CLOC: Authenticated Encryption for Short Input[⋆]

Tetsu Iwata[1], Kazuhiko Minematsu[2], Jian Guo[3], and Sumio Morioka[4]

[1] Nagoya University, Japan, iwata@cse.nagoya-u.ac.jp
[2] NEC Corporation, Japan, k-minematsu@ah.jp.nec.com
[3] Nanyang Technological University, Singapore, ntu.guo@gmail.com
[4] NEC Europe Ltd., United Kingdom, s-morioka@ak.jp.nec.com

Abstract. We define and analyze the security of a blockcipher mode of operation, CLOC, for provably secure authenticated encryption with associated data. The design of CLOC aims at optimizing previous schemes, CCM, EAX, and EAX-prime, in terms of the implementation overhead beyond the blockcipher, the precomputation complexity, and the memory requirement. With these features, CLOC is suitable for handling short input data, say 16 bytes, without needing precomputation nor large memory. This property is especially beneficial to small microprocessors, where the word size is typically 8 bits or 16 bits, and there are significant restrictions in the size and the number of registers. CLOC uses a variant of CFB mode in its encryption part and a variant of CBC MAC in the authentication part. We introduce various design techniques in order to achieve the above mentioned design goals. We prove CLOC secure, in a reduction-based provable security paradigm, under the assumption that the blockcipher is a pseudorandom permutation. We also present our preliminary implementation results.

Keywords: CLOC, blockcipher, authenticated encryption with associated data, security analysis, efficiency analysis.

1 Introduction

Background. An authenticated encryption with associated data scheme (AEAD) is a symmetric key cryptographic primitive that provides both confidentiality and integrity of plaintexts, and integrity of associated data. There are several ways of designing AEADs, and we focus on a design based on a blockcipher. CCM [39] was proposed by Whiting, Housley, and Ferguson for use within the IEEE 802.11 standard for Wireless LANs. It is adopted as NIST recommendation [17], and is broadly used in practice [21,22,9]. The mode is 2-pass, meaning that we run two algorithms, one for encryption and one for authentication. It is provably secure [25], but CCM suffers from a number of limitations, most notably it is not on-line; the encryption process cannot be started until knowing the whole input data. There are other issues in CCM [35], and EAX was proposed by Bellare, Rogaway, and Wagner to overcome these limitations [13]. EAX is included in ISO 19772 [9], and it has a number of attractive features; it is simple as it uses CMAC and CTR mode in a black-box manner, and it was designed by taking provable security into consideration. However, it has several implementation costs, and EAX-prime was designed by Moise, Beroset, Phinney, and Burns [31] to reduce the costs. It was designed to reduce the number of blockcipher calls both in precomputation and in processing the input data, to eliminate the key dependent constants, also called masks, to reduce memory requirement to store them, and to unify the associated data and the nonce, which contributes to reduce the memory requirement and the number of blockcipher calls as well. However, a practical attack was pointed out against EAX-prime [30], showing that it is not a secure AEAD. Later, Minematsu, Lucks, and Iwata proposed a variant of EAX called EAX$^+$, which has similar complexity as EAX-prime and is provably secure as EAX [29].

Presumably, though not clearly stated in the document [31], the most significant advantage of EAX-prime over original EAX (and CCM) is its efficient handling of *short input data* with small memory. As EAX-prime needs only one blockcipher call in precomputation whereas EAX needs three calls, EAX-prime gains the performance for short (say 16 bytes) input data, in particular if precomputation is difficult due to a limited amount of memory, or frequent key changes, or both. The performance for short input data is important for many practical applications, most notably for low-power wireless sensor networks, since messages are typically short to suppress the energy consumption of sensor nodes, which

[⋆] A preliminary version of this paper appears in the pre-proceedings of FSE 2014. This is the full version.

are usually battery-powered. For example, Zigbee [8] limits the maximum message length to be 127 bytes, and Bluetooth low energy limits the length to 47 bytes [4]. Another example is Electronic Product Code (EPC), which is a replacement of bar-code using RFID tags, and it typically has 96 bits [5].

Our Contributions. In this paper, we present a mode of operation, CLOC (which stands for Compact Low-Overhead CFB, and is pronounced as "clock"), to meet the demand. The design of CLOC aims at optimizing previous schemes, CCM, EAX, and EAX-prime, in terms of the implementation overhead beyond the blockcipher, the precomputation complexity, and the memory requirement. CLOC is sequential and its asymptotic performance (i.e. for long input data) is comparable to CCM, EAX, and EAX-prime. However, CLOC has a unique feature in its low overhead computation. CLOC works *without* any precomputation beyond the key scheduling of the blockcipher. Specifically, we do not need any blockcipher calls nor generating a key dependent table. This contributes to the improvement of the performance for short input data. For example, when the input data consists of 1-block nonce, 1-block associated data, and 1-block plaintext, CLOC needs 4 blockcipher calls, while we need 5 or 6 calls in CCM, 7 calls (where 3 out of 7 can be precomputed) in EAX, and 5 calls (where 1 out of 5 can be precomputed) in EAX-prime. We focus on provably secure schemes, but for comparison, there are lightweight AE schemes including ALE [16] and Fides [14], where ALE needs 44 AES rounds which amount to 4.4 AES calls (10 out of 44 AES rounds can be precomputed), and Fides needs 33 round function calls, where the round function is similar to that of AES but has larger state. This property of CLOC is particularly beneficial for embedded devices since the internal blockcipher is relatively slow due to limited computing power. Moreover, CLOC can be implemented using only two state blocks, i.e. the working memory of $2n$ bits with an n-bit blockcipher, except those needed for interfacing and blockcipher invocations. We do not aware of any provably secure AE mode with on-line capability to work with such a small amount of memory, and this property makes CLOC even suitable for small processors.

Important properties of CLOC can be summarized as follows.

1. It is a nonce-based authenticated encryption with associated data (AEAD).
2. It uses only the encryption of the blockcipher both for encryption and decryption.
3. It makes $\lceil |N|/n \rceil + \lceil |A|/n \rceil + 2\lceil |M|/n \rceil$ blockcipher calls for a nonce N, associated data A, and a plaintext M, when $|A| \geq 1$, where $|X|$ is the length of X in bits and n is the block length in bits of the blockcipher. No precomputation is needed. We note that in CLOC, $1 \leq |N| \leq n-1$ holds (hence we always have $\lceil |N|/n \rceil = 1$), and when $|A| = 0$, it needs $\lceil |N|/n \rceil + 1 + 2\lceil |M|/n \rceil$ blockcipher calls.
4. It works with two state blocks (i.e. $2n$ bits).

We introduce various design techniques in order to achieve the above mentioned design goals. We introduce *tweak functions* which are used to update the internal state at several points in the encryption and the decryption. While bit-wise operations, such as a constant multiplication over $\mathrm{GF}(2^n)$, are often employed in majority of previous schemes, considering the performance for small devices, we completely eliminate bit-wise operations. Instead, our tweak functions consist of word-wise permutations and xor's. As a result, each tweak function can be described by using a 4×4 binary matrix.

The use of word-wise permutations and xor's to update a mask or a key dependent constant was discussed in [23,29], and the approach was applied on CMAC and EAX. Here we use them directly to update the internal state, instead of updating a key dependent constant and xoring it to the state. This was employed for example in designs of MACs [32,40] using bit shift operations. The techniques introduced here seem to be worth for other areas, e.g., in designing MACs, and thus it may be of independent interest.

We also introduce bit-fixing functions. CFB mode leaks input and output pairs of the underlying blockcipher, which may result in the loss of security. We use the functions to logically separate the encryption part and the authentication part of CLOC.

With these techniques, we prove CLOC secure, in a reduction-based provable security paradigm, under the assumption that the blockcipher is a pseudorandom permutation. For security notions, CLOC fulfills the standard security notions for nonce-based AEADs, i.e., the privacy and the authenticity under nonce-respecting adversaries [34]. Furthermore, we prove that the authenticity notion holds even for nonce-reusing adversaries, where only a small number of schemes achieve this goal, and most of known modes do fail to provide [19]. See Table 1 for a brief comparison of CLOC to other AEADs.

Table 1. Comparison of AE modes, for a-block associated data and m-block message with one-block nonce, where $a \geq 1$

Property∘	CCM [17]	GCM [18]	EAX [13]	EAX-prime [31]	OCB3 [26]	CLOC
Calls	$a+2m+2$†	$m+1$‡	$a+2m+1$	$a+2m+1$	$a+m+1$†	$a+2m+1$
Setup	0	1	3	1	1	0
On-line	No	Yes	Yes	Yes	Yes	Yes
Static AD	No	Yes	Yes	Yes	Yes	Yes
Parallel	No	Yes	No	No	Yes	No
Primitive	E	E, GHASH	E	E	E, D	E
PRIV/AUTH⋆	$O(2^{n/2})$[25]	$O(2^{n/2})$[24]	$O(2^{n/2})$[13]	$O(1)$[30]	$O(2^{n/2})$[26]	$O(2^{n/2})$
N-AUTH⋄	$\ll 2^{n/2}$[20,19]	$O(1)$[19]	$O(1)$[19]	$O(1)$[30]	$O(1)$[19]	$O(2^{n/2})$

∘ "Setup" shows the number of blockcipher calls for setup, "Static AD" shows if efficient handling of static associated data is possible, "Parallel" shows if the blockcipher calls are parallelizable, and "Primitive" shows the components of the mode. E is the encryption of the blockcipher and D is the decryption.
† May have additional one call
‡ Plus $a+m$ multiplications over GF(2^n)
⋆ Attack workload of nonce-respecting adversaries to break the privacy notion or the authenticity notion
⋄ Attack workload of nonce-reusing adversaries to break the authenticity notion

2 Preliminaries

Let $\{0,1\}^*$ be the set of all finite bit strings, including the empty string ε. For an integer $\ell \geq 0$, let $\{0,1\}^\ell$ be the set of all bit strings of ℓ bits. For $X, Y \in \{0,1\}^*$, we write $X \,\|\, Y$, (X, Y), or simply XY to denote their concatenation. For $\ell \geq 0$, we write $0^\ell \in \{0,1\}^\ell$ to denote the bit string that consists of ℓ zeros, and $1^\ell \in \{0,1\}^\ell$ to denote the bit string that consists of ℓ ones. For $X \in \{0,1\}^*$, $|X|$ is its length in bits, and for $\ell \geq 1$, $|X|_\ell = \lceil |X|/\ell \rceil$ is the length in ℓ-bit blocks. For $X \in \{0,1\}^*$ and $\ell \geq 0$ such that $|X| \geq \ell$, $\mathsf{msb}_\ell(X)$ is the most significant (the leftmost) ℓ bits of X. For instance we have $\mathsf{msb}_1(1100) = 1$ and $\mathsf{msb}_3(1100) = 110$. For $X \in \{0,1\}^*$ and $\ell \geq 1$, we write its partition into ℓ-bit blocks as $(X[1], \ldots, X[x]) \overset{\ell}{\leftarrow} X$, which is defined as follows. If $X = \varepsilon$, then $x = 1$ and $X[1] \overset{\ell}{\leftarrow} X$, where $X[1] = \varepsilon$. Otherwise $X[1], \ldots, X[x] \in \{0,1\}^*$ are unique bit strings such that $X[1] \,\|\, \cdots \,\|\, X[x] = X$, $|X[1]| = \cdots = |X[x-1]| = \ell$, and $1 \leq |X[x]| \leq \ell$.

In what follows, we fix a block length n and a blockcipher $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}_E$ is a non-empty set of keys. Let Perm(n) be the set of all permutations over $\{0,1\}^n$. We write $E_K \in \text{Perm}(n)$ for the permutation specified by $K \in \mathcal{K}_E$, and $C = E_K(M)$ for the ciphertext of plaintext $M \in \{0,1\}^n$ under key $K \in \mathcal{K}_E$.

3 Specification of CLOC

CLOC takes three parameters, a blockcipher $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$, a nonce length ℓ_N, and a tag length τ. We require $1 \leq \ell_N \leq n-1$ and $1 \leq \tau \leq n$. We also require that $n/4$ is an integer. We write CLOC$[E, \ell_N, \tau]$ for CLOC that is parameterized by E, ℓ_N, and τ, and we often omit the parameters if they are irrelevant or they are clear from the context. CLOC$[E, \ell_N, \tau]$ = (CLOC-$\mathcal{E}$, CLOC-$\mathcal{D}$) consists of the encryption algorithm CLOC-$\mathcal{E}$ and the decryption algorithm CLOC-$\mathcal{D}$.

CLOC-$\mathcal{E}$ and CLOC-$\mathcal{D}$ have the following syntax.

$$\begin{cases} \text{CLOC-}\mathcal{E} : \mathcal{K}_{\text{CLOC}} \times \mathcal{N}_{\text{CLOC}} \times \mathcal{A}_{\text{CLOC}} \times \mathcal{M}_{\text{CLOC}} \to \mathcal{CT}_{\text{CLOC}} \\ \text{CLOC-}\mathcal{D} : \mathcal{K}_{\text{CLOC}} \times \mathcal{N}_{\text{CLOC}} \times \mathcal{A}_{\text{CLOC}} \times \mathcal{CT}_{\text{CLOC}} \to \mathcal{M}_{\text{CLOC}} \cup \{\perp\} \end{cases}$$

$\mathcal{K}_{\text{CLOC}} = \mathcal{K}_E$ is the key space, which is identical to the key space of the underlying blockcipher, $\mathcal{N}_{\text{CLOC}} = \{0,1\}^{\ell_N}$ is the nonce space, $\mathcal{A}_{\text{CLOC}} = \{0,1\}^*$ is the associated data space, $\mathcal{M}_{\text{CLOC}} = \{0,1\}^*$ is the plaintext space, $\mathcal{CT}_{\text{CLOC}} = \mathcal{C}_{\text{CLOC}} \times \mathcal{T}_{\text{CLOC}}$ is the ciphertext space, where $\mathcal{C}_{\text{CLOC}} = \{0,1\}^*$ and $\mathcal{T}_{\text{CLOC}} = \{0,1\}^\tau$ is the tag space, and $\perp \notin \mathcal{M}_{\text{CLOC}}$ is the distinguished reject symbol. We write $(C, T) \leftarrow \text{CLOC-}\mathcal{E}_K(N, A, M)$ and $M \leftarrow \text{CLOC-}\mathcal{D}_K(N, A, C, T)$ or $\perp \leftarrow \text{CLOC-}\mathcal{D}_K(N, A, C, T)$.

