Cryptanalysis of EAXprime

Kazuhiko Minematsu[1], Stefan Lucks[2], Hiraku Morita[3], and Tetsu Iwata[4]

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

Abstract. EAX' (EAXprime) is an authenticated encryption (AE) specified by ANSI C12.22 as a standard security function used for a smart grid. EAX' is based on EAX, a provably secure AE proposed by Bellare, Rogaway, and Wagner. This paper presents simple and efficient forgery attacks, distinguishers, and message recovery attacks against EAX' using single-block cleartext and plaintext.

1 Introduction

ANSI C12.22-2008 [1] specifies a blockcipher mode for authenticated encryption (AE) as the standard security function for Smart Grids. It is called EAX' (or EAXprime). As its name suggests, EAX' is based on EAX proposed by Bellare, Rogaway, and Wagner at FSE 2004 [2]. It was submitted to NIST [5] and NIST is planning to include EAX' in an additional part of the 800-38 series of Special Publications.

Though EAX' is similar to EAX, to the best of our knowledge, its formal security analysis is not known to date. In this paper, we investigate the security of EAX' as a general-purpose authenticated encryption, and provide attacks against EAX' for both authenticity and confidentiality. 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 message 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 the need to observe any valid plaintext/ciphertext pair at all. Our results imply that, while the original EAX has a proof of security, the security of EAX' cannot be proved as a general-purpose authenticated encryption.

We remark that the above attacks work only if the inputs to EAX' are quite short. The encryption of EAX' accepts two elements, N and P, where N is called a "cleartext" (corresponds to a combination of a nonce and a header), and P is called a "plaintext". Our forgery attacks and distinguishers are only applicable

when neither the length of N nor the length of P exceeds the block size of the underlying blockcipher (i.e., 128 bits for the AES). Our message recovery attacks are applicable when the length of N does not exceed the block size, even if the length of P exceeds the block size. However, at most the first block of P can be recovered.

Therefore, though we demonstrate the total insecurity of EAX' when N and P are allowed to be 128 bits or shorter, we do not know if our attacks work to the protocols strictly following ANSI C12.22.

2 Definition of EAX'

First, let $\{0,1\}^*$ be the set of all binary strings, including the null string NULL. The bit length of $x \in \{0,1\}^*$ is denoted by $\mathrm{size}(x)$. Here $\mathrm{size}(\mathrm{NULL}) = 0$. A concatenation of $x, y \in \{0,1\}^*$ is written as $x\|y$. A sequence of a zeros (ones) is denoted by 0^a (1^a). The first i bits of x is written as $\mathrm{MSB}_i(x)$. For $x, y \in \{0,1\}^*$ with $\mathrm{size}(x) \geq \mathrm{size}(y)$, let $x \oplus_{\mathrm{end}} y = x \oplus \big((0^{\mathrm{size}(x)-\mathrm{size}(y)})\|y\big)$.

We describe EAX'. Basically our description is almost the same as the ANSI documents [1, 5], with some minor notational differences. EAX' is a mode of operation based on an n-bit blockcipher, E_K, where we assume that (n, E_K) is $(128, \text{AES-128})$ as an example, while other choice is possible [5]. Formally, the encryption function of EAX' accepts a cleartext, $N \in \{0,1\}^*$, a plaintext, $P \in \{0,1\}^*$, and a secret key K to produce the ciphertext, $C \in \{0,1\}^*$, with $\mathrm{size}(C) = \mathrm{size}(P)$ and $T \in \{0,1\}^{32}$. The decryption function, which we also call the verification function, accepts N, C, T, and K to generate the decrypted plaintext $\widetilde{P}$ if (N, C, T) is valid, and the flag `INVALID` if invalid. Cleartext N contains information that needs only authenticity and the ANSI document requires that N must be unique for all encryptions using the same key. Hence N can be seen as a combination of a header and an initial vector (IV) in the standard context for authenticated encryption (e.g., see [2]). In ANSI C12.22, the uniqueness of N is guaranteed by including time information with a specific format. The plaintext P can be the null string NULL, and in this case EAX' works as a message authentication code for the corresponding N.

