger security assumption on the blockcipher.

## 5 Security Proofs

### 5.1 Proofs of Theorem 3 and Theorem 4

**OMAC-extension.** In proving Theorems 3 and 4, we observe that the most part are quite the same as those given for EAX-prime [14], which is based on the original proof of EAX [5] with extensions taken from [10].

**Algorithm** OMAC-e[P] :
**Initialization**
00 $L \leftarrow \mathsf{P}([0]_n)$, $L' \leftarrow \mathsf{P}([1]_n)$, $L'' \leftarrow \mathsf{P}([2]_n)$
**On query** $(t, X, d) \in \{0,1,2\} \times \{0,1\}^* \times \mathbb{N}$
10 $X[1]\|X[2]\|\cdots\|X[m] \xleftarrow{n} X$
11 **if** $|X| \bmod n \neq 0$ **or** $X = \varepsilon$ **then** $w \leftarrow 1$, **else** $w \leftarrow 0$
12 **if** $t = 0$
13 **if** $1 \leq |X| \leq n$ **then** $Y \leftarrow \mathsf{P}(\mathtt{bp}(X) \oplus L \oplus 2^{w+1}L)$; **return** $Y$
14 **else** $Y[1] \leftarrow \mathsf{P}(L \oplus X[1])$
15 **for** $i = 1$ **to** $m-2$ **do** $Y[i+1] \leftarrow \mathsf{P}(Y[i] \oplus X[i+1])$
16 $Y \leftarrow \mathsf{P}(Y[m-1] \oplus \mathtt{bp}(X[m]) \oplus 2^{w+1}L)$
17 **if** $d = 0$ **return** $Y$
18 **for** $j = 0$ **to** $d-1$ **do** $S[j+1] \leftarrow \mathsf{P}(Y+j)$
19 **return** $Y\|S[1]S[2]\cdots S[d]$
20 **if** $t = 1$
21 **if** $|X| = 0$ **then** $Y' \leftarrow \mathsf{P}(2L \oplus [1]_n)$; **return** $Y'$
22 **if** $1 \leq |X| \leq n$ **then** $Y' \leftarrow \mathsf{P}(\mathtt{bp}(X) \oplus L' \oplus 2^{w+1}L)$; **return** $Y'$
23 **else** $Y'[1] \leftarrow \mathsf{P}(L' \oplus X[1])$
24 **for** $i = 1$ **to** $m-2$ **do** $Y'[i+1] \leftarrow \mathsf{P}(Y'[i] \oplus X[i+1])$
25 $Y' \leftarrow \mathsf{P}(Y'[m-1] \oplus \mathtt{bp}(X[m]) \oplus 2^{w+1}L)$
26 **return** $Y'$
27 **if** $t = 2$
28 **if** $|X| = 0$ **then** $Y'' \leftarrow \mathsf{P}(2L \oplus [2]_n)$; **return** $Y''$
29 **if** $1 \leq |X| \leq n$ **then** $Y'' \leftarrow \mathsf{P}(\mathtt{bp}(X) \oplus L'' \oplus 2^{w+1}L)$; **return** $Y''$
30 **else** $Y''[1] \leftarrow \mathsf{P}(L'' \oplus X[1])$
31 **for** $i = 1$ **to** $m-2$ **do** $Y''[i+1] \leftarrow \mathsf{P}(Y''[i] \oplus X[i+1])$
32 $Y'' \leftarrow \mathsf{P}(Y''[m-1] \oplus \mathtt{bp}(X[m]) \oplus 2^{w+1}L)$
33 **return** $Y''$

**Fig. 4.** OMAC-extension using an $n$-bit URP, P.

We start with the most involved part: the pseudorandomness of OMAC-extension. OMAC-extension is a set of functions obtained by decomposing EAX [5,14]. Formally, we define OMAC-extension[6] as a set of three functions using an $n$-bit URP, P, obtained from EAX[Perm$(n), \tau$]. It is denoted by OMAC-e[P] = (OMAC-e[P]$^{(0)}$, OMAC-e[P]$^{(1)}$, OMAC-e[P]$^{(2)}$). See Figs. 4 and 7. Here, OMAC-e[P]$^{(0)}$ is a function that takes $(N, d)$, where $d = |M|_n$ ($d = |C|_n$) for encryption (decryption), and produces $\underline{N}$ and the $d$-block keystream before truncation, i.e., $S$ of Fig. 1. Similarly, OMAC-e[P]$^{(1)}$ takes $H$, and OMAC-e[P]$^{(2)}$ takes $C$. We may view OMAC-e[P] as single function taking $(t, X, d)$ as input and outputs OMAC-e[P]$^{(t)}(X, d)$ when $t = 0$ and OMAC-e[P]$^{(t)}(X)$ when $t = 1, 2$, assuming $d$ is a default value.

Similarly to Proposition 1 of [14], we have the following proposition.

**Proposition 1.** *For any fixed $\tau$, 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* EAX-$\mathcal{E}_{\mathsf{P}} \equiv f_e($OMAC-e[P]$)$ *and* EAX-$\mathcal{D}_{\mathsf{P}} \equiv f_d($OMAC-e[P]$)$.

Here, $F \equiv G$ means the equivalence of the output probability distribution functions for $F$ and $G$, 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 $y_1, \ldots, y_q$. Proposition 1 can be verified by comparing Figs. 2 and 7.

Then we need to evaluate the indistinguishability between OMAC-e[P] and a set of three random functions $\mathbb{RND} = (\mathbb{RND}^{(0)}, \mathbb{RND}^{(1)}, \mathbb{RND}^{(2)})$, where $\mathbb{RND}^{(i)}$ is compatible with OMAC-e[P]$^{(i)}$. Here $\mathbb{RND}^{(0)}(X, d)$ samples $Y \xleftarrow{\$} (\{0,1\}^n)^{d_{\max}+1}$ and outputs $\mathrm{msb}_{n(d+1)}(Y)$ if $X$ is new, where $d_{\max}$ is the largest possible value of $d$ determined by the game we consider.

To bound the indistinguishability, we further break down OMAC-e[P] into a set of 19 small functions, $\mathbf{Q} = \{\mathbf{Q}_i\}_{i=1,\ldots,19}$.

[6] Our OMAC-extension does not need the auxiliary output mask as in the proof of EAX-prime [14]. This is because of the difference in the processing for one-block inputs.

**Definition 1.** *Let* $\mathbf{Q}_i : \{0,1\}^n \to \{0,1\}^n$ *for* $i \in \{1,2,\ldots,19\} \setminus \{3,4,5,6\}$ *and let* $\mathbf{Q}_j : \{0,1\}^n \times \mathbb{N} \to (\{0,1\}^n)^{>0}$ *for* $j = 3,4,5,6$. *We define*

$$\mathbf{Q}_1(x) \stackrel{\text{def}}{=} \mathsf{P}(L \oplus x) \oplus \mathtt{Rnd}_1, \qquad \mathbf{Q}_2(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_1 \oplus x) \oplus \mathtt{Rnd}_1,$$
$$\mathbf{Q}_3(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(2L \oplus \mathtt{Rnd}_1 \oplus x), d), \qquad \mathbf{Q}_4(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(4L \oplus \mathtt{Rnd}_1 \oplus x), d),$$
$$\mathbf{Q}_5(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(L \oplus 2L \oplus x), d), \qquad \mathbf{Q}_6(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(L \oplus 4L \oplus x), d),$$
$$\mathbf{Q}_7(x) \stackrel{\text{def}}{=} \mathsf{P}(L' \oplus x) \oplus \mathtt{Rnd}_2, \qquad \mathbf{Q}_8(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_2 \oplus x) \oplus \mathtt{Rnd}_2,$$
$$\mathbf{Q}_9(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus \mathtt{Rnd}_2 \oplus x), \qquad \mathbf{Q}_{10}(x) \stackrel{\text{def}}{=} \mathsf{P}(4L \oplus \mathtt{Rnd}_2 \oplus x),$$
$$\mathbf{Q}_{11}(x) \stackrel{\text{def}}{=} \mathsf{P}(L' \oplus 2L \oplus x), \qquad \mathbf{Q}_{12}(x) \stackrel{\text{def}}{=} \mathsf{P}(L' \oplus 4L \oplus x),$$
$$\mathbf{Q}_{13}(x) \stackrel{\text{def}}{=} \mathsf{P}(L'' \oplus x) \oplus \mathtt{Rnd}_3, \qquad \mathbf{Q}_{14}(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_3 \oplus x) \oplus \mathtt{Rnd}_3,$$
$$\mathbf{Q}_{15}(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus \mathtt{Rnd}_3 \oplus x), \qquad \mathbf{Q}_{16}(x) \stackrel{\text{def}}{=} \mathsf{P}(4L \oplus \mathtt{Rnd}_3 \oplus x),$$
$$\mathbf{Q}_{17}(x) \stackrel{\text{def}}{=} \mathsf{P}(L'' \oplus 2L \oplus x), \qquad \mathbf{Q}_{18}(x) \stackrel{\text{def}}{=} \mathsf{P}(L'' \oplus 4L \oplus x),$$
$$\mathbf{Q}_{19}(x) \stackrel{\text{def}}{=} \mathsf{P}(2L \oplus x),$$