CLOC-$\mathcal{E}$ and CLOC-$\mathcal{D}$ are defined in Fig. 1. In these algorithms, we use four subroutines, HASH, PRF, ENC, and DEC. They have the following syntax.

$$\begin{cases} \mathsf{HASH} : \mathcal{K}_{\mathrm{CLOC}} \times \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \to \{0,1\}^n \\ \mathsf{PRF} : \mathcal{K}_{\mathrm{CLOC}} \times \{0,1\}^n \times \mathcal{C}_{\mathrm{CLOC}} \to \mathcal{T}_{\mathrm{CLOC}} \\ \mathsf{ENC} : \mathcal{K}_{\mathrm{CLOC}} \times \{0,1\}^n \times \mathcal{M}_{\mathrm{CLOC}} \to \mathcal{C}_{\mathrm{CLOC}} \\ \mathsf{DEC} : \mathcal{K}_{\mathrm{CLOC}} \times \{0,1\}^n \times \mathcal{C}_{\mathrm{CLOC}} \to \mathcal{M}_{\mathrm{CLOC}} \end{cases}$$

These subroutines are defined in Fig. 2, and illustrated in Fig. 3, Fig. 4, and Fig. 5. We also present equivalent figures in Fig. 6, Fig. 7, and Fig. 8, where it is easier to see that CLOC works with small state. In the figures, i is the identity function, and $\mathsf{i}(X) = X$ for all $X \in \{0,1\}^n$. In the subroutines, we use the one-zero padding function $\mathsf{ozp} : \{0,1\}^* \to \{0,1\}^*$, the bit-fixing functions $\mathsf{fix0}, \mathsf{fix1} : \{0,1\}^* \to \{0,1\}^*$, and five tweak functions f_1, f_2, g_1, g_2, and h, which are functions over $\{0,1\}^n$.

The one-zero padding function ozp is used to adjust the length of an input string so that the total length becomes a positive multiple of n bits. For $X \in \{0,1\}^*$, $\mathsf{ozp}(X)$ is defined as $\mathsf{ozp}(X) = X$ if $|X| = \ell n$ for some $\ell \geq 1$, and $\mathsf{ozp}(X) = X \,\|\, 10^{n-1-(|X| \bmod n)}$ otherwise. We note that $\mathsf{ozp}(\varepsilon) = 10^{n-1}$, and we also note that, in general, the function is not invertible.

The bit-fixing functions $\mathsf{fix0}$ and $\mathsf{fix1}$ are used to fix the most significant bit of an input string to zero and one, respectively. For $X \in \{0,1\}^*$, $\mathsf{fix0}(X)$ is defined as $\mathsf{fix0}(X) = X \wedge 01^{|X|-1}$, and $\mathsf{fix1}(X)$ is defined as $\mathsf{fix1}(X) = X \vee 10^{|X|-1}$, where $\wedge$ and $\vee$ are the bit-wise AND operation, and the bit-wise OR operation, respectively.

The tweak function h is used in HASH if the most significant bit of $\mathsf{ozp}(A[1])$ is zero. We use f_1 and f_2 in HASH and PRF, where f_1 is used if the last input block is full (i.e., if $|A[a]| = n$ or $|C[m]| = n$) and f_2 is used otherwise. We use g_1 and g_2 in PRF, where we use g_1 if the second argument of the input is the empty string (i.e., $|C| = 0$), and otherwise we use g_2. Now for $X \in \{0,1\}^n$, let $(X[1], X[2], X[3], X[4]) \overset{n/4}{\leftarrow} X$. Then f_1, f_2, g_1, g_2, and h are defined as follows.

$$\begin{cases} \mathsf{f}_1(X) = (X[1,3], X[2,4], X[1,2,3], X[2,3,4]) \\ \mathsf{f}_2(X) = (X[2], X[3], X[4], X[1,2]) \\ \mathsf{g}_1(X) = (X[3], X[4], X[1,2], X[2,3]) \\ \mathsf{g}_2(X) = (X[2], X[3], X[4], X[1,2]) \\ \mathsf{h}(X) = (X[1,2], X[2,3], X[3,4], X[1,2,4]) \end{cases}$$

Here $X[a,b]$ stands for $X[a] \oplus X[b]$ and $X[a,b,c]$ stands for $X[a] \oplus X[b] \oplus X[c]$.

Alternatively the tweak functions can be specified by a matrix. Let

$$\mathbf{M} = \begin{pmatrix} 0\,0\,0\,1 \\ 1\,0\,0\,1 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix} \tag{1}$$

be a 4×4 binary matrix, and let $\mathbf{M}^i$ for $i \geq 0$ be exponentiations of $\mathbf{M}$, where $\mathbf{M}^0$ denotes the identity matrix. Then we have $\mathsf{f}_1(X) = X \cdot \mathbf{M}^8$, $\mathsf{f}_2(X) = X \cdot \mathbf{M}$, $\mathsf{g}_1(X) = X \cdot \mathbf{M}^2$, $\mathsf{g}_2(X) = X \cdot \mathbf{M}$, and $\mathsf{h}(X) = X \cdot \mathbf{M}^4$, where $X = (X[1], X[2], X[3], X[4])$ is interpreted as a vector.

The design rationale for the tweak functions is explained in Sect. 4.

4 Design Rationale

Overall Structure. At abstract level CLOC is a straightforward combination of CFB and CBC MAC, where CBC MAC is called twice for processing associated data and a ciphertext, and CFB is called once to generate a ciphertext. However, when we want to achieve low-overhead computation and small memory consumption, we found that any other combination of a basic encryption mode and a MAC mode did not work. For instance, we could not use CTR or OFB, as they require one state block in processing a plaintext to hold a counter value or a blockcipher output. We then realized that combining CFB and CBC MAC was not an easy task. Since we avoid using two keys or using blockcipher pre-calls,

```
Algorithm CLOC-$\mathcal{E}_K(N, A, M)$

 1. $V \leftarrow \mathsf{HASH}_K(N, A)$
 2. $C \leftarrow \mathsf{ENC}_K(V, M)$
 3. $T \leftarrow \mathsf{PRF}_K(V, C)$
 4. return $(C, T)$
```

```
Algorithm CLOC-$\mathcal{D}_K(N, A, C, T)$

 1. $V \leftarrow \mathsf{HASH}_K(N, A)$
 2. $T^* \leftarrow \mathsf{PRF}_K(V, C)$
 3. if $T \neq T^*$ then return $\perp$
 4. $M \leftarrow \mathsf{DEC}_K(V, C)$
 5. return $M$
```

Fig. 1. Pseudocode of the encryption and the decryption algorithms of CLOC

```
Algorithm $\mathsf{HASH}_K(N, A)$

 1. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
 2. $S_\mathsf{H}[1] \leftarrow E_K(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
 3. if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$ then
 4.     $S_\mathsf{H}[1] \leftarrow \mathsf{h}(S_\mathsf{H}[1])$
 5. if $a \geq 2$ then
 6.     for $i \leftarrow 2$ to $a - 1$ do
 7.         $S_\mathsf{H}[i] \leftarrow E_K(S_\mathsf{H}[i-1] \oplus A[i])$
 8.     $S_\mathsf{H}[a] \leftarrow E_K(S_\mathsf{H}[a-1] \oplus \mathsf{ozp}(A[a]))$
 9. if $|A[a]| = n$ then
10.     $V \leftarrow \mathsf{f}_1(S_\mathsf{H}[a] \oplus \mathsf{ozp}(N))$
11. else                          // $0 \leq |A[a]| \leq n - 1$
12.     $V \leftarrow \mathsf{f}_2(S_\mathsf{H}[a] \oplus \mathsf{ozp}(N))$
13. return $V$
```

```
Algorithm $\mathsf{PRF}_K(V, C)$

 1. if $|C| = 0$ then
 2.     $T \leftarrow \mathsf{msb}_\tau(E_K(\mathsf{g}_1(V)))$
 3.     return $T$
 4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
 5. $S_\mathsf{P}[0] \leftarrow E_K(\mathsf{g}_2(V))$
 6. for $i \leftarrow 1$ to $m - 1$ do
 7.     $S_\mathsf{P}[i] \leftarrow E_K(S_\mathsf{P}[i-1] \oplus C[i])$
 8. if $|C[m]| = n$ then
 9.     $S_\mathsf{P}[m] \leftarrow E_K(\mathsf{f}_1(S_\mathsf{P}[m-1] \oplus C[m]))$
10. else                          // $1 \leq |C[m]| \leq n - 1$
11.     $S_\mathsf{P}[m] \leftarrow E_K(\mathsf{f}_2(S_\mathsf{P}[m-1] \oplus \mathsf{ozp}(C[m])))$
12. $T \leftarrow \mathsf{msb}_\tau(S_\mathsf{P}[m])$
13. return $T$
```

```
Algorithm $\mathsf{ENC}_K(V, M)$

 1. if $|M| = 0$ then
 2.     $C \leftarrow \varepsilon$
 3.     return $C$
 4. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
 5. $S_\mathsf{E}[1] \leftarrow E_K(V)$
 6. for $i \leftarrow 1$ to $m - 1$ do
 7.     $C[i] \leftarrow S_\mathsf{E}[i] \oplus M[i]$
 8.     $S_\mathsf{E}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
 9. $C[m] \leftarrow \mathsf{msb}_{|M[m]|}(S_\mathsf{E}[m]) \oplus M[m]$
10. $C \leftarrow (C[1], \ldots, C[m])$
11. return $C$
```

```
Algorithm $\mathsf{DEC}_K(V, C)$

 1. if $|C| = 0$ then
 2.     $M \leftarrow \varepsilon$
 3.     return $M$
 4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
 5. $S_\mathsf{D}[1] \leftarrow E_K(V)$
 6. for $i \leftarrow 1$ to $m - 1$ do
 7.     $M[i] \leftarrow S_\mathsf{D}[i] \oplus C[i]$
 8.     $S_\mathsf{D}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
 9. $M[m] \leftarrow \mathsf{msb}_{|C[m]|}(S_\mathsf{D}[m]) \oplus C[m]$
10. $M \leftarrow (M[1], \ldots, M[m])$
11. return $M$
```

Fig. 2. Subroutines used in the encryption and decryption algorithms of CLOC

such as $L = E_K(0^n)$ used in EAX, we could not computationally separate CFB and CBC MAC via input masking, such as Galois-field doubling (2^iL for the i-th block, where $2L$ denotes the multiplication of 2 and L in GF(2^n)) [13,33]. This implies that CFB leaks input and output pairs of the blockcipher calls, which can be freely used to guess or fake the internal chaining value of CBC MAC, leading to a break of the scheme. Lucks [28] proposed an AEAD scheme based on CFB, called CCFB. However, the problem is not relevant to CCFB due to the difference in the global structure. To overcome this obstacle in composition, we introduced the bit-fixing functions. Their role is to absolutely separate the input blocks of CFB and *the first input block* of CBC MAC. This imposes the most significant one bit of the input of CBC MAC being fixed to 0, implying one-bit input loss. The set of five tweak functions, which is another tool we introduced in this paper, is used to compensate for this information loss. It also works to compensate the information loss caused by padding functions applied to the last input block to CBC MAC. A similar technique can be found in literature [32,40], however, the previous works only considered MACs and the tweak functions required bit operations.

In the following we explain the specific requirements for the tweak functions.