We first define the following functions. Here $N, S \in \{0,1\}^*$ and $t, D, Q \in \{0,1\}^n$.

Pad function:

1: **function** PAD(S, D, Q)
2: **if** $\mathrm{size}(S) > 0$ **and** $\mathrm{size}(S) \bmod n = 0$ **then return** $S \oplus_{\mathrm{end}} D$
3: **else return** $(S\|10^{n-1-(\mathrm{size}(S) \bmod n)}) \oplus_{\mathrm{end}} Q$
4: **end if**
5: **end function**

CBC' function:

1: **function** CBC'$_K(t, S)$
2: Let $S_1, \ldots, S_m \leftarrow S$ where $\mathrm{size}(S_i) = n$
3: $C_0 \leftarrow t$

4: **for** $i \leftarrow 1$ **to** m **do** $C_i \leftarrow E_K(S_i \oplus C_{i-1})$
5: **end for**
6: **return** C_m
7: **end function**

CMAC' function:

1: **function** CMAC'$_K(t, S, D, Q)$
2: **return** CBC'$_K(t, \textsc{Pad}(S, D, Q))$
3: **end function**

CTR' function:

1: **function** CTR'$_K(N, S)$
2: $m \leftarrow \lceil \text{size}(S)/n \rceil$
3: $\text{Ctr} \leftarrow N \wedge (1^{n-32} \| 01^{15} \| 01^{15})$
4: $\text{Keystream} \leftarrow E_K(\text{Ctr}) \| E_K(\text{Ctr}+1) \| \cdots \| E_K(\text{Ctr}+m-1)$
5: **return** $S \oplus \text{MSB}_{\text{size}(S)}(\text{Keystream})$
6: **end function**

Using these functions, the encryption of EAX' is defined as follows. Decryption of EAX' is straightforward and omitted here.

1: **function** EAX'.encrypt$_K(N, P)$
2: $D \leftarrow \text{dbl}(E_K(0^n))$
3: $Q \leftarrow \text{dbl}(D)$
4: $\underline{N} \leftarrow \text{CMAC'}_K(D, N, D, Q)$
5: **if** $\text{size}(P) = 0$ **then** $C \leftarrow \text{NULL}$
6: **else**
7: $C \leftarrow \text{CTR'}_K(\underline{N}, P)$
8: **end if**
9: $T \leftarrow \underline{N} \oplus \text{CMAC'}_K(Q, C, D, Q)$
10: $T \leftarrow \text{MSB}_{32}(T)$
11: **return** (C, T)
12: **end function**

Here, $\text{dbl}(x)$ with $\text{size}(x) = n$ denotes the "doubling" operation over $\text{GF}(2^n)$, as defined by [5, 7].

In the above definition, we fixed an apparent error in line 72 of the definition of EAX'.encrypt$_K$ in [1, 5]. Some editorial errors of [5] were also pointed out by [6].

3 Forgeries

Chosen Message Existential Forgeries

The standard setting for cryptographic forgery attacks is as follows:

– The adversary asks $q \in \{0, 1, 2, \ldots\}$ queries (N_1, P_1), $\ldots$, (N_q, P_q) to the encryption oracle, receiving the corresponding ciphertext/tag pairs (C_i, T_i).
– The adversary queries the verification oracle for some (N, C, T) where no $i \in \{1, \ldots, q\}$ exists with $N = N_i$ and $C = C_i$ and $T = T_i$.
– The adversary succeeds if the (N, C, T) passes the verification.