*where* $\mathsf{P}$ *is an* $n$*-bit URP, and* $L = \mathsf{P}([0]_n)$, $L' = \mathsf{P}([1]_n)$, *and* $L'' = \mathsf{P}([2]_n)$. *Also,* $\mathtt{Rnd}_1$, $\mathtt{Rnd}_2$ *and* $\mathtt{Rnd}_3$ *are independent* $n$*-bit random sequences, and* $G_{\mathsf{P}}(v,d)$ *is* $v$ *if* $d = 0$ *and* $(v \| \mathsf{P}(v) \| \mathsf{P}(v+1) \| \cdots \| \mathsf{P}(v+(d-1)))$ *if* $d > 0$. *The sampling procedures for* $\mathsf{P}, \mathtt{Rnd}_j$ *for* $j = 1,2,3$ *are shared for all* $\mathbf{Q}_i$*s.*

We treat $\mathbf{Q}$ as a tweakable function with tweak $t \in \{1,\ldots,19\}$ by writing $\mathbf{Q}(t,x,d) = \mathbf{Q}_t(x,d)$ when $t \in \{3,4,5,6\}$ and otherwise $\mathbf{Q}(t,x,d) = \mathbf{Q}_t(x)$. We observe that OMAC-e[$\mathsf{P}$] can be simulated with black-box accesses to $\mathbf{Q}$. For example, when we want to simulate the computation of OMAC-e[$\mathsf{P}$]$(0,N,2)$ for $|N| = 3n$, we first parse $N$ into $n$-bit blocks, i.e., $N[1]\|N[2]\|N[3] \xleftarrow{n} N$ and then proceed as $Y[1] \leftarrow \mathbf{Q}_1(N[1])$, and $Y[2] \leftarrow \mathbf{Q}_3(N[2] \oplus Y[1])$, and $Y[3]\|S[1]S[2] \leftarrow \mathbf{Q}_5(N[3] \oplus Y[2], 2)$. Note that, $\mathbf{Q}_{19}$ is only used to simulate $\mathrm{CMAC}_K^{(1)}$ given $H = \varepsilon$, or $\mathrm{CMAC}_K^{(2)}$ given $C = \varepsilon$, i.e., to compute $\mathsf{P}(2L \oplus [1]_n)$ or $\mathsf{P}(2L \oplus [2]_n)$.

We next define $\widetilde{\mathbf{Q}} = \{\widetilde{\mathbf{Q}}_i\}_{i=1,\ldots,19}$. For all $i = 1,\ldots,19$, $\widetilde{\mathbf{Q}}_i$ is compatible to $\mathbf{Q}_i$.

**Definition 2.** *Let* $\mathsf{P}_i$ *for* $i = 1,2,7,8,13,14$ *be six independent* $n$*-bit URPs, and let* $\mathsf{R}_j$ *for* $j \in \{9,\ldots,19\} \setminus \{13,14\}$ *be nine independent* $n$*-bit URFs, and let* $\mathsf{R}_j$ *for* $j = 3,4,5,6$ *be four independent URFs with* $n$*-bit input and* $(d_{\max}+1)n$*-bit output. Using them we define*

$$\widetilde{\mathbf{Q}}_i(x) \stackrel{\text{def}}{=} \mathsf{P}_i(x), \text{ for } i = 1,2,7,8,13,14$$
$$\widetilde{\mathbf{Q}}_j(x,d) \stackrel{\text{def}}{=} \mathsf{R}_j^{d+1}(x), \text{ for } j = 3,4,5,6$$
$$\widetilde{\mathbf{Q}}_h(x) \stackrel{\text{def}}{=} \mathsf{R}_h(x), \text{ for } h = 9,\ldots,12,15,\ldots,19,$$

*where* $\mathsf{R}_i^{d+1}(x) = \mathrm{msb}_{n(d+1)}(\mathsf{R}_i(x))$ *for* $i = 3,4,5,6$. *Here* $d_{\max}$ *is the maximum possible value of queried* $d$, *which will be determined by the underlying game and adversary's parameter.*

A function compatible to $\mathbf{Q}$ 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 number of total output $n$-bit blocks for $t \in \{3,4,5,6\}$, $\sigma_{\text{out}}$. 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* $(q, \sigma_{\text{out}})$. *Then we have*

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{A}) \le \frac{(4.5q^2 + 10\sigma_{\text{out}}q + \sigma_{\text{out}}^2 + 4.5)}{2^n}.$$

**Algorithm** $\mathbb{CBC}$ (given $d_{\max}$):
**Initialization**
00 **for** $i = 1, 2, 7, 8, 13, 14$ **do** $\mathsf{P}_i \xleftarrow{\$} \mathrm{Perm}(n)$
01 **for** $j = 3, 4, 5, 6$ **do** $\mathsf{R}_j \xleftarrow{\$} \mathrm{Func}(n, d_{\max})$
02 **for** $k = 9, 10, 11, 12, 15, 16, 17, 18, 19$ **do** $\mathsf{R}_k \xleftarrow{\$} \mathrm{Func}(n)$
**On query** $(t, X, d) \in \{0,1,2\} \times \{0,1\}^* \times \mathbb{N}$
10 $X[1]\|X[2]\|\cdots\|X[m] \xleftarrow{n} X$
11 **if** $|X| \bmod n \neq 0$ **or** $X = \varepsilon$ **then** $w \leftarrow 1$, **else** $w \leftarrow 0$
12 **if** $t = 0$
13 **if** $m = 1$ **and** $d = 0$ **return** $Y \leftarrow \mathsf{R}^1_{5+w}(\mathtt{bp}(X))$
14 **if** $m = 1$ **and** $d > 0$ **return** $Y\|S[1]\|S[2]\|\cdots\|S[d] \leftarrow \mathsf{R}^{d+1}_{5+w}(\mathtt{bp}(X))$
15 $Y[1] \leftarrow \mathsf{P}_1(X[1])$
16 **for** $i = 1$ **to** $m - 2$ **do** $Y[i+1] \leftarrow \mathsf{P}_2(Y[i] \oplus X[i+1])$
17 **if** $d = 0$ **then** $Y \leftarrow \mathsf{R}^1_{3+w}(Y[m-1] \oplus \mathtt{bp}(X[m]))$; **return** $Y$
18 **else** $Y\|S[1]\|S[2]\|\cdots\|S[d] \leftarrow \mathsf{R}^{d+1}_{3+w}(Y[m-1] \oplus \mathtt{bp}(X[m]))$
19 **return** $Y\|S[1]\|S[2]\|\cdots\|S[d]$
20 **if** $t = 1$
21 **if** $X = \varepsilon$ **then** $Y' \leftarrow \mathsf{R}_{19}([1]_n)$; **return** $Y'$
22 **else if** $m = 1$ **then** $Y' \leftarrow \mathsf{R}_{11+w}(\mathtt{bp}(X))$; **return** $Y'$
23 **else** $Y'[1] \leftarrow \mathsf{P}_7(X[1])$
24 **for** $i = 1$ **to** $m - 2$ **do** $Y'[i+1] \leftarrow \mathsf{P}_8(Y'[i] \oplus X[i+1])$
25 $Y' \leftarrow \mathsf{R}_{9+w}(Y'[m-1] \oplus \mathtt{bp}(X[m]))$
26 **return** $Y'$
27 **if** $t = 2$
28 **if** $X = \varepsilon$ **then** $Y'' \leftarrow \mathsf{R}_{19}([2]_n)$; **return** $Y''$
29 **else if** $m = 1$ **then** $Y'' \leftarrow \mathsf{R}_{17+w}([2]_n)$; **return** $Y''$
30 **else** $Y''[1] \leftarrow \mathsf{P}_{13}(X[1])$
31 **for** $i = 1$ **to** $m - 2$ **do** $Y''[i+1] \leftarrow \mathsf{P}_{14}(Y''[i] \oplus X[i+1])$
32 $Y'' \leftarrow \mathsf{R}_{15+w}(Y''[m-1] \oplus \mathtt{bp}(X[m]))$
33 **return** $Y''$

**Fig. 5.** Modified CBC-MAC.

The proof is in Appendix A.