Definition of f_1*,* f_2*,* g_1*,* g_2*, and* h*.* These functions are defined to meet the following properties. First, they have the additive property. That is, for any $\mathsf{z} \in \{\mathsf{f}_1, \mathsf{f}_2, \mathsf{g}_1, \mathsf{g}_2, \mathsf{h}\}$, we have $\mathsf{z}(X \oplus X') = \mathsf{z}(X) \oplus \mathsf{z}(X')$ for all

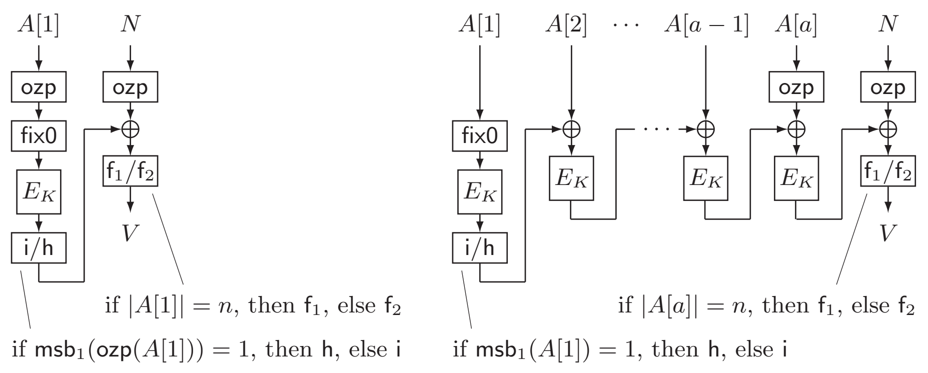

A[1]
N
ozp
ozp
fix0
E_K
f_1/f_2
V
i/h
if |A[1]| = n, then f_1, else f_2
if msb_1(ozp(A[1])) = 1, then h, else i
A[1]
A[2]
···
A[a − 1]
A[a]
N
ozp
ozp
fix0
E_K
E_K
E_K
E_K
f_1/f_2
V
i/h
if |A[a]| = n, then f_1, else f_2
if msb_1(A[1]) = 1, then h, else i

Fig. 3. $V \leftarrow \mathsf{HASH}_K(N, A)$ for $0 \le |A| \le n$ (left) and $|A| \ge n + 1$ (right)

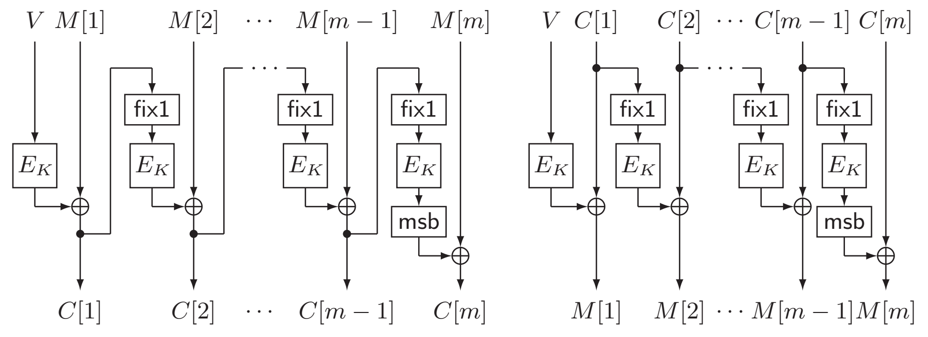

V M[1] M[2] ··· M[m − 1] M[m]
fix1
fix1
fix1
E_K
msb
C[1] C[2] ··· C[m − 1] C[m]
V C[1] C[2] ··· C[m − 1] C[m]
fix1
fix1
fix1
E_K
msb
M[1] M[2] ··· M[m − 1] M[m]

Fig. 4. $C \leftarrow \mathsf{ENC}_K(V, M)$ for $|M| \ge 1$ (left), and $\mathsf{DEC}_K(V, C)$ for $|C| \ge 1$ (right)

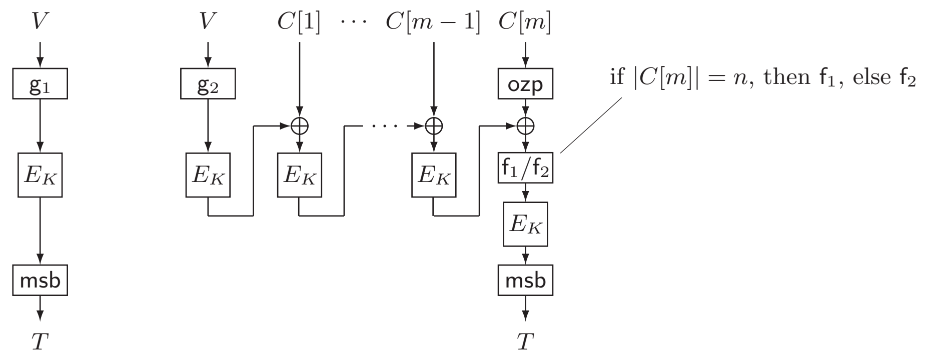


Fig. 5. $T \leftarrow \mathsf{PRF}_K(V, C)$ for $|C| = 0$ (left), and $|C| \ge 1$ (right)

$X, X' \in \{0,1\}^n$. Next, these functions are invertible over $\{0,1\}^n$. For any $\mathsf{z} \in \{\mathsf{f}_1, \mathsf{f}_2, \mathsf{g}_1, \mathsf{g}_2, \mathsf{h}\}$, we have $\mathsf{z} \in \mathrm{Perm}(n)$. Finally, they satisfy the differential probability constraints specified in Fig. 9. Let z be a function in Fig. 9. Then we require that, for any $Y \in \{0,1\}^n$, $\Pr[\mathsf{z}(K) = Y] = 1/2^n$, where the probability is taken over $K \xleftarrow{\$} \{0,1\}^n$. When z is of the form $\mathsf{z} = \mathsf{z}' \oplus \mathsf{z}''$, then $\mathsf{z}(K)$ stands for $\mathsf{z}'(K) \oplus \mathsf{z}''(K)$. When z is of the form $\mathsf{z} = \mathsf{z}'\mathsf{z}''$, then $\mathsf{z}(K)$ stands for $\mathsf{z}'(\mathsf{z}''(K))$. Recall that we define i as $\mathsf{i}(K) = K$.

Choosing Tweak Functions. Finding simple and word-wise tweak functions fulfilling all properties is not a trivial task. We start with matrix $\mathbf{M}$ of (1), which is invertible and has order 15 (i.e. $\mathbf{M}^{15} = \mathbf{M}^0$), and test all combinations of the form $(\mathsf{f}_1, \mathsf{f}_2, \mathsf{g}_1, \mathsf{g}_2, \mathsf{h}) = (i_1, \ldots, i_5) \in \{1, \ldots, 14\}^5$, where $i_1 = 2$ means $\mathsf{f}_1(X) = X \cdot \mathbf{M}^2$, using a computer. There are 864 candidates out of 537,824 fulfilling the differential

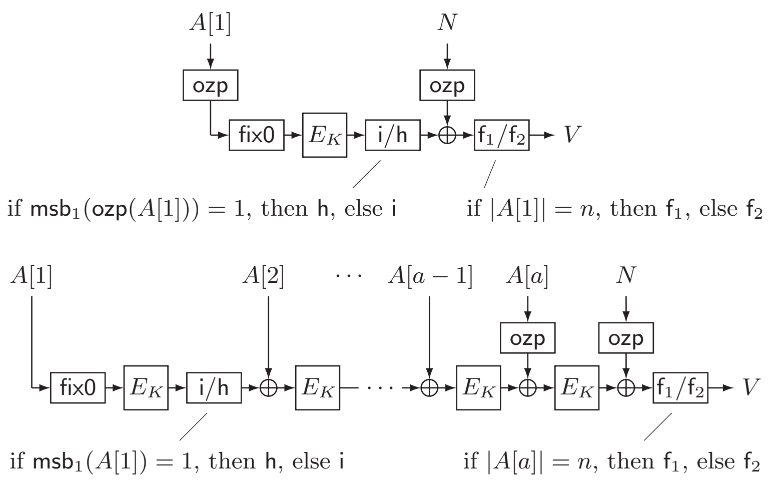

$A[1]$
N
ozp
ozp
fix0
E_K
i/h
$\mathsf{f}_1/\mathsf{f}_2$
V
if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$, then h, else i
if $|A[1]| = n$, then f_1, else f_2
$A[1]$
$A[2]$
$\cdots$
$A[a-1]$
$A[a]$
N
ozp
ozp
fix0
E_K
i/h
E_K
E_K
E_K
$\mathsf{f}_1/\mathsf{f}_2$
V
if $\mathsf{msb}_1(A[1]) = 1$, then h, else i
if $|A[a]| = n$, then f_1, else f_2

Fig. 6. $V \leftarrow \mathsf{HASH}_K(N, A)$ for $0 \le |A| \le n$ (top) and $|A| \ge n+1$ (bottom)

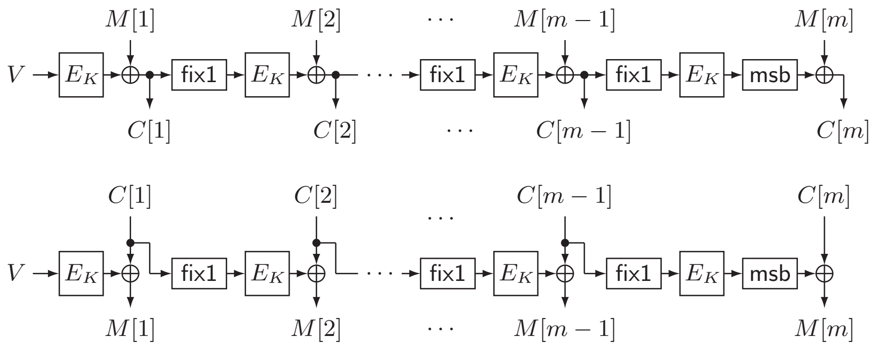


Fig. 7. $C \leftarrow \mathsf{ENC}_K(V, M)$ for $|M| \ge 1$ (top), and $\mathsf{DEC}_K(V, C)$ for $|C| \ge 1$ (bottom)

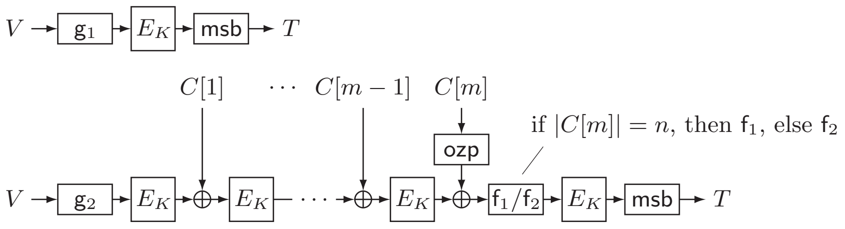


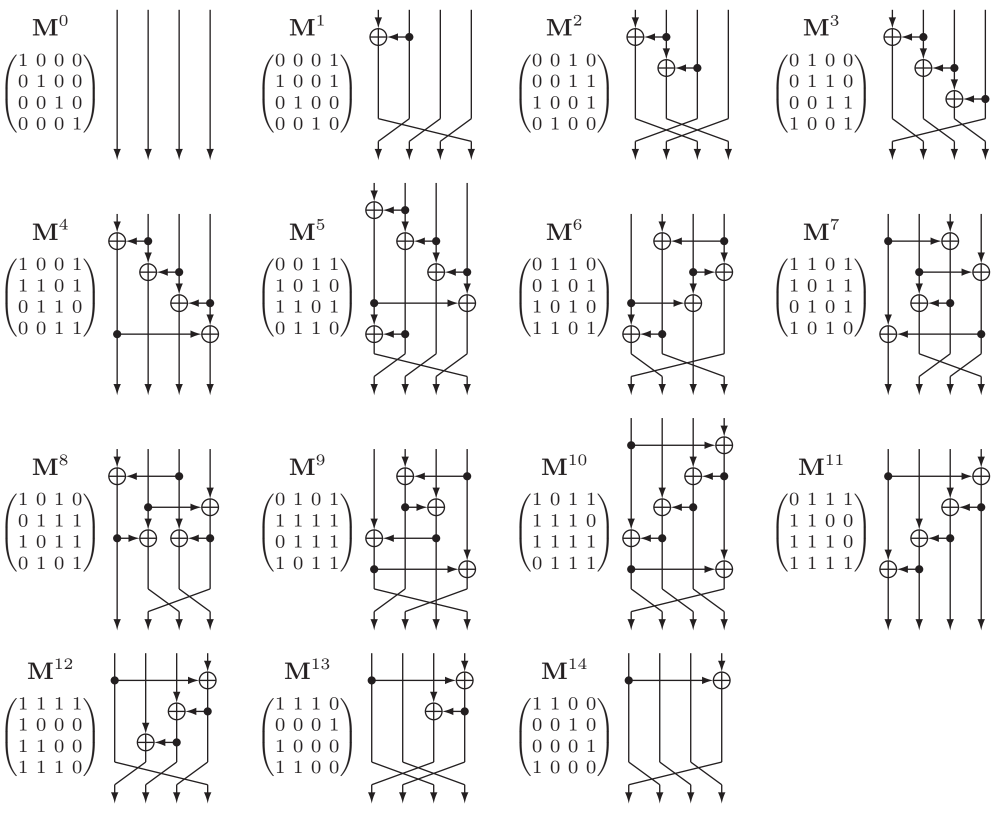
Fig. 8. $T \leftarrow \mathsf{PRF}_K(V, C)$ for $|C| = 0$ (top), and $|C| \ge 1$ (bottom)

probability constraints of Fig. 9. The complexity increases as the index of $\mathbf{M}$ grows, when we implement the tweak function by iterating $\mathbf{M}$, which seems suitable for hardware. For software we would directly implement $\mathbf{M}^i$ using a word-wise permutation and xor, and in this case we observe slight irregular, but similar phenomena (e.g. $\mathbf{M}^1$ needs one xor while $\mathbf{M}^3$ needs three xor's). Fig. 10 shows $\mathbf{M}^i$ and the Feistel-like implementations using a word-wise permutation and xor. It shows that, except for $\mathbf{M}^5$ and $\mathbf{M}^{10}$, we have a simple implementation using at most four xor's. Based on these observations, we simply define the cost of computing $\mathbf{M}^i$ as i, and define $f_{\mathrm{cost}}(i_1, \ldots, i_5)$ as

$$\left(i_1 \times \frac{1}{16} + i_2 \times \frac{15}{16}\right) \times 2 + i_4 + i_5 \times \frac{1}{2}.$$

$i \oplus f_1$	$i \oplus f_2h$	$f_1 \oplus f_2h$	$h \oplus g_2f_1$	$g_2f_1 \oplus g_1f_2h$
$i \oplus g_1f_1$	$i \oplus h$	$f_2 \oplus g_1f_1$	$h \oplus f_2$	$g_2f_1 \oplus f_2h$
$i \oplus g_1f_1h$	$i \oplus g_1$	$f_2 \oplus g_1f_1h$	$h \oplus g_1f_2$	$g_1f_2 \oplus g_2f_1$
$i \oplus g_2f_1$	$i \oplus g_2$	$f_2 \oplus g_2f_1$	$h \oplus g_2f_2$	$g_1f_2 \oplus g_2f_1h$
$i \oplus g_2f_1h$	$f_1 \oplus g_1f_1h$	$f_2 \oplus g_2f_1h$	$g_1f_1 \oplus f_1h$	$g_1f_2 \oplus f_1h$
$i \oplus f_1h$	$f_1 \oplus g_2f_1h$	$f_2 \oplus f_1h$	$g_1f_1 \oplus g_2f_1h$	$g_1f_2 \oplus g_2f_2h$
$i \oplus f_2$	$f_1 \oplus f_2$	$f_2 \oplus g_1f_2h$	$g_1f_1 \oplus g_2f_2$	$g_1f_2 \oplus f_2h$
$i \oplus g_1f_2$	$f_1 \oplus g_1f_2$	$f_2 \oplus g_2f_2h$	$g_1f_1 \oplus g_2f_2h$	$g_2f_2 \oplus g_1f_1h$
$i \oplus g_1f_2h$	$f_1 \oplus g_1f_2h$	$g_1 \oplus g_2$	$g_1f_1 \oplus f_2h$	$g_2f_2 \oplus f_1h$
$i \oplus g_2f_2$	$f_1 \oplus g_2f_2$	$h \oplus f_1$	$g_2f_1 \oplus g_1f_1h$	$g_2f_2 \oplus g_1f_2h$
$i \oplus g_2f_2h$	$f_1 \oplus g_2f_2h$	$h \oplus g_1f_1$	$g_2f_1 \oplus f_1h$	$g_2f_2 \oplus f_2h$