For $x, y \in \{0,1\}^*$ with $\mathrm{size}(x) \geq \mathrm{size}(y)$, let $x \oplus_{\mathrm{start}} y = x \oplus \big(y \| (0^{\mathrm{size}(x)-\mathrm{size}(y)})\big)$. Then, the definition of CMAC' in the previous section conforms to that

$$\mathrm{CMAC'}_K(t, S, D, Q) = \begin{cases} \mathrm{CMAC}_K(S \oplus_{\mathrm{start}} t) & \text{if } \mathrm{size}(S) > n, \\ E_K(t \oplus Q \oplus (S \| 10^{n-1-\mathrm{size}(S)})) & \text{if } 0 \leq \mathrm{size}(S) < n, \\ E_K(t \oplus D \oplus S) & \text{if } \mathrm{size}(S) = n, \end{cases}$$

where CMAC_K is as defined by [7] (or OMAC1 of [4]) using E_K. Hence, we have

$$\mathrm{CMAC'}_K(D, S, D, Q) = E_K(S) \text{ when } \mathrm{size}(S) = n$$

and

$$\mathrm{CMAC'}_K(Q, S, D, Q) = E_K(S \| 10^{n-1-\mathrm{size}(S)}) \text{ when } 0 \leq \mathrm{size}(S) < n.$$

The above observation immediately implies the following forgery attacks:

Forgery attack 1 ($\mathrm{size}(N) = n$ and $\mathrm{size}(C) < n$)**.**

1. Prepare (N, C) that satisfies $\mathrm{size}(N) = n$ and $\mathrm{size}(C) < n$ with $C \| 10^{n-1-\mathrm{size}(C)} = N$, and let $T = 0^{32}$.
2. Query (N, C, T) to the verification oracle.

This attack always succeeds as the "valid" tag for (N, C) is

$$\mathrm{MSB}_{32}(E_K(N) \oplus E_K(C \| 10^{n-1-\mathrm{size}(C)})) = 0^{32}.$$

Forgery attack 2 ($\mathrm{size}(N) < n$ and $\mathrm{size}(C) = n$)**.**

1. Prepare (N, C) that satisfies $\mathrm{size}(N) < n$ and $\mathrm{size}(C) = n$ with $N \| 10^{n-1-\mathrm{size}(N)} = C$, and let $T = 0^{32}$.
2. Query (N, C, T) to the verification oracle.

The attack is again successful as the valid tag for (N, C) is $\mathrm{MSB}_{32}(E_K(D \oplus Q \oplus N \| 10^{n-1-\mathrm{size}(N)}) \oplus E_K(Q \oplus D \oplus C)) = 0^{32}$. These attacks use only one forgery attempt and no encryption query.

By using one encryption query the forgery attack is possible even when $\mathrm{size}(N) = n$ and $\mathrm{size}(C) = n$:

Forgery attack 3.

1. Query (N, P) with $\mathrm{size}(N) = \mathrm{size}(P) = n$ and $N \neq 0^n$ to the encryption oracle[5].
2. Obtain (C, T) (where $\mathrm{size}(C) = n$) from the oracle and see if $C \neq 0^n$ (quit if $C = 0^n$) .
3. Query $(\widehat{N}, \widehat{C}, \widehat{T})$ to the verification oracle, where $\mathrm{size}(\widehat{N}) < n$, $\mathrm{size}(\widehat{C}) < n$, $\widehat{N} \| 10^{n-1-\mathrm{size}(\widehat{N})} = C$, $\widehat{C} \| 10^{n-1-\mathrm{size}(\widehat{C})} = N$, and $\widehat{T} = T$.

The above attack is almost always successful; unless $C = 0^n$ we have $T = \mathrm{MSB}_{32}(E_K(N) \oplus E_K(Q \oplus D \oplus C))$ and the valid tag for $(\widehat{N}, \widehat{C})$ is

$$\begin{aligned}&\mathrm{MSB}_{32}(E_K(D \oplus Q \oplus \widehat{N} \| 10^{n-1-\mathrm{size}(\widehat{N})}) \oplus E_K(Q \oplus Q \oplus \widehat{C} \| 10^{n-1-\mathrm{size}(\widehat{C})})) \\ &= \mathrm{MSB}_{32}(E_K(D \oplus Q \oplus C) \oplus E_K(N)),\end{aligned}$$

thus equals to T.

Forgery attack 4.