The remaining part of the proof is almost the same as the proof of EAX-prime. We define the Modified CBC-MAC, $\mathbb{CBC}$, which is compatible with OMAC-e[P] and consists of three functions taking $t = 0, 1, 2$, written as $\mathbb{CBC}^{(t)}$. A combined form of $\mathbb{CBC}$ is shown in Fig. 5. Here, $\mathsf{R}^i_j(X)$ for $j = 3, 4, 5, 6$ denotes $\mathrm{msb}_{n \cdot i}(\mathsf{R}_j(X))$. Then, we obtain the following proposition and lemma as counterparts of Proposition 2 and Lemma 2 of [14]. Proof of Proposition 2 is almost the same as [14]. Proof of Lemma 2 is given in Appendix B.

**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 $h(\cdot)$.*

**Lemma 2.** *Let $\mathcal{A}$ be an adversary querying a function of OMAC-e profile, and let $\sigma_{\mathrm{in}}$ denote the number of total blocks of queries made by $\mathcal{A}$. Then, $\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{A}) \leq 3\sigma^2_{\mathrm{in}}/2^n$.*

Our PRIV bound is derived by combining Propositions 1 and 2 and Lemma 2 in the same manner to [14]. Formally, let $\mathcal{A}$ be the CPA-adversary against AE with parameter list $(q, \sigma_N, \sigma_H, \sigma_M)$. Then there exist adversary $\mathcal{B}$ querying to a function of OMAC-e profile with $3q$ queries, $\sigma_{\mathrm{in}} = (\sigma_N + \sigma_H + \sigma_M)$ input blocks, and $\sigma_{\mathrm{out}} = \sigma_M + 3q$ output blocks, and adversary $\mathcal{C}$ querying to a set of 19 functions with $\mathbf{Q}$ profile, using $\sigma_N + \sigma_H + \sigma_M$ queries and $\sigma_M + q$ output $n$-bit blocks for queries with $t = 3, 4, 5, 6$, such that

$$
\begin{aligned}
&\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX[Perm}(n),\tau]}(\mathcal{A}) \\
&= \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{\text{EAX-}\mathcal{E}_{\mathsf{P}},\$}(\mathcal{A}) \\
&= \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\text{OMAC-e[P]}),\$}(\mathcal{A}) \text{ (from Proposition 1)} \\
&\le \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\text{OMAC-e[P]}),f_e(\mathbb{CBC})}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\mathbb{CBC}),f_e(\mathbb{RND})}(\mathcal{A}) + \underbrace{\mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\mathbb{RND}),\$}(\mathcal{A})}_{=0} \\
&\le \mathtt{Adv}^{\mathtt{cpa}}_{\text{OMAC-e[P]},\mathbb{CBC}}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) \\
&= \mathtt{Adv}^{\mathtt{cpa}}_{h(\mathbf{Q}),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) \text{ (from Proposition 2)} \\
&\le \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{3(\sigma_N+\sigma_H+\sigma_M)^2}{2^n} \text{ (from Lemma 2)} \\
&\le \frac{4.5(\sigma_N+\sigma_H+\sigma_M)^2 + 10(\sigma_M+q)(\sigma_N+\sigma_H+\sigma_M) + (\sigma_M+q)^2 + 4.5}{2^n} \\
&\quad + \frac{3(\sigma_N+\sigma_H+\sigma_M)^2}{2^n} \text{ (from Lemma 1)} \\
&\le \frac{18.5(\sigma_N+\sigma_H+\sigma_M)^2 + 4.5}{2^n} \\
&\le \frac{18.5\sigma_{\mathrm{priv}}^2 + 4.5}{2^n}.
\end{aligned}
$$

Here, $\mathtt{Adv}^{\mathtt{cpa\text{-}nr}}_{f_e(\mathbb{RND}),\$}(\mathcal{A}) = 0$ holds because when $\mathcal{A}$ queries $(N,H,M)$ 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 $\mathbb{RND}^{(1)}$ and $\mathbb{RND}^{(2)}$ (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 4.

In proving AUTH bound, let $\mathbb{EAX}$ be the AE algorithm compatible to 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 with parameter list $(q, q_v, \sigma_N, \sigma_H, \sigma_M, \sigma_{\widetilde{N}}, \sigma_{\widetilde{H}}, \sigma_{\widetilde{C}})$. Then we have

$$
\mathtt{Adv}^{\mathtt{auth}}_{\mathbb{EAX}}(\mathcal{A}) \le \frac{q_v}{2^\tau} \tag{1}
$$

with almost the same proof as Eq. (14) of [14]. For completeness the proof is given in Appendix C. In the same way as Eq. (15) to (22) of [14], we assume adversary $\mathcal{B}$ querying to a function of OMAC-e profile with $3(q+q_v)$ queries with $\sigma_{\mathrm{in}} = \sigma_N+\sigma_H+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{H}}+\sigma_{\widetilde{C}}$ and $\sigma_{\mathrm{out}} = \sigma_M + 3q+\sigma_{\widetilde{C}}+3q_v$, and adversary $\mathcal{C}$ querying to a function of $\mathbf{Q}$ profile with $\sigma_N+\sigma_H+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{H}}+\sigma_{\widetilde{C}}$ queries and $\sigma_M+q+\sigma_{\widetilde{C}}+q_v$ output blocks for queries with $t = 3,4,5,6$. Then using Proposition 1 and Eq. (1) we have

$$
\begin{aligned}
\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX[Perm}(n)]}(\mathcal{A}) &\le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{(\text{EAX-}\mathcal{E}_{\mathsf{P}},\text{EAX-}\mathcal{D}_{\mathsf{P}}),(f_e(\mathbb{RND}),f_d(\mathbb{RND}))}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{auth}}_{\mathbb{EAX}}(\mathcal{A}) \\
&\le \mathtt{Adv}^{\mathtt{cca\text{-}nr}}_{(f_e(\text{OMAC-e[P]}),f_d(\text{OMAC-e[P]})),(f_e(\mathbb{RND}),f_d(\mathbb{RND}))}(\mathcal{A}) + \frac{q_v}{2^\tau} \\
&\le \mathtt{Adv}^{\mathtt{cpa}}_{\text{OMAC-e[P]},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau}.
\end{aligned}
$$

The right hand side of the last equation is bounded by

$$
\begin{aligned}
&\mathtt{Adv}^{\mathtt{cpa}}_{\text{OMAC-e[P]},\mathbb{CBC}}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \\
&= \mathtt{Adv}^{\mathtt{cpa}}_{h(\mathbf{Q}),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \quad \text{(from Proposition 2)} \\
&\le \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{3(\sigma_N + \sigma_H + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{H}} + \sigma_{\widetilde{C}})^2}{2^n} + \frac{q_v}{2^\tau} \quad \text{(from Lemma 2)} \\
&\le \frac{4.5(\sigma_N + \sigma_H + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{H}} + \sigma_{\widetilde{C}})^2}{2^n} \\
&\quad + \frac{10(\sigma_M + q + \sigma_{\widetilde{C}} + q_v)(\sigma_N + \sigma_H + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{H}} + \sigma_{\widetilde{C}})}{2^n} \\
&\quad + \frac{(\sigma_M + q + \sigma_{\widetilde{C}} + q_v)^2 + 4.5}{2^n} + \frac{3(\sigma_N + \sigma_H + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{H}} + \sigma_{\widetilde{C}})^2}{2^n} + \frac{q_v}{2^\tau} \quad \text{(from Lemma 1)} \\
&\le \frac{18.5\sigma_{\text{auth}}^2 + 4.5}{2^n} + \frac{q_v}{2^\tau},
\end{aligned}
$$

since $\sigma_M + q + \sigma_{\widetilde{C}} + q_v \le \sigma_N + \sigma_M + \sigma_{\widetilde{N}} + \sigma_{\widetilde{C}}$. This concludes the proof of Theorem 3.

### 5.2 Proof of Theorem 5

As the proof of Theorem 5 is quite the same as those of Theorem 3 and Theorem 4, we only show the differences. The main difference is in Lemma 1. We in fact need to change the definitions of OMAC-e[P] and $\mathbf{Q}$, and evaluate the indistinguishability of $\mathbf{Q}$ from $\widetilde{\mathbf{Q}}$, where the definition of $\widetilde{\mathbf{Q}}$ does not change, to have a counterpart of Lemma 1. We define $\mathbf{Q}'$ as follows.