Fig. 9. Differential probability constraints of f_1, f_2, g_1, g_2, and h



M^0 (1 0 0 0 / 0 1 0 0 / 0 0 1 0 / 0 0 0 1)
M^1 (0 0 0 1 / 1 0 0 1 / 0 1 0 0 / 0 0 1 0)
M^2 (0 0 1 0 / 0 0 1 1 / 1 0 0 1 / 0 1 0 0)
M^3 (0 1 0 0 / 0 1 1 0 / 0 0 1 1 / 1 0 0 1)
M^4 (1 0 0 1 / 1 1 0 1 / 0 1 1 0 / 0 0 1 1)
M^5 (0 0 1 1 / 1 0 1 0 / 1 1 0 1 / 0 1 1 0)
M^6 (0 1 1 0 / 0 1 0 1 / 1 0 1 0 / 1 1 0 1)
M^7 (1 1 0 1 / 1 0 1 1 / 0 1 0 1 / 1 0 1 0)
M^8 (1 0 1 0 / 0 1 1 1 / 1 0 1 1 / 0 1 0 1)
M^9 (0 1 0 1 / 1 1 1 1 / 0 1 1 1 / 1 0 1 1)
M^10 (1 0 1 1 / 1 1 1 0 / 1 1 1 1 / 0 1 1 1)
M^11 (0 1 1 1 / 1 1 0 0 / 1 1 1 0 / 1 1 1 1)
M^12 (1 1 1 1 / 1 0 0 0 / 1 1 0 0 / 1 1 1 0)
M^13 (1 1 1 0 / 0 0 0 1 / 1 0 0 0 / 1 1 0 0)
M^14 (1 1 0 0 / 0 0 1 0 / 0 0 0 1 / 1 0 0 0)

Fig. 10. Matrix exponentiations for the tweak functions

This corresponds to the expected total cost for given $(i_1, \ldots, i_5)$, where associated data and a plaintext are assumed to be non-empty byte strings of random lengths (as we expect the standard use of CLOC is AEAD, not MAC), and we also assume that the most significant bit of the associated data is random. Then there remains only two candidates giving the minimum value of f_{cost}, which are $(i_1, \ldots, i_5) = (8, 1, 2, 1, 4)$ and $(8, 1, 6, 1, 4)$. As smaller i_3 is better, we choose the former as the sole winner. We also tested other matrices, say the one replacing the forth column of $\mathbf{M}$ by the transposition of $(1, 0, 1, 0)$, but no better solution was found.

We note that $\mathbf{M}^8 = \mathbf{M}^2 \oplus \mathbf{M}^0$ and $\mathbf{M}^4 = \mathbf{M}^1 \oplus \mathbf{M}^0$ hold, implying that we have $f_1(X) = g_1(X) \oplus X$ and $h(X) = f_2(X) \oplus X = g_2(X) \oplus X$, which may be useful in some implementations.

5 Security of CLOC

In this section, we define the security notions of a blockcipher and CLOC, and present our security theorems.

PRP Notion. We assume that the blockcipher $E : \mathcal{K}_E \times \{0,1\}^n \to \{0,1\}^n$ is a pseudo-random permutation, or a PRP [27]. We say that P is a random permutation if $P \xleftarrow{\$} \mathrm{Perm}(n)$, and define

$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr\left[\mathcal{A}^{E_K(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{P(\cdot)} \Rightarrow 1\right],$$

where the first probability is taken over $K \xleftarrow{\$} \mathcal{K}_E$ and the randomness of $\mathcal{A}$, and the last is over $P \xleftarrow{\$} \mathrm{Perm}(n)$ and $\mathcal{A}$. We write CLOC$[\mathrm{Perm}(n), \ell_N, \tau]$ for CLOC that uses P as E_K, and the encryption and decryption algorithms are written as CLOC-$\mathcal{E}_P$ and CLOC-$\mathcal{D}_P$. We also consider CLOC that uses a random function as E_K, which is naturally defined as the invertibility of E_K is irrelevant in the definition of CLOC. Let $\mathrm{Rand}(n)$ be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$, and we say that R is a random function if $R \xleftarrow{\$} \mathrm{Rand}(n)$. We write CLOC$[\mathrm{Rand}(n), \ell_N, \tau]$ for CLOC that uses R as E_K, and its encryption and decryption algorithms are written as CLOC-$\mathcal{E}_R$ and CLOC-$\mathcal{D}_R$.

Privacy Notion. We define the privacy notion for CLOC$[E, \ell_N, \tau] = ($CLOC-$\mathcal{E}$, CLOC-$\mathcal{D})$. This notion captures the indistinguishably of a nonce-respecting adversary in a chosen plaintext attack setting. We consider an adversary $\mathcal{A}$ that has access to the CLOC encryption oracle, or a random-bits oracle. The encryption oracle takes $(N, A, M) \in \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{M}_{\mathrm{CLOC}}$ as input and returns $(C, T) \leftarrow$ CLOC-$\mathcal{E}_K(N, A, M)$. The random-bits oracle, \$-oracle, takes $(N, A, M) \in \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{M}_{\mathrm{CLOC}}$ as input and returns a random string $(C, T) \xleftarrow{\$} \{0,1\}^{|M|+\tau}$. We define the privacy advantage as

$$\mathbf{Adv}_{\mathrm{CLOC}[E,\ell_N,\tau]}^{\mathrm{priv}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr\left[\mathcal{A}^{\mathrm{CLOC}\text{-}\mathcal{E}_K(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot,\cdot)} \Rightarrow 1\right],$$

where the first probability is taken over $K \xleftarrow{\$} \mathcal{K}_{\mathrm{CLOC}}$ and the randomness of $\mathcal{A}$, and the last is over the random-bits oracle and $\mathcal{A}$. We assume that $\mathcal{A}$ in the privacy game is nonce-respecting, that is, $\mathcal{A}$ does not make two queries with the same nonce.

Privacy Theorem. Let $\mathcal{A}$ be an adversary that makes q queries, and suppose that the queries are $(N_1, A_1, M_1), \ldots, (N_q, A_q, M_q)$. Then we define the total associated data length as $a_1 + \cdots + a_q$, and the total plaintext length as $m_1 + \cdots + m_q$, where $(A_i[1], \ldots, A_i[a_i]) \xleftarrow{n} A_i$ and $(M_i[1], \ldots, M_i[m_i]) \xleftarrow{n} M_i$. We have the following information theoretic result.

Theorem 1. *Let* $\mathrm{Perm}(n)$*,* ℓ_N*, and* τ *be the parameters of* CLOC*. Let* $\mathcal{A}$ *be an adversary that makes at most* q *queries, where the total associated data length is at most* σ_A*, and the total plaintext length is at most* σ_M*. Then we have* $\mathbf{Adv}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}^{\mathrm{priv}}(\mathcal{A}) \le 5\sigma_{\mathrm{priv}}^2/2^n$*, where* $\sigma_{\mathrm{priv}} = q + \sigma_A + 2\sigma_M$*.*

A proof overview is given in Sect. 6, and a complete proof is presented in Appendix A. If we use a blockcipher E, which is secure in the sense of the PRP notion, instead of $\mathrm{Perm}(n)$, then the corresponding complexity theoretic result can be shown by a standard argument. See e.g. [11]. We note that the privacy of CLOC is broken if the nonce is reused.

Authenticity Notion. We next define the authenticity notion, which captures the unforgeability of an adversary in a chosen ciphertext attack setting. We consider a strong adversary that can repeat the same nonce multiple times. Let $\mathcal{A}$ be an adversary that has access to the CLOC encryption oracle and the CLOC decryption oracle. The encryption oracle is defined as above. The decryption oracle takes $(N, A, C, T) \in \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{C}_{\mathrm{CLOC}} \times \mathcal{T}_{\mathrm{CLOC}}$ as input and returns $M \leftarrow$ CLOC-$\mathcal{D}_K(N, A, C, T)$ or $\perp \leftarrow$ CLOC-$\mathcal{D}_K(N, A, C, T)$. The authenticity advantage is defined as

$$\mathbf{Adv}_{\mathrm{CLOC}[E,\ell_N,\tau]}^{\mathrm{auth}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr\left[\mathcal{A}^{\mathrm{CLOC}\text{-}\mathcal{E}_K(\cdot,\cdot,\cdot),\mathrm{CLOC}\text{-}\mathcal{D}_K(\cdot,\cdot,\cdot,\cdot)} \text{ forges}\right],$$

where the probability is taken over $K \xleftarrow{\$} \mathcal{K}_{\mathrm{CLOC}}$ and the randomness of $\mathcal{A}$, and the adversary forges if the decryption oracle returns a bit string (other than $\perp$) for a query (N, A, C, T), but (C, T) was not previously returned to $\mathcal{A}$ from the encryption oracle for a query (N, A, M). The adversary $\mathcal{A}$ in the authenticity game is not necessarily nonce-respecting, and $\mathcal{A}$ can make two or more queries with the same nonce. Specifically, $\mathcal{A}$ can repeat using the same nonce for encryption queries, a nonce used for encryption queries can be used for decryption queries and vice-versa, and the same nonce can be repeated for decryption queries. Without loss of generality, we assume that $\mathcal{A}$ does not make trivial queries, i.e., if the encryption oracle returns (C, T) for a query (N, A, M), then $\mathcal{A}$ does not make a query (N, A, C, T) to the decryption oracle, and $\mathcal{A}$ does not repeat a query.

Authenticity Theorem. Let $\mathcal{A}$ be an adversary that makes q encryption queries and q' decryption queries. Let $(N_1, A_1, M_1), \ldots, (N_q, A_q, M_q)$ be the encryption queries, and $(N'_1, A'_1, C'_1, T'_1), \ldots, (N'_{q'}, A'_{q'}, C'_{q'}, T'_{q'})$ be the decryption queries. Then we define the total associated data length in encryption queries as $a_1 + \cdots + a_q$, the total plaintext length as $m_1 + \cdots + m_q$, the total associated data length in decryption queries as $a'_1 + \cdots + a'_{q'}$, and the total ciphertext length as $m'_1 + \cdots + m'_{q'}$, where $(A_i[1], \ldots, A_i[a_i]) \overset{n}{\leftarrow} A_i$, $(M_i[1], \ldots, M_i[m_i]) \overset{n}{\leftarrow} M_i$, $(A'_i[1], \ldots, A'_i[a'_i]) \overset{n}{\leftarrow} A'_i$, and $(C'_i[1], \ldots, C'_i[m'_i]) \overset{n}{\leftarrow} C'_i$. We have the following information theoretic result.

Theorem 2. *Let* $\mathrm{Perm}(n)$*,* ℓ_N*, and* τ *be the parameters of* CLOC*. Let* $\mathcal{A}$ *be an adversary that makes at most* q *encryption queries and at most* q' *decryption queries, where the total associated data length in encryption queries is at most* σ_A*, the total plaintext length is at most* σ_M*, the total associated data length in decryption queries is at most* $\sigma_{A'}$*, and the total ciphertext length is at most* $\sigma_{C'}$*. Then we have* $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) \le 5\sigma^2_{\mathrm{auth}}/2^n + q'/2^\tau$*, where* $\sigma_{\mathrm{auth}} = q + \sigma_A + 2\sigma_M + q' + \sigma_{A'} + \sigma_{C'}$*.*

A proof overview is given in Sect. 6, and a complete proof is presented in Appendix A. As in the privacy case, if we use a blockcipher E secure in the sense of the PRP notion, then we obtain the corresponding complexity theoretic result by a standard argument in, e.g., [11].

6 Overview of Security Proofs

PRP/PRF Switching. The first step is to replace the random permutation P in $\mathrm{CLOC}[\mathrm{Perm}(n), \ell_N, \tau]$ with a random function R, and use the PRP/PRF switching lemma [12] to obtain the following differences.

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) \end{cases}$$

Defining $Q_1, \ldots, Q_{26}$ *and* CLOC2. We define twenty six functions $Q_1, \ldots, Q_{26} : \{0,1\}^n \to \{0,1\}^n$ based on R, K_1, K_2, and K_3, where $K_1, K_2, K_3 \overset{\$}{\leftarrow} \{0,1\}^n$ are three independent random n-bit strings. We also define a modified version of $\mathrm{CLOC}[\mathrm{Rand}(n), \ell_N, \tau]$ called $\mathrm{CLOC2}[\ell_N, \tau]$, which uses $Q = (Q_1, \ldots, Q_{26})$ as oracles. Q and CLOC2 are designed so that CLOC-$\mathcal{E}_R$ and CLOC2-$\mathcal{E}_Q$ are the same algorithms, CLOC-$\mathcal{D}_R$ and CLOC2-$\mathcal{D}_Q$ are the same algorithms (except that CLOC2-$\mathcal{D}_Q$ is used for the verification only, and it does not output a plaintext even if the verification succeeds), and $Q_1, \ldots, Q_{26}$ are indistinguishable from $F_1, \ldots, F_{26}$, which are independent random functions. We then have

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}), \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}), \end{cases}$$

and we show the distinguishing probability of $Q = (Q_1, \ldots, Q_{26})$ and $F = (F_1, \ldots, F_{26})$ in Lemma 1. However, the indistinguishability does not hold for arbitrary adversaries. We formalize an input-respecting adversary, and our indistinguishability result in Lemma 1 holds only for these adversaries.