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

This is the converse of Forgery attack 3; we have $T = \mathrm{MSB}_{32}(E_K(D \oplus Q \oplus N \| 10^{n-1-\mathrm{size}(N)}) \oplus E_K(Q \oplus Q \oplus C \| 10^{n-1-\mathrm{size}(C)}))$ and the valid tag for $(\widehat{N}, \widehat{C})$ is

$$\begin{aligned}&\mathrm{MSB}_{32}(E_K(D \oplus D \oplus \widehat{N}) \oplus E_K(Q \oplus D \oplus \widehat{C})) \\ &= \mathrm{MSB}_{32}(E_K(C \| 10^{n-1-\mathrm{size}(C)}) \oplus E_K(Q \oplus D \oplus N \| 10^{n-1-\mathrm{size}(N)})) = T.\end{aligned}$$

From Existential to 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 P and the non-confidential associated data (or "cleartext") N. While the above attacks don't allow to choose the plaintext, the adversary can arbitrarily choose the cleartext N (restricted to $\mathrm{size}(N) \leq n$ and, for $\mathrm{size}(N) = n$, $N \neq 0^n$). In this sense, the forgery attacks above are *partially selective*.

[5] Here we assume the adversary is allowed to query any (N, P) as long as N is unique; that is, nonce (here N)-respecting adversary introduced by Rogaway [8].

4 Chosen Plaintext Distinguishers

The forgery attacks above were based on the idea of generating cleartexts N and ciphertexts C, such that the authentication tag T is zero, i.e., $T = 0^{32}$. To distinguish EAX' from a random encryption process one can apply a similar idea: Generate cleartexts N and plaintexts P such that the EAX' encryption will generate any ciphertext C and a zero authentication tag T.

Distinguishing attack 1.

1. Query (N, P) to the encryption oracle, where $N = 10^{n-1}$ (thus size$(N) = n$) and $P =$ NULL (thus size$(P) = 0$).
2. Obtain T from the oracle.
3. If $T = 0^{32}$ then return 1, otherwise return 0.

As EAX' returns $T = 0^{32}$ with probability 1 while the same event occurs with probability $1/2^{32}$ with a random encryption process, this enables us to easily distinguish T from random with the distinguishing advantage almost 1, using only one chosen plaintext query.

Distinguishing attack 2.

1. Fix small $1 \leq i \leq n-1$ and any $P \in \{0,1\}^i$, and query (N, P) to the encryption oracle with $N = P \| 10^{n-1-\text{size}(P)}$ (thus size$(N) = n$).
2. Obtain (C, T) from the oracle.
3. If $C = P$ and $T = 0^{32}$ then return 1, otherwise return 0.

In this case, we have $C = P$ with probability $1/2^i$ for both EAX' and a random encryption process. Given the event $C = P$, we have

$$T = \text{MSB}_{32}(E_K(N) \oplus E_K(C \| 10^{n-1-\text{size}(C)})) = 0^{32}$$

with probability 1 for EAX', while $T = 0^{32}$ occurs with probability $1/2^{32}$ for the random encryption process. Thus the distinguisher succeeds with probability $1/2^i$, which is non-negligible when i is small.

5 Chosen-Ciphertext Message Recovery Attacks

Consider a triple (N^*, C^*, T^*) of cleartext N^*, ciphertext C^* and tag T^*. The corresponding plaintext P^* 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 P^* would be trivial). The adversary receives either an error message (if the tag T doesn't fit), or the decryption P of C. This is the setting in a *chosen ciphertext attack*. Below, we focus on *message recovery attacks*, where the adversary actually finds P^*.

We describe two message recovery attacks: The first for the size$(N^*) = n$, the second for size$(N^*) < n$.

Message recovery attack 1 ($\mathrm{size}(N^*) = n$)**.**

1. Require $\mathrm{size}(N^*) = n$
2. Prepare C with $\mathrm{size}(C) < n$ and with $C \| 10^{n-1-\mathrm{size}(C)} = N^*$.
3. Query $(N^*, C, 0^{32})$ to the decryption oracle. Let P be the answer.
4. Compute the keystream $K(N^*) = C \oplus P \in \{0,1\}^{\mathrm{size}(C)}$.