**Definition 3.** *Let $\mathbf{Q}' = \{\mathbf{Q}'_i\}_{i=1,\ldots,19}$ be a set of 19 functions defined as*

$$
\begin{aligned}
&\mathbf{Q}'_1(x) \stackrel{\text{def}}{=} \mathsf{P}(A(0) \oplus x) \oplus \mathtt{Rnd}_1, && \mathbf{Q}'_2(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_1 \oplus x) \oplus \mathtt{Rnd}_1, \\
&\mathbf{Q}'_3(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(D \oplus \mathtt{Rnd}_1 \oplus x), d), && \mathbf{Q}'_4(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(Q \oplus \mathtt{Rnd}_1 \oplus x), d), \\
&\mathbf{Q}'_5(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(A(0) \oplus D \oplus x), d), && \mathbf{Q}'_6(x,d) \stackrel{\text{def}}{=} G_{\mathsf{P}}(\mathsf{P}(A(0) \oplus Q \oplus x), d), \\
&\mathbf{Q}'_7(x) \stackrel{\text{def}}{=} \mathsf{P}(A(1) \oplus x) \oplus \mathtt{Rnd}_2, && \mathbf{Q}'_8(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_2 \oplus x) \oplus \mathtt{Rnd}_2, \\
&\mathbf{Q}'_9(x) \stackrel{\text{def}}{=} \mathsf{P}(D \oplus \mathtt{Rnd}_2 \oplus x), && \mathbf{Q}'_{10}(x) \stackrel{\text{def}}{=} \mathsf{P}(Q \oplus \mathtt{Rnd}_2 \oplus x), \\
&\mathbf{Q}'_{11}(x) \stackrel{\text{def}}{=} \mathsf{P}(A(1) \oplus D \oplus x), && \mathbf{Q}'_{12}(x) \stackrel{\text{def}}{=} \mathsf{P}(A(1) \oplus Q \oplus x), \\
&\mathbf{Q}'_{13}(x) \stackrel{\text{def}}{=} \mathsf{P}(A(2) \oplus x) \oplus \mathtt{Rnd}_3, && \mathbf{Q}'_{14}(x) \stackrel{\text{def}}{=} \mathsf{P}(\mathtt{Rnd}_3 \oplus x) \oplus \mathtt{Rnd}_3, \\
&\mathbf{Q}'_{15}(x) \stackrel{\text{def}}{=} \mathsf{P}(D \oplus \mathtt{Rnd}_3 \oplus x), && \mathbf{Q}'_{16}(x) \stackrel{\text{def}}{=} \mathsf{P}(Q \oplus \mathtt{Rnd}_3 \oplus x), \\
&\mathbf{Q}'_{17}(x) \stackrel{\text{def}}{=} \mathsf{P}(A(2) \oplus D \oplus x), && \mathbf{Q}'_{18}(x) \stackrel{\text{def}}{=} \mathsf{P}(A(2) \oplus Q \oplus x), \\
&\mathbf{Q}'_{19}(x) \stackrel{\text{def}}{=} \mathsf{P}(D \oplus x), &&
\end{aligned}
$$

*where $\mathsf{P}$ is an URP, and $A(i) = g_{A(i)}(L)$ for $i = 0,1,2$, $D = g_D(L)$ and $Q = g_Q(L)$ with $L = \mathsf{P}([0]_n)$. The set of $g$ functions are one of the schemes in Section 4. $\mathtt{Rnd}_i$ is independently random, and $G_{\mathsf{P}}(v,d)$ is $v$ if $d = 0$ and $(v \| \mathsf{P}(v \wedge \alpha) \| \mathsf{P}((v \wedge \alpha) + 1) \| \cdots \| \mathsf{P}((v \wedge \alpha) + (d-1)))$ if $d > 0$, with $\alpha = (1^{n-64} \| 01^{31} \| 01^{31})$.*

The indistinguishability between $\mathbf{Q}'$ and $\widetilde{\mathbf{Q}}$ is bounded as follows.

**Lemma 3.** *For $\mathbf{Q}'$ of Definition 3, and $\widetilde{\mathbf{Q}}$ of Definition 2, and for any $\mathcal{A}$ with parameter list $(q, \sigma_{\text{out}})$, we have*

$$
\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q}',\widetilde{\mathbf{Q}}}(\mathcal{A}) \le \frac{3q^2 + 6.5\sigma_{\text{out}}q + 2.5\sigma_{\text{out}}^2}{2^n}.
$$

The proof of Lemma 3 is largely the same as that of Lemma 1. We obtain the mask function corresponding to Eq. (3) of Appendix A by the substitution of masks, i.e., $(L, L', L'', 2L, 4L)$ is now $(A(0), A(1), A(2), D, Q)$, while omask function does not change.

Let $\mathcal{M}_1 = \{A(0), A(1), A(2)\}$ and $\mathcal{M}_2 = \{D, Q\}$ and $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$. With these definitions, we need to prove counterparts of Eqs. (4) and (5) of Appendix A. As masks in $\mathcal{M}$ are functions of $L$, when the equation contains an XOR of $\mathtt{Rnd}_i$ for some $i$, the probability is always $1/2^n$. Consequently, we only need to see if the masks satisfy the following conditions[7];

1. $\max_{X \in \mathcal{M}, c \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \{0,1\}^n : X = c] \le 1/2^n$
2. $\max_{X,X' \in \mathcal{M}, X \neq X', c \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \{0,1\}^n : X \oplus X' = c] \le 1/2^n$
3. $\max_{X,X' \in \mathcal{M}_1, X \neq X', c \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \{0,1\}^n : X \oplus X' \oplus D = c] \le 1/2^n$
4. $\max_{X,X' \in \mathcal{M}_1, X \neq X', c \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \{0,1\}^n : X \oplus X' \oplus Q = c] \le 1/2^n$
5. $\max_{X,X' \in \mathcal{M}_1, X \neq X', c \in \{0,1\}^n} \Pr[L \xleftarrow{\$} \{0,1\}^n : X \oplus X' \oplus D \oplus Q = c] \le 1/2^n$

In fact, for any instance of $\mathrm{EAX}^+_j$, we observe that all five conditions are satisfied.

**Proposition 3.** *For any $j = 1, 2, 3$, the mask-generation of $\mathrm{EAX}^+_j$ satisfies the above 5 conditions.*

Here is the rough proof sketch. For $\mathrm{EAX}^+_1$, the first and second conditions hold true, as $X$ and $X \oplus X'$ (for $X \neq X'$) can be written as an element of $\mathrm{GF}(2^n)$, $\mathtt{c} \cdot L$, with a non-zero constant $\mathtt{c}$. For the third to fifth conditions, we observe that $A(0) \oplus A(1) = 2^2 L \oplus L$, $A(0) \oplus A(2) = 2^3 L \oplus 2^2 L \oplus 2^L \oplus L$, and $A(1) \oplus A(2) = 2^3 L \oplus 2L$. Any of these sums is not included in $\{D, Q, D \oplus Q\}$, which is $\{2L, 2^2 L, 2^2 L \oplus 2L\}$. This shows that the third to fifth conditions are satisfied.

For $\mathrm{EAX}^+_2$ and $\mathrm{EAX}^+_3$, each mask generation function can be defined as a matrix-vector multiplication with a binary $4 \times 4$ matrix, where each element is in $\mathrm{GF}(2^{n/4})$. For instance, the sequence of $g_{A(0)}, g_{A(1)}, g_{A(2)}, g_D, g_Q$ of $\mathrm{EAX}^+_2$ is represented by the following five matrices.

$$\begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix}, \begin{pmatrix} 1&1&1&1 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \end{pmatrix}, \begin{pmatrix} 0&0&0&1 \\ 1&1&1&1 \\ 1&0&0&0 \\ 0&1&0&0 \end{pmatrix}, \begin{pmatrix} 0&0&1&0 \\ 0&0&0&1 \\ 1&1&1&1 \\ 1&0&0&0 \end{pmatrix}, \begin{pmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&1&1 \end{pmatrix}.$$

Then we observe that the five conditions are decomposed into the following 24 cases with $\epsilon = 1/2^n$.