The three random strings, K_1, K_2, and K_3, are secret keys from the adversary's perspective, and we introduce them to show the indistinguishability between Q and F. For instance we know that the input $\mathsf{fix0}(\mathsf{ozp}(A[1]))$ to produce $S_{\mathsf{H}}[1]$ in $\mathsf{HASH}_K(N, A)$ (The 2nd line of $\mathsf{HASH}_K(N, A)$ in Fig. 2) never collides with the input $\mathsf{fix1}(C[i])$ to produce $S_{\mathsf{E}}[i+1]$ in $\mathsf{ENC}_K(V, M)$ (The 8th line of $\mathsf{ENC}_K(V, M)$ in Fig. 2), and hence we can safely assume that they are independent. Likewise, we show that the collision probability between $\mathsf{fix0}(\mathsf{ozp}(A[1]))$ and, say, $S_{\mathsf{H}}[i-1] \oplus A[i]$ in $\mathsf{HASH}_K(N, A)$ (The 7th line of $\mathsf{HASH}_K(N, A)$ in Fig. 2) is low, and the three random strings are introduced to help this argument.

Defining CLOC3. We define another version of $\mathrm{CLOC}[\mathrm{Rand}(n), \ell_N, \tau]$ called $\mathrm{CLOC3}[\ell_N, \tau]$. It uses $F = (F_1, \ldots, F_{26})$ as oracles, and the encryption algorithm CLOC3-$\mathcal{E}_F$ and the decryption algorithm CLOC3-$\mathcal{D}_F$ are obtained from CLOC2-$\mathcal{E}_Q$ and CLOC2-$\mathcal{D}_Q$ by replacing $Q_1, \ldots, Q_{26}$ with $F_1, \ldots, F_{26}$, respectively. We use Lemma 1 to obtain the following differences.

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) \end{cases}$$

The simulations work with input-respecting adversaries, and hence Lemma 1 is sufficient for our purpose.

Indistinguishability of $(\mathsf{HASH3}, \mathsf{HASH3}', \mathsf{HASH3}'')$. We then consider three subroutines $\mathsf{HASH3}$, $\mathsf{HASH3}'$, and $\mathsf{HASH3}''$ in CLOC3$[\ell_N, \tau]$. $\mathsf{HASH3}$ roughly corresponds to a function that computes $S_{\mathsf{E}}[1]$ from (N, A) in CLOC$[E, \ell_N, \tau]$, i.e., $E_K(\mathsf{HASH}_K(N, A))$. $\mathsf{HASH3}'$ computes the tag T when $|C| = 0$, i.e., this function roughly corresponds to $\mathsf{msb}_\tau(E_K(\mathsf{g}_1(\mathsf{HASH}_K(N, A))))$. $\mathsf{HASH3}''$ computes $S_{\mathsf{P}}[0]$ from (N, A), which is used when $|C| \geq 1$, i.e., $E_K(\mathsf{g}_2(\mathsf{HASH}_K(N, A)))$. Then in Lemma 2, we show that these functions are indistinguishable from three independent random functions $\mathsf{HASH4}$, $\mathsf{HASH4}'$, and $\mathsf{HASH4}''$.

Defining CLOC4. We define another version of CLOC$[\mathrm{Rand}(n), \ell_N, \tau]$, called CLOC4$[\ell_N, \tau]$. This is obtained by replacing $\mathsf{HASH3}$, $\mathsf{HASH3}'$, and $\mathsf{HASH3}''$ in CLOC3 with $\mathsf{HASH4}$, $\mathsf{HASH4}'$, and $\mathsf{HASH4}''$, respectively. We use Lemma 2 to obtain the following differences.

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) \end{cases}$$

Indistinguishability of $\mathsf{PRF4}$. We then consider a subroutine called $\mathsf{PRF4}$ in CLOC4. This function outputs a tag T from (N, A, C), and internally uses $\mathsf{HASH4}'$, $\mathsf{HASH4}''$, F_{24}, F_{25}, and F_{26}. We show in Lemma 3 that this function is indistinguishable from a random function $\mathsf{PRF5}$.

Defining CLOC5. We define our final version of CLOC$[\mathrm{Rand}(n), \ell_N, \tau]$, called CLOC5$[\ell_N, \tau]$, which is obtained from CLOC4 by replacing $\mathsf{PRF4}$ with $\mathsf{PRF5}$. This function is used in both encryption and decryption, and we obtain the following differences from Lemma 3.

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) - \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \end{cases}$$

Privacy and Authenticity of CLOC5. Finally, we analyze the privacy and the authenticity of CLOC5 in Lemma 4. The privacy result shows the upper bound on $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A})$, and the proof is reduced to bounding the collision probability among the input values of the random function which is used to encrypt plaintexts. The authenticity result shows the upper bound on $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A})$, and its proof is simple and the result is obtained from the fact that the adversary, even if the nonce is reused, has to guess the output of a random function $\mathsf{PRF5}$ for the input that was not queried before.

We finally obtain the proofs of Theorem 1 and Theorem 2 by combining the above differences between advantage functions.

7 Software Implementation

We first tested CLOC on a general-purpose CPU. It is interesting to note that the encryption process and tag generation can be done in parallel, which could speed up the overall computation by a factor close to 2 for long messages, then the final speed could be close to that of encryption only in serial mode. To show that, we implemented CLOC instantiated with AES-128 using the AES new instruction set, and tested against Intel processor, Core i5-3427U 1.80GHz [6]. It is known that Intel's AES instruction allows fast parallel processing (up to 4 or 8 blocks), and we used this technique for two parallel inputs to AES. The tested speed is around 4.9 cycles per byte (cpb), while AES-128 encrypts at a speed of 4.3 cpb in serial mode. In Table 2, we provide the test vectors.

We then tested CLOC on embedded software. We used an 8-bit microprocessor, Atmel AVR ATmega128 [2]. For comparison we also implemented EAX and OCB3 [26]. For OCB3 we used a byte-oriented code from [7]. OCB3 needs relatively large precomputation for GF doublings, but we modify the code so that the doublings are on-line, since large precomputation may not be suitable to handle short input data for microprocessors. We also considered GCM for comparison, however, recent studies show that GCM does not perform well on constrained devices (see e.g. [10,38]), hence we decided not to include it. All modes are written in C and combined with AES-128. Our AES code is taken from [3], which is written in assembler. AES runs at 156.7 cpb for encryption, 196.8 cpb for decryption, both without key scheduling, and the key scheduling runs at 1,979 cycles. Our codes are complied with Atmel Studio 6 available from [2]. Cycles counts are measured on the simulator of Atmel Studio 6. Table 3 shows the implementation result. ROM denotes the object size in bytes. The speed is measured based

Table 2. Test vector of CLOC instantiated with AES-128

	length (bytes)	value (in hex)
Key	16	`000102030405060708090a0b0c0d0e0f0`
Associated data	14	`ff0102030405060708090a0b0c0d`
Nonce	12	`00112233445566778899aabb`
Plaintext	30	`86012204ccebf09ad5305ea8967aebd0`
		`0dd9c05cbde9407ff1ef52f043a2`
Ciphertext	30	`ebd908c23eac555dee406434fb2cffd4`
		`e1bee4401002063e2d13cdf9df3b`
Tag	16	`6621dae27674aa6fbc303426824b2c05`

Table 3. Software implementation on ATmega128

	ROM	RAM	Init	Speed (cycles/byte)					
	(bytes)	(bytes)	(cycles)	Data 16	32	64	96	128	256
CLOC	2980	362	1999	750.1	549.0	448.4	404.9	398.2	373.0
EAX	3304	n/a	2617	1842.4	1094.1	719.8	595.1	532.7	439.1
OCB-E	5010	971	4956	1217.5	736.1	495.5	412.2	375.1	314.9
OCB-D	5010	971	4956	1252.2	773.4	534.0	451.2	414.3	354.4

on the scenario of non-static associated data, i.e., we excluded key setup and other computations before processing associated data and nonce, defined as "Init", and figures for Data b denote cycles per byte to process a b-byte message with 16-byte associated data. For OCB3 we also measured the decryption performance, whereas those of CLOC and EAX are almost the same as encryption. The result shows a superior performance of CLOC for short input data, up to around 128 bytes, which would be sufficiently long for low-power wireless networks, as we mentioned in Sect. 1. We also measure the RAM usage of the AVR implementations, using a public tool [41], based on data of 16 bytes. It is clear to see that CLOC requires much less RAM than OCB3. Due to an unknown reason, this tool is not able to tell the RAM usage for EAX.

8 Hardware Implementation

Although the primary focus of CLOC is embedded software, we also implemented CLOC on hardware to see basic performance figures. We used Altera FPGA, Cyclone IV GX (EP4CGX110DF31C7) [1], and implemented CLOC using AES-128. AES implementation is round-based, and the S-box of AES is based on a composite field [37]. For reference we also wrote EAX for the same device, using the same AES. Both CLOC and EAX use one AES core for encryption and authentication. In EAX implementation, all input masks are stored to registers. Table 4 shows the results. The size is measured by the number of logic elements (LEs). Our implementation is not optimized. Still, these figures show that CLOC has slightly smaller size and faster speed than EAX. Table 4 lacks other important modes, in particular OCB. A more comprehensive comparison and optimized implementation for short input data are interesting future topics.

9 Conclusions

We presented a blockcipher mode of operation called CLOC for authenticated encryption with associated data. It uses a variant of CFB mode in its encryption part and a variant of CBC MAC in the authentication part. The scheme efficiently handles short input data without heavy precomputation nor large memory, and it is suitable for use in microprocessors. We proved CLOC secure, in a reduction-based provable security paradigm, under the assumption that the blockcipher is a pseudorandom permutation. We also presented our preliminary implementation results.

It would be interesting to see improved implementation results using possibly lightweight blockciphers.

Table 4. Hardware implementation. Throughput figures of CLOC and EAX are measured for 8-block messages with one-block associated data.

	Size (LE)	Max. Freq. (MHz)	Throughput (Mbit/sec)
CLOC	5628	82.1	400.7
EAX	6453	61.3	342.2
AES Enc	3175	98.7	971.7

Acknowledgments. The authors would like to thank the anonymous FSE 2014 reviewers for helpful comments. The work by Tetsu Iwata was carried out in part while visiting Nanyang Technological University, Singapore. The work by Jian Guo was supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

1. Altera Corporation, `http://www.altera.com/`
2. ATMEL Corporation, `http://www.atmel.com/`
3. AVR-Crypto-Lib, `http://www.das-labor.org/wiki/AVR-Crypto-Lib/en/`
4. Bluetooth low energy, `http://www.bluetooth.com/Pages/Low-Energy.aspx/`
5. Electronic Product Code (EPC) Tag Data Standard (TDS), `http://www.epcglobalinc.org/standards/tds/`
6. Intel Corporation, `http://www.intel.com/`
7. OCB News and Code, `http://www.cs.ucdavis.edu/~rogaway/ocb/news/`
8. ZigBee Alliance, `http://www.zigbee.org/`
9. Information Technology — Security Techniques — Authenticated Encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009)
10. Bauer, G.R., Potisk, P., Tillich, S.: Comparing Block Cipher Modes of Operation on MICAz Sensor Nodes. In: Baz, D.E., Spies, F., Gross, T. (eds.) PDP. pp. 371–378. IEEE Computer Society (2009)
11. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. 61(3), 362–399 (2000)
12. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer (2006)
13. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy and Meier [36], pp. 389–407
14. Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: FIDES: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In: Bertoni, G., Coron, J.S. (eds.) CHES. Lecture Notes in Computer Science, vol. 8086, pp. 142–158. Springer (2013)
15. Black, J., Rogaway, P.: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. J. Cryptology 18(2), 111–131 (2005)
16. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption. Pre-proceedings of FSE 2013 (2013)
17. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C (2004)
18. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)
19. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: Canteaut, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 7549, pp. 196–215. Springer (2012)
20. Fouque, P.A., Martinet, G., Valette, F., Zimmer, S.: On the Security of the CCM Encryption Mode and of a Slight Variant. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS. Lecture Notes in Computer Science, vol. 5037, pp. 411–428 (2008)
21. Housley, R.: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). IETF RFC 4309 (2005)
22. Housley, R.: Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS). IETF RFC 5084 (2007)
23. Iwata, T., Minematsu, K.: Generating a Fixed Number of Masks with Word Permutations and XORs. DIAC: Directions in Authenticated Ciphers (2013), `http://2013.diac.cr.yp.to/`
24. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and Repairing GCM Security Proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 31–49. Springer (2012)

25. Jonsson, J.: On the Security of CTR + CBC-MAC. In: Nyberg, K., Heys, H.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2595, pp. 76–93. Springer (2002)
26. 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)
27. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
28. Lucks, S.: Two-Pass Authenticated Encryption Faster Than Generic Composition. In: Gilbert, H., Handschuh, H. (eds.) FSE. Lecture Notes in Computer Science, vol. 3557, pp. 284–298. Springer (2005)
29. 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)
30. 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`
31. Moise, A., Beroset, E., Phinney, T., Burns, M.: EAX' Cipher Mode (May 2011). NIST Submission (2011), `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax-prime/eax-prime-spec.pdf`
32. Nandi, M.: Fast and Secure CBC-Type MAC Algorithms. In: Dunkelman, O. (ed.) FSE. Lecture Notes in Computer Science, vol. 5665, pp. 375–393. Springer (2009)
33. 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)
34. Rogaway, P.: Nonce-Based Symmetric Encryption. In: Roy and Meier [36], pp. 348–359
35. Rogaway, P., Wagner, D.: A Critique of CCM (2003), `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/CCM/RW_CCM_comments.pdf`
36. 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)
37. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 2248, pp. 239–254. Springer (2001)
38. Simplício Jr., M.A., de Oliveira, B.T., Barreto, P.S.L.M., Margi, C.B., Carvalho, T.C.M.B., Näslund, M.: Comparison of Authenticated-Encryption schemes in Wireless Sensor Networks. In: Chou, C.T., Pfeifer, T., Jayasumana, A.P. (eds.) LCN. pp. 450–457. IEEE (2011)
39. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC. Submission to NIST (2002), `http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html`
40. Zhang, L., Wu, W., Zhang, L., Wang, P.: CBCR: CBC MAC with rotating transformations. SCIENCE CHINA Information Sciences 54(11), 2247–2255 (2011)
41. Zharkov, E.: EZSTACK: A tool to measure the RAM usage of AVR implementations. Published online `http://home.comcast.net/~ezstack/ezstack.c`