Since the decryption of (N^*, C^*, T^*) uses the same keystream, we now can compute the first $\mathrm{size}(C)$ bits of P^* (or the full P^* if $\mathrm{size}(P^*) \leq \mathrm{size}(C)$). It succeeds for the same reason as Forgery attack 1 (unless $C^* \| 10^{n-1-\mathrm{size}(C^*)} = N^*$ and $T^* = 0^{32}$, in which case the decryption query in Step 3 makes the attack trivial).

Message recovery attack 2 ($\mathrm{size}(N^*) < n$)**.**

1. Require $\mathrm{size}(N^*) < n$
2. Prepare C with $\mathrm{size}(C) = n$ and $N^* \| 10^{n-1-\mathrm{size}(N^*)} = C$.
3. Query $(N^*, C, 0^{32})$ to the decryption oracle. Let P be the answer.
4. The keystream is $K(N^*) = C \oplus P \in \{0,1\}^n$.

Unless $N^* \| 10^{n-1-\mathrm{size}(N^*)} = C^*$ and $T^* = 0^{32}$, the attack succeeds for the same reason as Forgery attack 2.

6 Remarks

As our attacks demonstrate, there is a gap between the security of the original EAX and EAX', and it is not possible to prove the security of EAX' as a general-purpose authenticated encryption.

6.1 Differences between EAX' and the Original EAX

Let $\mathrm{CMAC}_K[V](*)$ be the CMAC using an n-bit blockcipher E_K with initial block mask $V \in \{0,1\}^n$. That is, for $P \in \{0,1\}^*$ we have

$$\mathrm{CMAC}_K[V](P) = \begin{cases} E_K(V \oplus P \| 10^{n-1-\mathrm{size}(P)} \oplus \mathrm{dbl}(\mathrm{dbl}(L))) & \text{if } \mathrm{size}(P) < n \\ \mathrm{CMAC}_K(P \oplus_{\mathrm{start}} V) & \text{if } \mathrm{size}(P) \geq n, \end{cases}$$

where $L = E_K(0^n)$ and CMAC_K is the standard CMAC [7] using E_K. Note that if $V = E_K(t)$ we have $\mathrm{CMAC}_K[V](P) = \mathrm{CMAC}_K(t \| P)$.

The major differences of EAX' from EAX are summarized as follows[6].

1. Role of N. Inputs to the EAX' encryption function (except for the key) consist of a "cleartext" N and a plaintext P, whereas those to the original EAX consist of a nonce N, a header H, and a plaintext P. 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.

[6] For other minor differences, see Sect. 3 of [5].

2. Tweaking method for CMAC. For the original EAX, the tag is (the first τ bits of) the xor of $\mathrm{CMAC}_K[L](N)$ and $\mathrm{CMAC}_K[W](H)$ and $\mathrm{CMAC}_K[Z](C)$ with $L = E_K(0^n)$, $W = E_K(0^{n-1}\|1)$, and $Z = E_K(0^{n-2}\|10)$. Hence it uses three tweaked variants of CMAC.
   EAX' uses two different CMAC variants. In EAX' the tag is the first 32 bits of $\mathrm{CMAC}_K[\mathrm{dbl}(L)](N) \oplus \mathrm{CMAC}_K[\mathrm{dbl}(\mathrm{dbl}(L))](C)$, where $L = E_K(0^n)$.
3. Counter mode incrementation. The original EAX uses $\mathrm{CMAC}_K[0^n](N)$ as an initial counter block for CTR mode, while the initial counter block of EAX' is $\mathrm{CMAC}_K[\mathrm{dbl}(L)](N) \wedge (1^{n-32}\|01^{15}\|01^{15})$.

The third item increases the collision probability of counter blocks, which leads to a small degradation in security (as mentioned by the authors of EAX' [1, 5]).