$$\begin{cases} \Pr[A(0) = c] \le \epsilon \\ \Pr[A(1) = c] \le \epsilon \\ \Pr[A(2) = c] \le \epsilon \\ \Pr[D = c] \le \epsilon \\ \Pr[Q = c] \le \epsilon \\ \Pr[A(0) \oplus A(1) = c] \le \epsilon \\ \Pr[A(0) \oplus A(2) = c] \le \epsilon \\ \Pr[A(0) \oplus D = c] \le \epsilon \end{cases} \begin{cases} \Pr[A(0) \oplus Q = c] \le \epsilon \\ \Pr[A(1) \oplus A(2) = c] \le \epsilon \\ \Pr[A(1) \oplus D = c] \le \epsilon \\ \Pr[A(1) \oplus Q = c] \le \epsilon \\ \Pr[A(2) \oplus D = c] \le \epsilon \\ \Pr[A(2) \oplus Q = c] \le \epsilon \\ \Pr[D \oplus Q = c] \le \epsilon \\ \Pr[A(0) \oplus A(1) \oplus D = c] \le \epsilon \end{cases} \begin{cases} \Pr[A(0) \oplus A(2) \oplus D = c] \le \epsilon \\ \Pr[A(1) \oplus A(2) \oplus D = c] \le \epsilon \\ \Pr[A(0) \oplus A(1) \oplus Q = c] \le \epsilon \\ \Pr[A(0) \oplus A(2) \oplus Q = c] \le \epsilon \\ \Pr[A(1) \oplus A(2) \oplus Q = c] \le \epsilon \\ \Pr[A(0) \oplus A(1) \oplus D \oplus Q = c] \le \epsilon \\ \Pr[A(0) \oplus A(2) \oplus D \oplus Q = c] \le \epsilon \\ \Pr[A(1) \oplus A(2) \oplus D \oplus Q = c] \le \epsilon \end{cases}$$

We can verify Proposition 3 by computing the rank (modulo 2) of the sums of the matrices, and seeing that the rank is full, for all of the above 24 cases. For $\mathrm{EAX}^+_2$ and $\mathrm{EAX}^+_3$, we confirmed

[7] The proof can be generalized to the case that these probabilities are bounded by a small number not restricted to $1/2^n$.

this by software. Lemma 2, Proposition 2 and Eq. (1) can be used without a change, including function $h(*)$. Using Lemma 3, $\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}^+_j[\mathrm{Perm}(n),\tau]}(\mathcal{A})$ can be bounded as

$$\begin{aligned}\mathtt{Adv}^{\mathtt{priv}}_{\mathrm{EAX}^+_j[\mathrm{Perm}(n),\tau]}(\mathcal{A}) &\le \mathtt{Adv}^{\mathtt{cpa}}_{h(\mathbf{Q}'),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) \\ &\le \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q}',\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{3(\sigma_N+\sigma_H+\sigma_M)^2}{2^n} \\ &\le \frac{2.5\sigma^2_{\mathrm{priv}} + 6.5\sigma^2_{\mathrm{priv}} + 3\sigma^2_{\mathrm{priv}}}{2^n} + \frac{3\sigma^2_{\mathrm{priv}}}{2^n} = \frac{15\sigma^2_{\mathrm{priv}}}{2^n},\end{aligned}$$

for any $j = 1, 2, 3$, by defining appropriate adversaries $\mathcal{B}$ and $\mathcal{C}$. In a similar manner, we derive the AUTH bound as

$$\begin{aligned}\mathtt{Adv}^{\mathtt{auth}}_{\mathrm{EAX}^+_j[\mathrm{Perm}(n),\tau]}(\mathcal{A}) &\le \mathtt{Adv}^{\mathtt{cpa}}_{h(\mathbf{Q}'),h(\widetilde{\mathbf{Q}})}(\mathcal{B}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{B}) + \frac{q_v}{2^\tau} \\ &\le \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q}',\widetilde{\mathbf{Q}}}(\mathcal{C}) + \frac{3(\sigma_N+\sigma_H+\sigma_M+\sigma_{\widetilde{N}}+\sigma_{\widetilde{H}}+\sigma_{\widetilde{C}})^2}{2^n} + \frac{q_v}{2^\tau} \\ &\le \frac{2.5\sigma^2_{\mathrm{auth}} + 6.5\sigma^2_{\mathrm{auth}} + 3\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q_v}{2^\tau} \le \frac{15\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q_v}{2^\tau}\end{aligned}$$

for any $j = 1, 2, 3$.

## 6 Conclusion

In this paper, we have presented an improved authenticity bound for EAX, an authenticated encryption mode proposed by Bellare, Rogaway and Wagner. While the original bound guarantees the standard birthday-type security in the case of one verification query, we proved the birthday-type security in the case of multiple verification queries. We also showed refinements of EAX for reducing the amounts of pre-processing blockcipher calls and working memory, which will be useful for constrained devices.

**Acknowledgments.** The authors thank the anonymous reviewers for helpful comments. A part of the work by Tetsu Iwata was carried out while visiting Nanyang Technological University, Singapore.

## References

1. Bouncy Castle, `http://www.bouncycastle.org/`
2. Information technology - Security techniques - Authenticated encryption. ISO/IEC 19772:2009 (2009)
3. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive, Report 2004/309 (2004), `http://eprint.iacr.org/`
4. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation (A Two-Pass Authenticated-Encryption Scheme Optimized for Simplicity and Efficiency), `www.cs.ucdavis.edu/~rogaway/papers/eax.pdf`
5. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy and Meier [18], pp. 389–407
6. Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. In: Bellare, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1880, pp. 197–215. Springer (2000)
7. Chakraborty, D., Sarkar, P.: A general construction of tweakable block ciphers and different modes of operations. IEEE Transactions on Information Theory 54(5), 1991–2006 (2008)
8. Dai, W.: Crypto++ Library, `http://www.cryptopp.com/`
9. Gladman, B.: `http://www.gladman.me.uk/`
10. Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC. In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887, pp. 129–153. Springer (2003)
11. Iwata, T., Kurosawa, K.: Stronger Security Bounds for OMAC, TMAC, and XCBC. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 2904, pp. 402–415. Springer (2003)

12. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: Joux, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011)
13. Minematsu, K., Lucks, S., Iwata, T.: Improved Authenticity Bound of EAX, and Refinements. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec. Lecture Notes in Computer Science, vol. 8209, pp. 184–201. Springer (2013)
14. Minematsu, K., Lucks, S., Morita, H., Iwata, T.: Attacks and Security Proofs of EAX-Prime. Pre-proceedings of Fast Software Encryption 2013 (2013), full-version available at `http://eprint.iacr.org/2012/018`
15. Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)
16. Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy and Meier [18], pp. 348–359
17. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer (2006)
18. Roy, B.K., Meier, W. (eds.): Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers, Lecture Notes in Computer Science, vol. 3017. Springer (2004)
19. Zeng, G., Han, W., He, K.: High Efficiency Feedback Shift Register: $\sigma$-LFSR. Cryptology ePrint Archive, Report 2007/114 (2007), `http://eprint.iacr.org/`

## A Proof of Lemma 1

Let $\mathbf{Q}^{\mathbf{r}} = \{\mathbf{Q}_i^{\mathbf{r}}\}_{i=1,\ldots,19}$ be the set of 19 functions defined in the same way as $\mathbf{Q}$ but the internal $n$-bit URP, $\mathsf{P}$, is substituted with $n$-bit URF, $\mathsf{R}$. From the PRF/PRP switching lemma (e.g. [6]), we have

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\mathbf{Q}^{\mathbf{r}}}(\mathcal{A}) \leq (q + \sigma_{\mathrm{out}} + 3)^2/2^{n+1}, \tag{2}$$

for any adversary $\mathcal{A}$ with parameter list $(q, \sigma_{\mathrm{out}})$. Let $\mathbf{R} = \{\mathbf{R}_i\}_{i=1,\ldots,19}$ be defined in the same way as $\widetilde{\mathbf{Q}}$, except that $\mathbf{R}_i$ for $i = 1, 2, 7, 8, 13, 14$ are independent $n$-bit URFs. That is, each $\mathbf{R}_i$ is compatible to $\mathbf{Q}_i$ and outputs are completely random. We consider the advantage in distinguishing between $\mathbf{Q}$ and $\mathbf{R}$. Then, let $\mathrm{mask}(i, L, L', L'', \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ be the input masking value used by $\mathbf{Q}_i$. The value of $\mathrm{mask}(i, L, L', L'', \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ for each $i$ is defined as follows.