A Security Proofs of CLOC

PRP/PRF Switching. We first replace P in CLOC[Perm$(n), \ell_N, \tau$] with a random function $R \xleftarrow{\$} \mathrm{Rand}(n)$. From the PRP/PRF switching lemma [12], we have

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) + \frac{0.5\sigma_{\mathrm{priv}}^2}{2^n}, \tag{2}$$

since for a query (N_i, A_i, M_i), we need $\lceil |N_i|/n \rceil + \max\{1, \lceil |A_i|/n \rceil\} + 2\lceil |M_i|/n \rceil \le 1 + a_i + 2m_i$ calls of P in CLOC-$\mathcal{E}_P$, and we have $\sum_{1\le i\le q}(1 + a_i + 2m_i) \le \sigma_{\mathrm{priv}}$. For the authenticity notion, without loss of generality, we assume that the decryption oracle, if $\mathcal{A}$ succeeds in forgery, returns a bit 1 instead of the plaintext since the returned value has no effect on the success probability of $\mathcal{A}$. Then for a decryption query (N'_j, A'_j, C'_j, T'_j), CLOC-$\mathcal{D}_P$ makes $\lceil |N'_j|/n \rceil + \max\{1, \lceil |A'_j|/n \rceil\} + \lceil |C'_j|/n \rceil \le 1 + a'_j + m'_j$ calls of P, and from $\sum_{1\le i\le q}(1 + a_i + 2m_i) + \sum_{1\le j\le q'}(1 + a'_j + m'_j) \le \sigma_{\mathrm{auth}}$, we obtain

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) + \frac{0.5\sigma_{\mathrm{auth}}^2}{2^n}. \tag{3}$$

In what follows, we evaluate $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A})$ and $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A})$.

Definition of $Q_1, \ldots, Q_{26}$. Let $R \xleftarrow{\$} \mathrm{Rand}(n)$ be a random function, and let $K_1, K_2, K_3 \xleftarrow{\$} \{0,1\}^n$ be three independent random n-bit strings. We define twenty six functions $Q_1, \ldots, Q_{26} : \{0,1\}^n \to \{0,1\}^n$ based on R, K_1, K_2, and K_3 in Fig 11, and we write $Q = (Q_1, \ldots, Q_{26})$.

$Q_1(X) = R(\mathsf{fix0}(X)) \oplus K_1$	$Q_{14}(X) = R(\mathsf{f}_2(\mathsf{h}(X \oplus K_1)))$
$Q_2(X) = R(X \oplus K_1) \oplus K_2$	$Q_{15}(X) = R(\mathsf{f}_1(X \oplus K_2))$
$Q_3(X) = R(\mathsf{h}(X \oplus K_1)) \oplus K_2$	$Q_{16}(X) = R(\mathsf{f}_2(X \oplus K_2))$
$Q_4(X) = R(X \oplus K_2) \oplus K_2$	$Q_{17}(X) = R(\mathsf{fix1}(X))$
$Q_5(X) = R(\mathsf{g}_1(\mathsf{f}_1(X \oplus K_1)))$	$Q_{18}(X) = R(\mathsf{g}_2(\mathsf{f}_1(X \oplus K_1))) \oplus K_3$
$Q_6(X) = R(\mathsf{g}_1(\mathsf{f}_1(\mathsf{h}(X \oplus K_1))))$	$Q_{19}(X) = R(\mathsf{g}_2(\mathsf{f}_1(\mathsf{h}(X \oplus K_1)))) \oplus K_3$
$Q_7(X) = R(\mathsf{g}_1(\mathsf{f}_2(X \oplus K_1)))$	$Q_{20}(X) = R(\mathsf{g}_2(\mathsf{f}_2(X \oplus K_1))) \oplus K_3$
$Q_8(X) = R(\mathsf{g}_1(\mathsf{f}_2(\mathsf{h}(X \oplus K_1))))$	$Q_{21}(X) = R(\mathsf{g}_2(\mathsf{f}_2(\mathsf{h}(X \oplus K_1)))) \oplus K_3$
$Q_9(X) = R(\mathsf{g}_1(\mathsf{f}_1(X \oplus K_2)))$	$Q_{22}(X) = R(\mathsf{g}_2(\mathsf{f}_1(X \oplus K_2))) \oplus K_3$
$Q_{10}(X) = R(\mathsf{g}_1(\mathsf{f}_2(X \oplus K_2)))$	$Q_{23}(X) = R(\mathsf{g}_2(\mathsf{f}_2(X \oplus K_2))) \oplus K_3$
$Q_{11}(X) = R(\mathsf{f}_1(X \oplus K_1))$	$Q_{24}(X) = R(X \oplus K_3) \oplus K_3$
$Q_{12}(X) = R(\mathsf{f}_1(\mathsf{h}(X \oplus K_1)))$	$Q_{25}(X) = R(\mathsf{f}_1(X \oplus K_3))$
$Q_{13}(X) = R(\mathsf{f}_2(X \oplus K_1))$	$Q_{26}(X) = R(\mathsf{f}_2(X \oplus K_3))$

Fig. 11. Definition of $Q_1, \ldots, Q_{26}$

```
Algorithm CLOC2-$\mathcal{E}_Q(N, A, M)$

1. if $|M| = 0$ then
2.     $C \leftarrow \varepsilon$
3. else                                              // $|M| \geq 1$
4.     $S_\mathsf{E}[1] \leftarrow \mathsf{HASH2}_{Q_1,\ldots,Q_4,Q_{11},\ldots,Q_{16}}(N, A)$
5.     $C \leftarrow \mathsf{ENC2}_{Q_{17}}(S_\mathsf{E}[1], M)$
6. $T \leftarrow \mathsf{PRF2}_{Q_1,\ldots,Q_{10},Q_{18},\ldots,Q_{26}}(N, A, C)$
7. return $(C, T)$
```

```
Algorithm CLOC2-$\mathcal{D}_Q(N, A, C, T)$

1. $T^* \leftarrow \mathsf{PRF2}_{Q_1,\ldots,Q_{10},Q_{18},\ldots,Q_{26}}(N, A, C)$
2. if $T \neq T^*$ then return $\perp$
3. return 1
```

Fig. 12. Pseudocode of the encryption and the decryption algorithms of CLOC2

Definition of CLOC2. We next define a version of CLOC[Rand(n), ℓ_N, τ] which we write CLOC2[ℓ_N, τ]. It is based on Q, and the encryption algorithm CLOC2-$\mathcal{E}$ and the decryption algorithm CLOC2-$\mathcal{D}$ are presented in Fig. 12 and Fig. 13. We also show figures of some of the subroutines used in these algorithms in Fig. 14, Fig. 15, Fig. 16, and Fig. 17. CLOC2-$\mathcal{E}$ and CLOC2-$\mathcal{D}$ take R, K_1, K_2, and K_3 as a key, but we write CLOC2-$\mathcal{E}_Q$ and CLOC2-$\mathcal{D}_Q$, and we describe them using $Q = (Q_1, \ldots, Q_{26})$ as subroutines.

We briefly describe the intuition how CLOC2 works. There are three main subroutines, HASH2, ENC2, and PRF2.

- HASH2 takes N and A as input, and is used when $|M| \geq 1$ to generate the first mask to encrypt $M[1]$, i.e., $S_\mathsf{E}[1]$.
- ENC2 takes $S_\mathsf{E}[1]$ and M as input and it returns the ciphertext C.
- PRF2 takes N, A, and C as input to return the tag T. This function is different from PRF in that, PRF takes V as a part of the input, which is a value that can be computed from N and A and is independent of C, but PRF2 directly computes the tag from N, A, and C. We use three functions in PRF2, which we call HASH2$'$, HASH2$''$, and PRF2$'$, and these functions are described below.
  - HASH2$'$ is used when $|C| = 0$ to generate a tag T from N and A.
  - HASH2$''$ and PRF2$'$ are used when $|C| \geq 1$ to generate a tag T. HASH2$''$ takes N and A as input and computes a value that roughly corresponds to $E_K(\mathsf{g}_2(V))$ of CLOC. Then PRF2$'$ takes the output of HASH2$''$ and C as input to output the tag T.

We note that HASH2, HASH2$'$, and HASH2$''$ are almost the same algorithms except for the functions used to process the last input block (and hence they share a similar name). We use the name PRF2 as it is used to generate a tag, while we use the name PRF2$'$ as it roughly corresponds to PRF.