6.2 Applicability of our Attacks to the Original EAX

We stress that our attacks against EAX' do not apply to the original EAX.[7]

By the second item in the above list of differences, EAX' uses the same constants D and Q as tweaks that CMAC uses internally. The original EAX uses two independent constants L and W at that place. This has three consequences:

1. Compared to an efficient implementation of the original EAX, an efficient implementation of EAX' saves the RAM to store L and W ($2 * 16$ bytes if the blockcipher is the AES).
2. The security proof for EAX does not apply to EAX' any more. (We believe one could adapt the proof of security for the original EAX to cope with the other differences between EAX and EAX', with slightly weaker bounds.)
3. Our attacks become possible.

6.3 Applicability to the ANSI C12.22 Protocol

All our attacks presented here require $\mathrm{size}(N) \leq n$. The forgery and distinguishing attacks also require $\mathrm{size}(P) \leq n$, and the message recovery attacks actually require at most the first n bits of the ciphertext.

In addition, the forgery and message recovery attacks could not be prevented by restricting the input length at encryption: one must implement the input length check at decryption as well.

Specifically, all our attacks would fail if $\mathrm{size}(N) > n$ would be guaranteed. We do not know whether this actually holds for the protocol specified by ANSI C12.22 [1]. One can find some ANSI C12.22 communication examples (Annex G of [1]) or test vectors of EAX' (Section V of [5]). Among them, there is no example[8] satisfying $\mathrm{size}(N) \leq n$ ($= 128$), however, some test vectors of [5] satisfy $\mathrm{size}(C) = n$.

[7] How could they? The original EAX has been rigorously proven to resist such attacks.

[8] In the test vectors of [5], there seems an editorial error; the cleartext may mean the plaintext and vice versa.

In [5], "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 P or C can be null (as stated by ANSI) and CMAC' taking the null string certainly operates on the single-block CBC, but it may be a hint that size$(N) > n$ would hold for any legitimate ANSI C12.22 messages.

6.4 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 128 bits. But even if the ANSI C12.22 prohibited any cleartexts of size 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 grids, 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 [3] 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.

6.5 Provable Security and Open Problem

In modern cryptography the standard approach for the designers of new blockcipher-based schemes is to formally *prove* the security of their schemes, assuming that the underlying blockcipher is secure. The authors of the original EAX did follow that approach.

As the current paper shows, even seemingly minor modifications of a scheme that has been proven secure need attention – the original security proof may not be applicable any more, and the modified scheme may be broken or not.

It is an open problem to prove or disprove the security of EAX' if size$(N) > n$.

Acknowledgments. This paper is based on the collaboration started at Dagstuhl Seminar 12031, Symmetric Cryptography. The authors thank participants of the seminar for useful comments, and discussions with Greg Rose were invaluable for writing Sect. 6.4. 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. American National Standards Institute. American National Standard Protocol Specification For Interfacing to Data Communication Networks. ANSI C12.22-2008.
2. M. Bellare, P. Rogaway, and D. Wagner. The EAX Mode of Operation. *Fast Software Encryption, FSE 2004*, LNCS 3017, pp. 389–407, Springer, 2004.
3. D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. *CRYPTO '98*, LNCS 1462, pp. 1–12, Springer, 1998.
4. T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. *Fast Software Encryption, FSE 2003*, LNCS 2887, pp. 129–153, Springer, 2003.
5. A. Moise, E. Beroset, T. Phinney, and M. Burns. EAX' Cipher Mode (May 2011). NIST Submission. Available from `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax-prime/eax-prime-spec.pdf`.
6. Toshiba Corporation. Comment for EAX' Cipher Mode. Available from `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/EAX%27/Toshiba_Report2NIST_rev051.pdf`.
7. National Institute of Standards and Technology. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B, 2005.
8. P. Rogaway. Nonce-Based Symmetric Encryption. *Fast Software Encryption, FSE 2004*, LNCS 3017, pp. 348–359, Springer, 2004.