$$\mathrm{mask}(i, L, L', L'', \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3) = \begin{cases} L & \text{if } i = 1 \\ \mathtt{Rnd}_1 & \text{if } i = 2 \\ 2L \oplus \mathtt{Rnd}_1 & \text{if } i = 3 \\ 4L \oplus \mathtt{Rnd}_1 & \text{if } i = 4 \\ L \oplus 2L & \text{if } i = 5 \\ L \oplus 4L & \text{if } i = 6 \\ L' & \text{if } i = 7 \\ \mathtt{Rnd}_2 & \text{if } i = 8 \\ 2L \oplus \mathtt{Rnd}_2 & \text{if } i = 9 \\ 4L \oplus \mathtt{Rnd}_2 & \text{if } i = 10 \\ L' \oplus 2L & \text{if } i = 11 \\ L' \oplus 4L & \text{if } i = 12 \\ L'' & \text{if } i = 13 \\ \mathtt{Rnd}_3 & \text{if } i = 14 \\ 2L \oplus \mathtt{Rnd}_3 & \text{if } i = 15 \\ 4L \oplus \mathtt{Rnd}_3 & \text{if } i = 16 \\ L'' \oplus 2L & \text{if } i = 17 \\ L'' \oplus 4L & \text{if } i = 18 \\ 2L & \text{if } i = 19 \end{cases} \tag{3}$$

Similarly let $\mathrm{omask}(t, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ be the outer masking value, defined as $\mathtt{Rnd}_1$ if $t \in \{1,2\}$ and $\mathtt{Rnd}_2$ if $t \in \{7,8\}$ and $\mathtt{Rnd}_3$ if $t \in \{13,14\}$, and otherwise $0^n$, that is,

$$\mathrm{omask}(i, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3) = \begin{cases} \mathtt{Rnd}_1 & \text{if } i \in \{1,2\} \\ \mathtt{Rnd}_2 & \text{if } i \in \{7,8\} \\ \mathtt{Rnd}_3 & \text{if } i \in \{13,14\} \\ 0^n & \text{otherwise} \end{cases}$$

We may abbreviate $\mathrm{mask}(i, L, L', L'', \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ to $\mathrm{mask}(i)$, and $\mathrm{omask}(j, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ to $\mathrm{omask}(j)$. Here, $\mathsf{P}(\mathrm{mask}(i) \oplus x) \oplus \mathrm{omask}(i)$ corresponds to $\mathbf{Q}_i(x)$ when $i \neq 3,4,5,6$ and $\mathrm{msb}_n(\mathbf{Q}_i(x,d))$ when $i = 3,4,5,6$. From the property of Galois field it is easy to see that

$$\max_{1 \le i < j \le 19, \delta \in \{0,1\}^n} \Pr[\mathrm{mask}(i) \oplus \mathrm{mask}(j) = \delta] \le 1/2^n \tag{4}$$

$$\max_{1 \le i \le 19, \delta \in \{0,1\}^n} \Pr[\mathrm{mask}(i) = \delta] \le 1/2^n \tag{5}$$

where both probabilities are defined by the independent uniform samplings of $L$, $L'$, $L''$, and $\mathtt{Rnd}_i$ for $i = 1,2,3$.

For any adversary querying $\mathbf{Q}^{\mathbf{r}}$ or $\mathbf{R}$, let $(t_i, X_i, d_i)$ be the $i$-th query. Without loss of generality, we assume $d_i$ is fixed to 0 whenever $t_i \notin \{3,4,5,6\}$, and all queries are distinct, i.e. $(t_i, X_i, d_i) \neq (t_j, X_j, d_j)$ for any $1 \le i < j \le q$, and when $t_i = 19$, $X_i$ is fixed to $[1]_n$ or $[2]_n$. For query $(t, X, d)$, we define $XE = X \oplus \mathrm{mask}(t)$ which is an actual input to the underlying random function when $\mathbf{Q}^{\mathbf{r}}$ is queried.

Fig. 6 defines two games, Game$\mathbf{Q}^{\mathbf{r}}$ and Game$\mathbf{R}$, and it is easy to observe that Game$\mathbf{Q}^{\mathbf{r}}$ perfectly simulates $\mathbf{Q}^{\mathbf{r}}$. Note that Game$\mathbf{R}$ behaves identically to $\mathbf{R}$, as $Y$ is $V \oplus \mathrm{omask}(t, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$ and $V$ is uniform and independent of $\mathtt{Rnd}_1, \mathtt{Rnd}_2$. Because a collision in $(t, X, d)$ is not allowed the output of Game$\mathbf{R}$ is always independent and uniformly random. We define the flag *bad* and set it when two inputs with input maskings collide. Then both games are identical until *bad* gets set to `true`, thus $\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q}^{\mathbf{r}},\mathbf{R}}(\mathcal{A})$ is bounded by

$$\Pr[\mathcal{A}^{\mathrm{Game}\mathbf{Q}^{\mathbf{r}}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathrm{Game}\mathbf{R}} \Rightarrow 1] \le \Pr[\mathcal{A}^{\mathrm{Game}\mathbf{R}} \text{ sets } \mathit{bad}]. \tag{6}$$

That is, what we need is to bound the last probability.

We first focus on *bad* at line 13. The probability of this event is not increased by an adaptive choice of queries, since outputs are completely random and independent of $XE$ for both games until *bad* sets. The existence of $\mathrm{omask}(t)$ in the output does not help, since it is XORed to a perfectly random value. Thus we fix all queries and measure the probability of *bad*.

Let us assume *bad* first occurs at line 13 with the $i$-th query, $(t_i, X_i, d_i)$. We define $XE_i$ and $Y_{i,h} \stackrel{\mathrm{def}}{=} Y_i + h$ as the corresponding internal variables appeared in the $i$-th run of the game (where the latter only appears when $t_i \in \{3,4,5,6\}$ and $d_i \ge 1$). We must have one of the three sub-events,

- $XE_i = XE_j$ for $i \le q$, $j < i$, or
- $XE_i \in \{[0]_n, [1]_n, [2]_n\}$ for $i \le q$ or
- $XE_i = Y_j + h$ for $i \le q$ and $j < i$ and $0 \le h \le d_j - 1$.

From Eqs. (4) and (5) the first and the last sub-events occur with probability at most $1/2^n$, and the middle one with probability at most $3/2^n$. We next focus on *bad* at line 18, which implies the occurrence of one of the three sub-events (when $t_i, t_j \in \{3,4,5,6\}$),

– $Y_i + h = XE_j$ for some $i \le q$, $j \le i$, $0 \le h \le d_i - 1$ or
– $Y_i + h \in \{[0]_n, [1]_n, [2]_n\}$ for some $i \le q$ and $0 \le h \le d_i - 1$, or
– $Y_i + h = Y_j + h'$ for some $(i, h) \ne (j, h')$, $i, j \le q$ and $0 \le h \le d_i - 1$ and $0 \le h' \le d_j - 1$.

As $Y_i$ is random and independent of all previous variables, we have

$$\Pr[Y_i + h = X_j] = \max_{\delta} \Pr[Y_i = \delta - h] = 1/2^n.$$

Now we have $\Pr[Y_i + h \in \{[0]_n, [1]_n, [2]_n\}] \le 3/2^n$ and $\Pr[Y_i + h = Y_j + h'] \le 1/2^n$. By counting the number of sub-events, we have

$$\begin{aligned}\Pr[\mathcal{A}^{\mathrm{Game}\mathbf{R}} \text{ sets } bad] \le & \underbrace{\binom{q}{2}\frac{1}{2^n}}_{XE_i = XE_j} + \underbrace{\frac{3q}{2^n}}_{XE_i \in \{[0]_n,[1]_n,[2]_n\}} + \underbrace{\frac{\sigma_{\mathrm{out}} q}{2^n}}_{\substack{XE_i = Y_j + h \\ \text{for both } i < j \text{ and } j \le i}} \\ & + \underbrace{\frac{3\sigma_{\mathrm{out}}}{2^n}}_{Y_i + h \in \{[0]_n,[1]_n,[2]_n\}} + \underbrace{\binom{\sigma_{\mathrm{out}}}{2}\frac{1}{2^n}}_{Y_i + h = Y_j + h'} \\ & \le \frac{(0.5q^2 + 6\sigma_{\mathrm{out}} q + 0.5\sigma_{\mathrm{out}}^2)}{2^n}. \end{aligned} \tag{7}$$

We also need to evaluate the distinguishing advantage of $\mathbf{R}$ and $\widetilde{\mathbf{Q}}$. The difference between them is that $\mathbf{R}$ uses $n$-bit URFs when $t$ is in $\{1, 2, 7, 8, 13, 14\}$ while $\widetilde{\mathbf{Q}}$ uses $n$-bit URPs. For other values of $t$ their behaviors are identical and independent of the responses obtained when $t = 1, 2, 7, 8, 13, 14$. Combining this observation and the PRP/PRF switching lemma we have

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{R},\widetilde{\mathbf{Q}}}(\mathcal{A}) \le q^2/2^{n+1}. \tag{8}$$

Combining Eqs. (2), (8), (7), and (6), we have

$$\begin{aligned}\mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\widetilde{\mathbf{Q}}}(\mathcal{A}) &\le \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q},\mathbf{Q}^{\mathtt{r}}}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{Q}^{\mathtt{r}},\mathbf{R}}(\mathcal{A}) + \mathtt{Adv}^{\mathtt{cpa}}_{\mathbf{R},\widetilde{\mathbf{Q}}}(\mathcal{A}) \\ &\le \frac{(q + \sigma_{\mathrm{out}} + 3)^2}{2^{n+1}} + \frac{0.5q^2 + 6\sigma_{\mathrm{out}} q + 0.5\sigma_{\mathrm{out}}^2}{2^n} + \frac{q^2}{2^{n+1}} \\ &\le \frac{4.5q^2 + 10\sigma_{\mathrm{out}} q + \sigma_{\mathrm{out}}^2 + 4.5}{2^n},\end{aligned}$$

which concludes the proof.