Now with careful scrutiny, we can verify that CLOC-$\mathcal{E}_R$ and CLOC2-$\mathcal{E}_Q$ are exactly the same algorithms, and furthermore, CLOC-$\mathcal{D}_R$ and CLOC2-$\mathcal{D}_Q$ are the same algorithms, since K_1, K_2, and K_3 are

```
Algorithm $\mathsf{HASH2}_{Q_1,\ldots,Q_4,Q_{11},\ldots,Q_{16}}(N,A)$

 1. $(A[1],\ldots,A[a]) \overset{n}{\leftarrow} A$
 2. $S_{\mathsf{H}}[1] \leftarrow Q_1(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
 3. if $0 \le |A[1]| \le n-1$ then
 4.     if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 0$ then
 5.         $S_{\mathsf{E}}[1] \leftarrow Q_{13}(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N))$
 6.     else                          // $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$
 7.         $S_{\mathsf{E}}[1] \leftarrow Q_{14}(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N)))$
 8. if $|A[1]| = n$ then
 9.     if $\mathsf{msb}_1(A[1]) = 0$ then
10.         $S_{\mathsf{E}}[1] \leftarrow Q_{11}(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N))$
11.     else                          // $\mathsf{msb}_1(A[1]) = 1$
12.         $S_{\mathsf{E}}[1] \leftarrow Q_{12}(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N)))$
13. if $|A| \ge n+1$ then
14.     if $\mathsf{msb}_1(A[1]) = 0$ then
15.         $S_{\mathsf{H}}[2] \leftarrow Q_2(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$
16.     else                          // $\mathsf{msb}_1(A[1]) = 1$
17.         $S_{\mathsf{H}}[2] \leftarrow Q_3(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(A[2])))$
18.     if $a \ge 3$ then
19.         for $i \leftarrow 3$ to $a-1$ do   // only for $a \ge 4$
20.             $S_{\mathsf{H}}[i] \leftarrow Q_4(S_{\mathsf{H}}[i-1] \oplus A[i])$
21.         $S_{\mathsf{H}}[a] \leftarrow Q_4(S_{\mathsf{H}}[a-1] \oplus \mathsf{ozp}(A[a]))$
22.     if $|A[a]| = n$ then
23.         $S_{\mathsf{E}}[1] \leftarrow Q_{15}(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
24.     else                          // $1 \le |A[a]| \le n-1$
25.         $S_{\mathsf{E}}[1] \leftarrow Q_{16}(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
26. return $S_{\mathsf{E}}[1]$
```

```
Algorithm $\mathsf{ENC2}_{Q_{17}}(S_{\mathsf{E}}[1], M)$          // $|M| \ge 1$

 1. $(M[1],\ldots,M[m]) \overset{n}{\leftarrow} M$
 2. for $i \leftarrow 1$ to $m-1$ do          // only for $m \ge 2$
 3.     $C[i] \leftarrow S_{\mathsf{E}}[i] \oplus M[i]$
 4.     $S_{\mathsf{E}}[i+1] \leftarrow Q_{17}(\mathsf{fix1}(C[i]))$
 5. $C[m] \leftarrow \mathsf{msb}_{|M[m]|}(S_{\mathsf{E}}[m]) \oplus M[m]$
 6. $C \leftarrow (C[1],\ldots,C[m])$
 7. return $C$
```

```
Algorithm $\mathsf{PRF2}_{Q_1,\ldots,Q_{10},Q_{18},\ldots,Q_{26}}(N,A,C)$

 1. if $|C| = 0$ then
 2.     $T \leftarrow \mathsf{HASH2}'_{Q_1,\ldots,Q_{10}}(N,A)$
 3. else                                    // $|C| \ge 1$
 4.     $S_{\mathsf{P}}[0] \leftarrow \mathsf{HASH2}''_{Q_1,\ldots,Q_4,Q_{18},\ldots,Q_{23}}(N,A)$
 5.     $T \leftarrow \mathsf{PRF2}'_{Q_{24},Q_{25},Q_{26}}(S_{\mathsf{P}}[0],C)$
 6. return $T$
```

```
Algorithm $\mathsf{PRF2}'_{Q_{24},Q_{25},Q_{26}}(S_{\mathsf{P}}[0],C)$        // $|C| \ge 1$

 1. $(C[1],\ldots,C[m]) \overset{n}{\leftarrow} C$
 2. for $i \leftarrow 1$ to $m-1$ do          // only for $m \ge 2$
 3.     $S_{\mathsf{P}}[i] \leftarrow Q_{24}(S_{\mathsf{P}}[i-1] \oplus C[i])$
 4. if $|C[m]| = n$ then
 5.     $S_{\mathsf{P}}[m] \leftarrow Q_{25}(S_{\mathsf{P}}[m-1] \oplus C[m])$
 6. else                          // $1 \le |C[m]| \le n-1$
 7.     $S_{\mathsf{P}}[m] \leftarrow Q_{26}(S_{\mathsf{P}}[m-1] \oplus \mathsf{ozp}(C[m]))$
 8. $T \leftarrow \mathsf{msb}_{\tau}(S_{\mathsf{P}}[m])$
 9. return $T$
```

```
Algorithm $\mathsf{HASH2}'_{Q_1,\ldots,Q_{10}}(N,A)$

 1. $(A[1],\ldots,A[a]) \overset{n}{\leftarrow} A$
 2. $S_{\mathsf{H}}[1] \leftarrow Q_1(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
 3. if $0 \le |A[1]| \le n-1$ then
 4.     if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 0$ then
 5.         $T \leftarrow \mathsf{msb}_{\tau}(Q_7(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N)))$
 6.     else                          // $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$
 7.         $T \leftarrow \mathsf{msb}_{\tau}(Q_8(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N))))$
 8. if $|A[1]| = n$ then
 9.     if $\mathsf{msb}_1(A[1]) = 0$ then
10.         $T \leftarrow \mathsf{msb}_{\tau}(Q_5(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N)))$
11.     else                          // $\mathsf{msb}_1(A[1]) = 1$
12.         $T \leftarrow \mathsf{msb}_{\tau}(Q_6(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N))))$
13. if $|A| \ge n+1$ then
14.     if $\mathsf{msb}_1(A[1]) = 0$ then
15.         $S_{\mathsf{H}}[2] \leftarrow Q_2(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$
16.     else                          // $\mathsf{msb}_1(A[1]) = 1$
17.         $S_{\mathsf{H}}[2] \leftarrow Q_3(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(A[2])))$
18.     if $a \ge 3$ then
19.         for $i \leftarrow 3$ to $a-1$ do   // only for $a \ge 4$
20.             $S_{\mathsf{H}}[i] \leftarrow Q_4(S_{\mathsf{H}}[i-1] \oplus A[i])$
21.         $S_{\mathsf{H}}[a] \leftarrow Q_4(S_{\mathsf{H}}[a-1] \oplus \mathsf{ozp}(A[a]))$
22.     if $|A[a]| = n$ then
23.         $T \leftarrow \mathsf{msb}_{\tau}(Q_9(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N)))$
24.     else                          // $1 \le |A[a] \le n-1$
25.         $T \leftarrow \mathsf{msb}_{\tau}(Q_{10}(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N)))$
26. return $T$
```

```
Algorithm $\mathsf{HASH2}''_{Q_1,\ldots,Q_4,Q_{18},\ldots,Q_{23}}(N,A)$

 1. $(A[1],\ldots,A[a]) \overset{n}{\leftarrow} A$
 2. $S_{\mathsf{H}}[1] \leftarrow Q_1(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
 3. if $0 \le |A[1]| \le n-1$ then
 4.     if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 0$ then
 5.         $S_{\mathsf{P}}[0] \leftarrow Q_{20}(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N))$
 6.     else                          // $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$
 7.         $S_{\mathsf{P}}[0] \leftarrow Q_{21}(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N)))$
 8. if $|A[1]| = n$ then
 9.     if $\mathsf{msb}_1(A[1]) = 0$ then
10.         $S_{\mathsf{P}}[0] \leftarrow Q_{18}(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N))$
11.     else                          // $\mathsf{msb}_1(A[1]) = 1$
12.         $S_{\mathsf{P}}[0] \leftarrow Q_{19}(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N)))$
13. if $|A| \ge n+1$ then
14.     if $\mathsf{msb}_1(A[1]) = 0$ then
15.         $S_{\mathsf{H}}[2] \leftarrow Q_2(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$
16.     else                          // $\mathsf{msb}_1(A[1]) = 1$
17.         $S_{\mathsf{H}}[2] \leftarrow Q_3(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(A[2])))$
18.     if $a \ge 3$ then
19.         for $i \leftarrow 3$ to $a-1$ do   // only for $a \ge 4$
20.             $S_{\mathsf{H}}[i] \leftarrow Q_4(S_{\mathsf{H}}[i-1] \oplus A[i])$
21.         $S_{\mathsf{H}}[a] \leftarrow Q_4(S_{\mathsf{H}}[a-1] \oplus \mathsf{ozp}(A[a]))$
22.     if $|A[a]| = n$ then
23.         $S_{\mathsf{P}}[0] \leftarrow Q_{22}(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
24.     else                          // $1 \le |A[a] \le n-1$
25.         $S_{\mathsf{P}}[0] \leftarrow Q_{23}(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
26. return $S_{\mathsf{P}}[0]$
```

Fig. 13. Subroutines used in the encryption and decryption algorithms of CLOC2

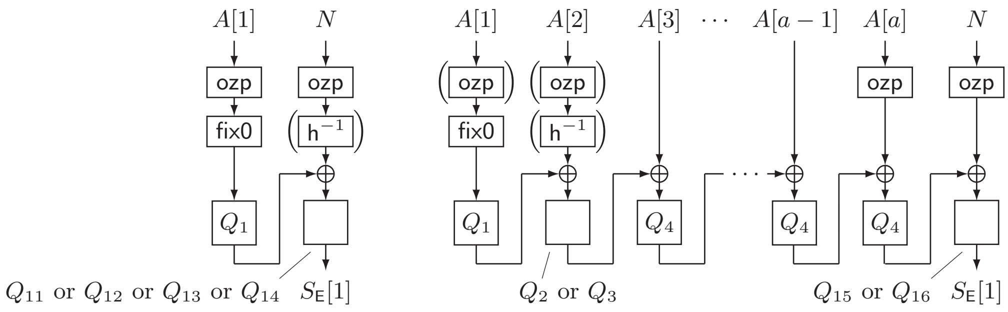


Fig. 14. $S_{\mathsf{E}}[1] \leftarrow \mathsf{HASH2}_{Q_1,\ldots,Q_4,Q_{11},\ldots,Q_{16}}(N, A)$ for $0 \le |A| \le n$ (left) and $|A| \ge n+1$ (right). The functions in parenthesis may not be executed depending on the input.

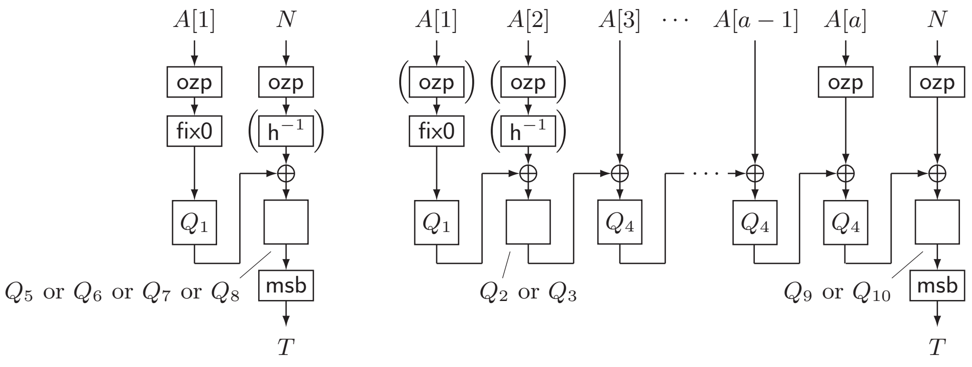


Fig. 15. $T \leftarrow \mathsf{HASH2}'_{Q_1,\ldots,Q_{10}}(N, A)$ for $0 \le |A| \le n$ (left) and $|A| \ge n+1$ (right). This function is used as a subroutine in $\mathsf{PRF2}$ to generate a tag T when $|C| = 0$.

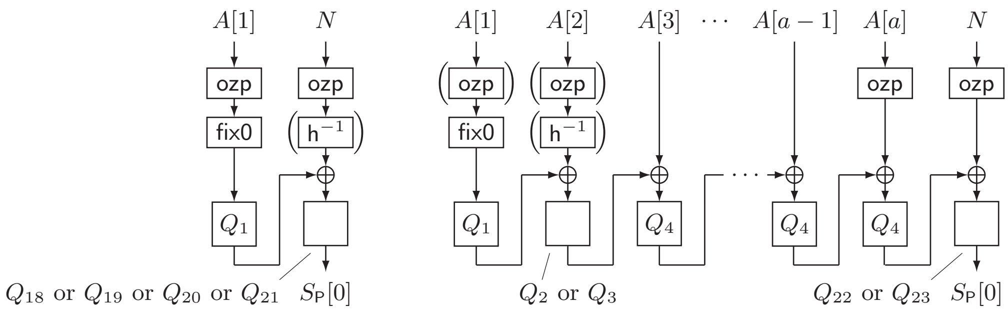


Fig. 16. $T \leftarrow \mathsf{HASH2}''_{Q_1,\ldots,Q_4,Q_{18},\ldots,Q_{23}}(N, A)$ for $0 \le |A| \le n$ (left) and $|A| \ge n+1$ (right). This function is used in $\mathsf{PRF2}$ when $|C| \ge 1$.

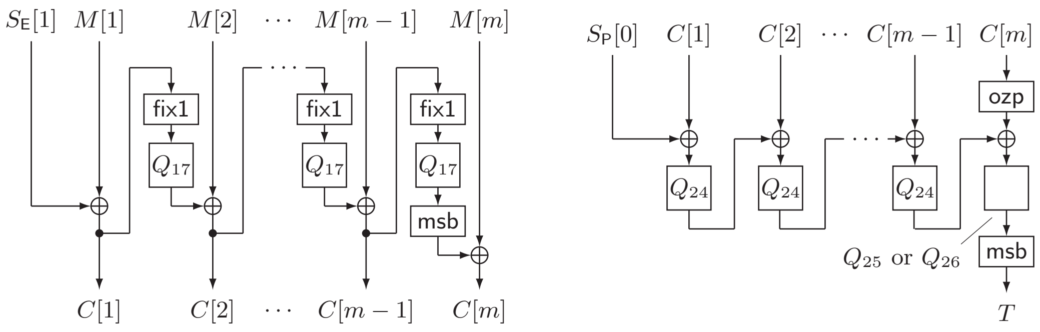


Fig. 17. $C \leftarrow \mathsf{ENC2}_{Q_{17}}(S_{\mathsf{E}}[1], M)$ for $|M| \ge 1$ (left), and $\mathsf{PRF2}'_{Q_{24},Q_{25},Q_{26}}(S_{\mathsf{P}}[0], C)$ for $|C| \ge 1$ (right). $\mathsf{PRF2}'$ is used as a subroutine in $\mathsf{PRF2}$, together with $\mathsf{HASH2}''$, to generate a tag T when $|C| \ge 1$.

all canceled. We therefore have

$$\begin{cases}\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}),\\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]}(\mathcal{A}) = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}).\end{cases} \tag{4}$$

Indistinguishability of Q. Next, let $F_1,\ldots,F_{26} \xleftarrow{\$} \mathrm{Rand}(n)$ be twenty six independent random functions, and we write $F = (F_1,\ldots,F_{26})$. We show that $Q = (Q_1,\ldots,Q_{26})$ is indistinguishable from $F = (F_1,\ldots,F_{26})$. For an adversary $\mathcal{B}$, we define $\mathbf{Adv}^{\mathrm{ind}}_{Q}(\mathcal{B})$ as

$$\mathbf{Adv}^{\mathrm{ind}}_{Q}(\mathcal{B}) \stackrel{\mathrm{def}}{=} \Pr\left[\mathcal{B}^{Q_1(\cdot),\ldots,Q_{26}(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{B}^{F_1(\cdot),\ldots,F_{26}(\cdot)} \Rightarrow 1\right],$$

where the first probability is taken over $R \xleftarrow{\$} \mathrm{Rand}(n)$, $K_1,K_2,K_3 \xleftarrow{\$} \{0,1\}^n$, and the randomness of $\mathcal{B}$, and the last is over $F_1,\ldots,F_{26} \xleftarrow{\$} \mathrm{Rand}(n)$ and $\mathcal{B}$. The adversary makes queries of the form $(j,X) \in \{1,\ldots,26\}\times\{0,1\}^n$, and receives $Q_j(X)$ or $F_j(X)$. We say that the adversary is input-respecting if $\mathsf{msb}_1(X) = 0$ holds for all queries with $j = 1$, and $\mathsf{msb}_1(X) = 1$ holds for all queries with $j = 17$. Without loss of generality, we assume that $\mathcal{B}$ does not repeat a query. We have the following lemma.

Lemma 1. *Let $\mathcal{B}$ be an input-respecting adversary that makes at most q queries. Then $\mathbf{Adv}^{\mathrm{ind}}_{Q}(\mathcal{B}) \le 0.5q^2/2^n$.*

A proof is in Appendix B.

Definition of CLOC3. We define another version of $\mathrm{CLOC}[\mathrm{Rand}(n),\ell_N,\tau]$ which we write $\mathrm{CLOC3}[\ell_N,\tau]$. It is based on F, and the encryption algorithm CLOC3-$\mathcal{E}$ and the decryption algorithm CLOC3-$\mathcal{D}$ are the same as CLOC2-$\mathcal{E}$ and CLOC2-$\mathcal{D}$, except that we use $F_1,\ldots,F_{26}$ instead of $Q_1,\ldots,Q_{26}$, respectively. Therefore, CLOC3-$\mathcal{E}$ and CLOC3-$\mathcal{D}$ take $F = (F_1,\ldots,F_{26})$ as a key, and we write CLOC3-$\mathcal{E}_F$ and CLOC3-$\mathcal{D}_F$. We write the subroutines in CLOC3-$\mathcal{E}_F$ and CLOC3-$\mathcal{D}_F$ as $\mathsf{HASH3}$, $\mathsf{HASH3}'$, $\mathsf{HASH3}''$, $\mathsf{ENC3}$, $\mathsf{PRF3}$, and $\mathsf{PRF3}'$, instead of $\mathsf{HASH2}$, $\mathsf{HASH2}'$, $\mathsf{HASH2}''$, $\mathsf{ENC2}$, $\mathsf{PRF2}$, and $\mathsf{PRF2}'$. From Lemma 1, we obtain

$$\begin{cases}\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) + 0.5\sigma^2_{\mathrm{priv}}/2^n,\\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC2}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) + 0.5\sigma^2_{\mathrm{auth}}/2^n,\end{cases} \tag{5}$$

since otherwise we can construct an input-respecting adversary $\mathcal{B}$ that contradicts Lemma 1.