## B Proof of Lemma 2

First we define $\mathrm{CBC}_{G,G'} : (\{0,1\}^n)^{>0} \to \{0,1\}^n$ for any $n$-bit (keyed) permutations, $G$ and $G'$. It is defined as

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

where $\mathrm{CBC}_{G'}$ is the standard CBC-MAC using permutation $G'$. For convenience we also define $\mathrm{CBC}^{\oplus}_{G,G'} : (\{0,1\}^n)^{\ge 2} \to \{0,1\}^n$ as $\mathrm{CBC}^{\oplus}_{G,G'}(X[1]\| \dots \|X[m]) = \mathrm{CBC}_{G,G'}(X[1]\| \dots \|X[m-1]) \oplus X[m]$ for $m \ge 2$, i.e. $\mathrm{CBC}_{G,G'}$ without the final application of $G'$. Let us focus on the case $t = 1$. Then $\mathbb{CBC}$ can be seen as an information-theoretic variant of Carter-Wegman MAC, that is, $\mathbb{CBC}^{(1)} : \{0,1\}^* \to \{0,1\}^n$ is a composition of two functions, where the first one

**Initialization**
00 $L \leftarrow \rho([0]_n) \xleftarrow{\$} \{0,1\}^n$, $L' \leftarrow \rho([1]_n) \xleftarrow{\$} \{0,1\}^n$, $L'' \leftarrow \rho([2]_n) \xleftarrow{\$} \{0,1\}^n$
01 $\mathtt{Rnd}_1 \xleftarrow{\$} \{0,1\}^n$, $\mathtt{Rnd}_2 \xleftarrow{\$} \{0,1\}^n$, $\mathtt{Rnd}_3 \xleftarrow{\$} \{0,1\}^n$
**On query** $(t, X, d) \in \{0,1,2\} \times \{0,1\}^* \times \mathbb{N}$
10 $XE \leftarrow \mathrm{mask}(t, L, L', L'', \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3) \oplus X$
11 $Y \xleftarrow{\$} \{0,1\}^n$
12 $V \leftarrow Y \oplus \mathrm{omask}(t, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)$
13 **if** $XE \in \mathrm{Dom}(\rho)$ **then** bad $\leftarrow$ `true`, $\boxed{V \leftarrow \rho(XE), Y \leftarrow V \oplus \mathrm{omask}(t, \mathtt{Rnd}_1, \mathtt{Rnd}_2, \mathtt{Rnd}_3)}$
14 **else** $\rho(XE) \leftarrow V$
15 **if** $t \notin \{3,4,5,6\}$ **or** $t \in \{3,4,5,6\}$ **and** $d = 0$ **then return** $Y$
16 **for** $i = 0$ **to** $d-1$ **do**
17 $S[i+1] \xleftarrow{\$} \{0,1\}^n$
18 **if** $V + i \in \mathrm{Dom}(\rho)$ **then** bad $\leftarrow$ `true`, $\boxed{S[i+1] \leftarrow \rho(V+i)}$
19 **else** $\rho(V+i) \leftarrow S[i+1]$
20 **return** $Y \| S[1] \| S[2] \| \cdots \| S[d]$

**Fig. 6.** Game$\mathbf{Q^r}$ contains the boxed arguments, while Game$\mathbf{R}$ does not.

(called hashing) takes an input to hash it into $n$ bits, and the second one (called finalizing) takes that hashed value to compute the $n$-bit output. The hashing function applies $\mathrm{CBC}_{\mathsf{P}_7,\mathsf{P}_8}$ with $\mathtt{bp}(*)$ for the final input block, or just applies $\mathtt{bp}(*)$ to the input itself if the input is at most $n$ bits. The finalizing function applies one of the 4 independent $n$-bit URFs, $\mathsf{R}_9$ to $\mathsf{R}_{12}$, depending on the input length, or $\mathsf{R}_{19}$ if input is empty. For a pair of distinct inputs to $\mathbb{CBC}^{(1)}$, let $X = X[1]\|X[2]\|\ldots\|X[m]$ and $X' = X'[1]\|\|X'[2]\|\ldots\|X'[m']$, let $Z$ and $Z'$ be the corresponding hash values and $\mathsf{R}_\kappa$ and $\mathsf{R}_{\kappa'}$ be the URFs for the finalizing function. From the definition of $\mathbb{CBC}$ a simultaneous collision $(Z,\kappa) = (Z',\kappa')$ occurs only if both $X$ and $X'$ are more than $n$ bits (and non-empty), and therefore $(Z,\kappa) = (Z',\kappa')$ implies

$$\mathrm{CBC}^{\oplus}_{\mathsf{P}_7,\mathsf{P}_8}(X[1]\|\ldots\|X[m-1]\|\mathtt{bp}(X[m])) = \mathrm{CBC}^{\oplus}_{\mathsf{P}_7,\mathsf{P}_8}(X'[1]\|\ldots\|X'[m-1]\|\mathtt{bp}(X'[m']))$$

satisfying $X[1]\|\ldots\|X[m-1]\|\mathtt{bp}(X[m]) \neq X'[1]\|\ldots\|X'[m-1]\|\mathtt{bp}(X'[m'])$ and $m, m' \geq 2$.

For a keyed function $F : \mathcal{X} \to \mathcal{Y}$, let $\mathrm{Coll}_F(q,\sigma)$ be the maximum collision probability of $F$'s outputs when accessed via $q$ *non-adaptive* chosen-plaintext queries with total $\sigma$ $n$-bit blocks. Now, from the above observation and (a slight generalized version of) Lemma 2 of [6],we have

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC}^{(1)},\mathbb{RND}^{(1)}}(\mathcal{A}) \leq \mathrm{Coll}_{\mathrm{CBC}^{\oplus}_{\mathsf{P}_7,\mathsf{P}_8}}(q,\sigma_{\mathrm{in}}),$$

for any (possibly adaptive) $\mathcal{A}$ that has parameter list $(q,\sigma_{\mathrm{in}},\sigma_{\mathrm{out}})$.

Moreover, Lemma 4.2 of Iwata and Kurosawa [11] (called MOMAC-E Collision Bound) proves

$$\mathrm{Coll}_{\mathrm{CBC}^{\oplus}_{\mathsf{P}_7,\mathsf{P}_8}}(q,\sigma_{\mathrm{in}}) \leq \frac{(\sigma_{\mathrm{in}}-q)^2}{2^n}.$$

Thus we have

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC}^{(1)},\mathbb{RND}^{(1)}}(\mathcal{A}) \leq \frac{(\sigma_{\mathrm{in}}-q)^2}{2^n}. \tag{9}$$

Similarly, we have

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC}^{(0)},\mathbb{RND}^{(0)}}(\mathcal{A}) \leq \frac{(\sigma_{\mathrm{in}}-q)^2}{2^n}, \text{ and} \tag{10}$$

$$\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC}^{(2)},\mathbb{RND}^{(2)}}(\mathcal{A}) \leq \frac{(\sigma_{\mathrm{in}}-q)^2}{2^n}, \tag{11}$$

as $\mathbb{CBC}^{(0)}$ and $\mathbb{CBC}^{(2)}$ have the same structure as $\mathbb{CBC}^{(1)}$. For $\mathbb{CBC}^{(0)}$ having input $X$, it applies $\mathsf{R}_5^{d+1}$ or $\mathsf{R}_6^{d+1}$ when $|X| \leq n$, and when $|X| > n$ it uses $\mathrm{CBC}^{\oplus}_{\mathsf{P}_1,\mathsf{P}_2}$ for hashing and $\mathsf{R}_3^{d+1}$ or $\mathsf{R}_4^{d+1}$

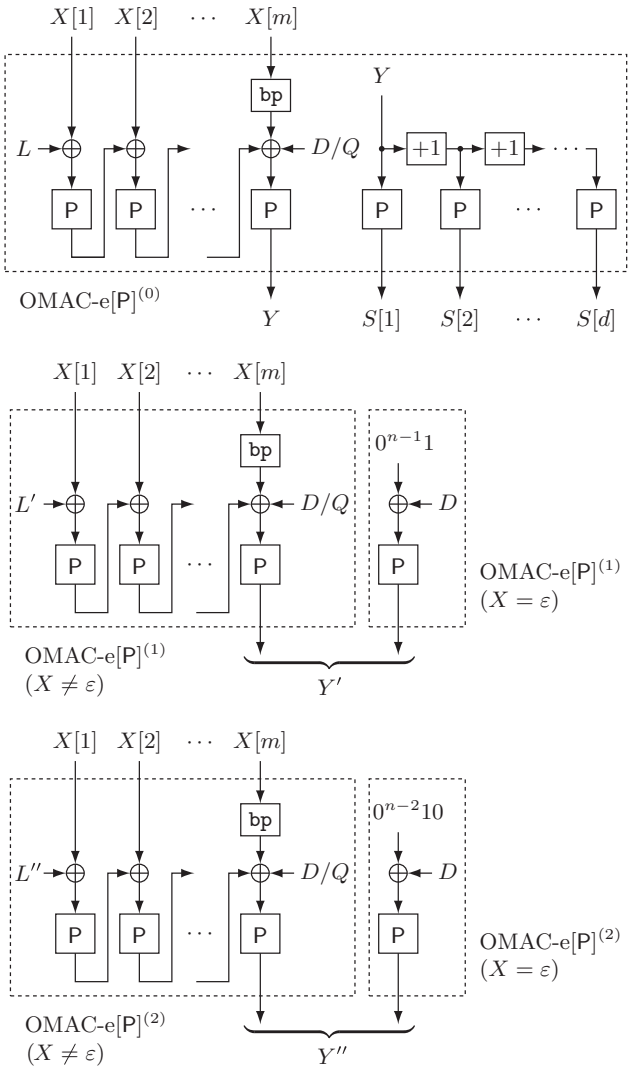


**Fig. 7.** OMAC-extension in the proofs of Theorems 3 and 4.

for finalization, where the input length determines which URF is to be used. The finalization is done by URF of variable-output length, however this apparently does not gain the advantage in distinguishing it from $\mathbb{RND}^{(0)}$. For $\mathbb{CBC}^{(2)}$, the difference from $\mathbb{CBC}^{(1)}$ is that it can accept the empty string, which is processed by an independent URF ($\mathsf{R}_{19}$).

Note that the internal URP/URFs of $\mathbb{CBC}^{(0)}$, $\mathbb{CBC}^{(1)}$, and $\mathbb{CBC}^{(2)}$ have no overlap, thus their probability spaces are independent. Here, an exception is $\mathsf{R}_{19}$ for case $t = 1$ and $t = 2$, however in these cases $\mathsf{R}_{19}$ takes distinct inputs, hence the outputs are independent and random. Therefore, using the hybrid argument and Eqs. (9) to (11) we have

$$\begin{aligned}
\mathtt{Adv}^{\mathtt{cpa}}_{\mathbb{CBC},\mathbb{RND}}(\mathcal{A}) &= \mathtt{Adv}^{\mathtt{cpa}}_{(\mathbb{CBC}^{(0)},\mathbb{CBC}^{(1)},\mathbb{CBC}^{(2)}),(\mathbb{RND}^{(0)},\mathbb{RND}^{(1)},\mathbb{RND}^{(2)})}(\mathcal{A}) \\
&\le \mathtt{Adv}^{\mathtt{cpa}}_{(\mathbb{CBC}^{(0)},\mathbb{CBC}^{(1)},\mathbb{CBC}^{(2)}),(\mathbb{RND}^{(0)},\mathbb{CBC}^{(1)},\mathbb{CBC}^{(2)})}(\mathcal{A}) \\
&\quad + \mathtt{Adv}^{\mathtt{cpa}}_{(\mathbb{RND}^{(0)},\mathbb{CBC}^{(1)},\mathbb{CBC}^{(2)}),(\mathbb{RND}^{(0)},\mathbb{RND}^{(1)},\mathbb{CBC}^{(2)})}(\mathcal{A}) \\
&\quad + \mathtt{Adv}^{\mathtt{cpa}}_{(\mathbb{RND}^{(0)},\mathbb{RND}^{(1)},\mathbb{CBC}^{(2)}),(\mathbb{RND}^{(0)},\mathbb{RND}^{(1)},\mathbb{RND}^{(2)})}(\mathcal{A}) \le \frac{3(\sigma_{\mathrm{in}} - q)^2}{2^n} \le \frac{3\sigma_{\mathrm{in}}^2}{2^n}.
\end{aligned}$$

This completes the proof.

## C Proof of Eq. (1)

For $f_e$ and $f_d$ appearing at Proposition 1, we observe that, if $f_e(\mathbb{RND})$ given $(N, H, M)$ outputs $(C, T)$, then $T$ can be written as

$$\begin{aligned} T &= \mathrm{msb}_\tau(\mathrm{msb}_n(\mathbb{RND}^{(0)}(N, |M|_n)) \oplus \mathbb{RND}^{(1)}(H) \oplus \mathbb{RND}^{(2)}(C)) \\ &= \mathrm{msb}_\tau(\mathbb{RND}^{(0)}(N, |C|_n)) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(1)}(H) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(2)}(C)). \end{aligned}$$

Note that $C$ can be $\varepsilon$ and $\mathbb{RND}^{(2)}$ treats $\varepsilon$ as an input (i.e. outputs random $n$ bits).

We first consider the case $q_v = 1$. Without loss of generality we assume that the single decryption query, $(\widetilde{N}, \widetilde{H}, \widetilde{C}, \widetilde{T})$, is issued after obtaining $q$ pairs of encryption queries and answers, $((N_1, H_1, M_1, C_1, T_1), \ldots, (N_q, H_q, M_q, C_q, T_q))$. Now, the success probability of a forgery is $1/2^\tau$ when $\widetilde{N} \neq N_i$ for all $i = 1, \ldots, q$. Otherwise, we have a *unique* $j \in \{1, \ldots, q\}$ such that $\widetilde{N} = N_j$ and the successful forgery corresponds to the event that

$$\begin{aligned} &[\widetilde{T} = \mathrm{msb}_\tau(\mathbb{RND}^{(0)}(\widetilde{N}, |\widetilde{C}|_n)) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(1)}(\widetilde{H})) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(2)}(\widetilde{C}))] \\ &\Leftrightarrow [\widetilde{T} = \mathrm{msb}_\tau(\mathbb{RND}^{(0)}(N_j, |C_j|_n)) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(1)}(\widetilde{H})) \oplus \mathrm{msb}_\tau(\mathbb{RND}^{(2)}(\widetilde{C}))] \\ &\Leftrightarrow [\widetilde{T} \oplus T_j = \mathrm{msb}_\tau(\mathbb{RND}^{(1)}(H_j) \oplus \mathbb{RND}^{(2)}(C_j) \oplus \mathbb{RND}^{(1)}(\widetilde{H}) \oplus \mathbb{RND}^{(2)}(\widetilde{C}))]. \end{aligned} \tag{12}$$

As $(\widetilde{H}, \widetilde{C}, \widetilde{T}) \neq (H_j, C_j, T_j)$ holds true, the probability of Eq. (12) is at most $1/2^\tau$. Here, note that the choice of $\widetilde{H}$ and $\widetilde{C}$ (e.g. choosing $\widetilde{C} = C_{j'}$ for some $j' \neq j$) and the distribution of $H_1, \ldots, H_q$ and $C_1, \ldots, C_q$, which can contain collisions on $C_i$s or $H_i$s, do not contribute to gaining the probability since the transcript obtained by the encryption queries completely hides the information on $\mathbb{RND}^{(1)}(*)$ and $\mathbb{RND}^{(2)}(*)$ no matter what $H_1, \ldots, H_q$ and $C_1, \ldots, C_q$ are, as $N_1, \ldots, N_q$ are unique. This implies that $\mathtt{Adv}^{\mathtt{auth}}_{\mathbb{EAX}}(\mathcal{A}) = 1/2^\tau$, when $q_v = 1$.

From Theorem B.2 of [3], any AE scheme having the maximum forgery probability being $\epsilon$ when $q_v = 1$ has the maximum forgery probability $\epsilon \cdot q_v$ when $q_v \geq 1$. Combining this with the above analysis of the case $q_v = 1$, the proof is completed.