Indistinguishability of $(\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}'')$. Let $\mathsf{HASH4}$, $\mathsf{HASH4}'$, and $\mathsf{HASH4}''$ be three independent random functions, where $\mathsf{HASH4},\mathsf{HASH4}'' : \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \to \{0,1\}^n$, and $\mathsf{HASH4}' : \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \to \mathcal{T}_{\mathrm{CLOC}}$. We show that $(\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}'')$ is indistinguishable from random functions $(\mathsf{HASH4},\mathsf{HASH4}',\mathsf{HASH4}'')$. For an adversary $\mathcal{B}$, define $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}''}(\mathcal{B})$ as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}''}(\mathcal{B}) \stackrel{\mathrm{def}}{=} &\Pr\left[\mathcal{B}^{\mathsf{HASH3}(\cdot,\cdot),\mathsf{HASH3}'(\cdot,\cdot),\mathsf{HASH3}''(\cdot,\cdot)} \Rightarrow 1\right]\\ &- \Pr\left[\mathcal{B}^{\mathsf{HASH4}(\cdot,\cdot),\mathsf{HASH4}'(\cdot,\cdot),\mathsf{HASH4}''(\cdot,\cdot)} \Rightarrow 1\right],\end{aligned}$$

where the first probability is taken over $F_1,\ldots,F_{16},F_{18},\ldots,F_{23} \xleftarrow{\$} \mathrm{Rand}(n)$ and the randomness of $\mathcal{B}$, and the last is over the randomness of $\mathsf{HASH4}$, $\mathsf{HASH4}'$, $\mathsf{HASH4}''$, and $\mathcal{B}$. The adversary makes queries of the form $(j,N,A) \in \{1,2,3\} \times \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}}$, and receives $\mathsf{HASH3}(N,A)$ or $\mathsf{HASH4}(N,A)$ if $j = 1$, $\mathsf{HASH3}'(N,A)$ or $\mathsf{HASH4}'(N,A)$ if $j = 2$, and $\mathsf{HASH3}''(N,A)$ or $\mathsf{HASH4}''(N,A)$ if $j = 3$. If $\mathcal{B}$ makes q queries and the queries are $(j_1,N_1,A_1),\ldots,(j_q,N_q,A_q)$, then we define the total associated data length as $a_1+\cdots+a_q$, where $(A_i[1],\ldots,A_i[a_i]) \xleftarrow{n} A_i$. Without loss of generality, we assume that $\mathcal{B}$ does not repeat a query, but the same nonce can be repeated across different queries. We show the following lemma.

Lemma 2. *Let $\mathcal{B}$ be an adversary that makes at most q queries, where the total associated data length is at most σ_A. Then we have $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}''}(\mathcal{B}) \le \sigma_A^2/2^n$.*

A proof is in Appendix C.

Definition of CLOC4. We define yet another version of $\mathrm{CLOC}[\mathrm{Rand}(n), \ell_N, \tau]$ called $\mathrm{CLOC4}[\ell_N, \tau]$. The encryption algorithm CLOC4-$\mathcal{E}$ and the decryption algorithm CLOC4-$\mathcal{D}$ are the same as CLOC3-$\mathcal{E}$ and CLOC3-$\mathcal{D}$, respectively, except that we use random functions $\mathsf{HASH4}$, $\mathsf{HASH4}'$, and $\mathsf{HASH4}''$, instead of $\mathsf{HASH3}$, $\mathsf{HASH3}'$, and $\mathsf{HASH3}''$. Therefore, CLOC4-$\mathcal{E}$ and CLOC4-$\mathcal{D}$ take $\mathsf{HASH4}$, $\mathsf{HASH4}'$, $\mathsf{HASH4}''$, F_{17}, F_{24}, F_{25}, and F_{26} as a key, and we write their subroutines as $\mathsf{ENC4}$, $\mathsf{PRF4}$, and $\mathsf{PRF4}'$, instead of $\mathsf{ENC3}$, $\mathsf{PRF3}$, and $\mathsf{PRF3}'$. From Lemma 2, we obtain

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) + 4\sigma_A^2/2^n, \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC3}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) + 4(\sigma_A + \sigma_{A'})^2/2^n. \end{cases} \tag{6}$$

To see this, for privacy, suppose that $\mathcal{A}$ makes a query (N_i, A_i, M_i). If $|M_i| = 0$, then $\mathcal{B}$ makes a query $(2, N_i, A_i)$. If $|M_i| \ge 1$, then $\mathcal{B}$ makes a query $(1, N_i, A_i)$, and then $(3, N_i, A_i)$. For authenticity, $\mathcal{B}$ behaves as above for encryption queries. For a decryption query (N'_j, A'_j, C'_j, T'_j), if $|C'_j| = 0$, then $\mathcal{B}$ makes a query $(2, N'_j, A'_j)$. Otherwise $\mathcal{B}$ makes a query $(3, N'_j, A'_j)$. Therefore, the total associated data length of $\mathcal{B}$ is no more than the twice of that of $\mathcal{A}$.

Indistinguishability of $\mathsf{PRF4}$. Recall that $\mathsf{PRF4} : \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{C}_{\mathrm{CLOC}} \to \mathcal{T}_{\mathrm{CLOC}}$ takes $\mathsf{HASH4}'$, $\mathsf{HASH4}''$, F_{24}, F_{25}, and F_{26} as a key. Let $\mathsf{PRF5}$ be a random function from $\mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{C}_{\mathrm{CLOC}}$ to $\mathcal{T}_{\mathrm{CLOC}}$. We show that $\mathsf{PRF4}$ is indistinguishable from $\mathsf{PRF5}$. Let $\mathcal{B}$ be an adversary, and we define $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{PRF4}}(\mathcal{B})$ as

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{PRF4}}(\mathcal{B}) \overset{\mathrm{def}}{=} \Pr\left[\mathcal{B}^{\mathsf{PRF4}(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{B}^{\mathsf{PRF5}(\cdot,\cdot,\cdot)} \Rightarrow 1\right],$$

where the first probability is taken over the randomness of $\mathsf{HASH4}', \mathsf{HASH4}'', F_{24}, F_{25}$, F_{26}, and $\mathcal{B}$, and the last is over the randomness of $\mathsf{PRF5}$ and $\mathcal{B}$. Suppose that $\mathcal{B}$ makes q queries, and if the queries are $(N_1, A_1, C_1), \ldots, (N_q, A_q, C_q)$, then we define the total ciphertext length as $m_1 + \cdots + m_q$, where $(C_i[1], \ldots, C_i[m_i]) \overset{n}{\leftarrow} C_i$. The same nonce can be repeated across different queries, but without loss of generality, we assume that $\mathcal{B}$ does not repeat a query. We show the following lemma.

Lemma 3. *Let $\mathcal{B}$ be an adversary that makes at most q queries, where the total ciphertext length is at most σ_C. Then we have $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{PRF4}}(\mathcal{B}) \le 0.5q^2/2^n + \sigma_C^2/2^n$.*

A proof is in Appendix D.

Definition of CLOC5. We define our final version of $\mathrm{CLOC}[\mathrm{Rand}(n), \ell_N, \tau]$, which we write $\mathrm{CLOC5}[\ell_N, \tau]$. The encryption algorithm CLOC5-$\mathcal{E}$ and the decryption algorithm CLOC5-$\mathcal{D}$ are the same as CLOC4-$\mathcal{E}$ and CLOC4-$\mathcal{D}$, respectively, except that we use a random function $\mathsf{PRF5}$ instead of $\mathsf{PRF4}$. We write $\mathsf{HASH5}$ and $\mathsf{ENC5}$ for $\mathsf{HASH4}$ and $\mathsf{ENC4}$. Then CLOC5-$\mathcal{E}$ and CLOC5-$\mathcal{D}$ take $\mathsf{HASH5}$, $\mathsf{PRF5}$, and F_{17} as a key, where $\mathsf{HASH5} : \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \to \{0,1\}^n$, $\mathsf{PRF5} : \mathcal{N}_{\mathrm{CLOC}} \times \mathcal{A}_{\mathrm{CLOC}} \times \mathcal{C}_{\mathrm{CLOC}} \to \mathcal{T}_{\mathrm{CLOC}}$, and $F_{17} : \{0,1\}^n \to \{0,1\}^n$ are all random functions. For reference, we present the specification in Fig. 18. From Lemma 3, we obtain

$$\begin{cases} \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) + 0.5q^2/2^n + \sigma_M^2/2^n, \\ \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC4}[\ell_N,\tau]}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) + 0.5(q+q')^2/2^n + (\sigma_M + \sigma_{C'})^2/2^n. \end{cases} \tag{7}$$

Privacy and Authenticity of CLOC5. We have the following lemma on the privacy and the authenticity of CLOC5.

Lemma 4. *We have $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \le \sigma_M^2/2^n$ and $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \le q'/2^\tau$.*

A proof is in Appendix E.

Proof (of Theorem 1). We are now ready to show our proof of Theorem 1. From (2), (4), (5), (6), (7), and Lemma 4, we obtain

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) \le \frac{\sigma_{\mathrm{priv}}^2}{2^n} + \frac{4\sigma_A^2}{2^n} + \frac{0.5q^2}{2^n} + \frac{2\sigma_M^2}{2^n} \le \frac{5\sigma_{\mathrm{priv}}^2}{2^n},$$

since $\sigma_{\mathrm{priv}} = q + \sigma_A + 2\sigma_M$. □

```
Algorithm CLOC5-$\mathcal{E}_{\mathsf{HASH5},\mathsf{PRF5},F_{17}}(N,A,M)$

1. if $|M|=0$ then
2.     $C \leftarrow \varepsilon$
3. else                                          // $|M| \geq 1$
4.     $S_{\mathsf{E}}[1] \leftarrow \mathsf{HASH5}(N,A)$
5.     $C \leftarrow \mathsf{ENC5}_{F_{17}}(S_{\mathsf{E}}[1],M)$
6. $T \leftarrow \mathsf{PRF5}(N,A,C)$
7. return $(C,T)$
```

```
Algorithm $\mathsf{ENC5}_{F_{17}}(S_{\mathsf{E}}[1],M)$          // $|M| \geq 1$

1. $(M[1],\ldots,M[m]) \xleftarrow{n} M$
2. for $i \leftarrow 1$ to $m-1$ do              // only for $m \geq 2$
3.     $C[i] \leftarrow S_{\mathsf{E}}[i] \oplus M[i]$
4.     $S_{\mathsf{E}}[i+1] \leftarrow F_{17}(\mathsf{fix1}(C[i]))$
5. $C[m] \leftarrow \mathsf{msb}_{|M[m]|}(S_{\mathsf{E}}[m]) \oplus M[m]$
6. $C \leftarrow (C[1],\ldots,C[m])$
7. return $C$
```

```
Algorithm CLOC5-$\mathcal{D}_{\mathsf{HASH5},\mathsf{PRF5},F_{17}}(N,A,C,T)$

1. $T^* \leftarrow \mathsf{PRF5}(N,A,C)$
2. if $T \neq T^*$ then return $\perp$
3. return 1
```

Fig. 18. Pseudocode of the encryption and the decryption algorithms of CLOC5

Proof (of Theorem 2). We finally show our proof of Theorem 2. From (3), (4), (5), (6), (7), and Lemma 4, we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{CLOC}[\mathrm{Perm}(n),\ell_N,\tau]}(\mathcal{A}) &\leq \frac{\sigma^2_{\mathrm{auth}}}{2^n} + \frac{4(\sigma_A+\sigma_{A'})^2}{2^n} + \frac{0.5(q+q')^2}{2^n} + \frac{(\sigma_M+\sigma_{C'})^2}{2^n} + \frac{q'}{2^\tau}\\ &\leq \frac{5\sigma^2_{\mathrm{auth}}}{2^n} + \frac{q'}{2^\tau},\end{aligned}$$

since $\sigma_{\mathrm{auth}} = q + \sigma_A + 2\sigma_M + q' + \sigma_{A'} + \sigma_{C'}$. $\square$

B Proof of Lemma 1

Without loss of generality, assume that $\mathcal{B}$ makes exactly q queries, and let $(j_1,X_1),\ldots,(j_q,X_q)$ be the queries made by $\mathcal{B}$. Suppose that $\mathcal{B}$ interacts with $Q_1,\ldots,Q_{26}$ oracles. We say that a bad event occurs and write $\mathcal{B}^{Q_1(\cdot),\ldots,Q_{26}(\cdot)}$ sets bad, if there exist two distinct queries (j,X), $(j',X') \in \{(j_1,X_1),\ldots,(j_q,X_q)\}$ such that $I(j,X)=I(j',X')$, where $I(j,X)$ denotes the input value of R in $Q_j(X)$. See Fig. 19 for the concrete descriptions of $I(1,X),\ldots,I(26,X)$. In the figure, we let

$$\begin{cases}\mathcal{J}_1 = \{2,3,5,6,7,8,11,12,13,14,18,19,20,21\},\\ \mathcal{J}_2 = \{4,9,10,15,16,22,23\},\\ \mathcal{J}_3 = \{24,25,26\},\\ \mathcal{J}_4 = \{1,17\}.\end{cases}$$

If $j \in \mathcal{J}_1$, then we use the xor of K_1 in computing $Q_j(X)$. Similarly, we use K_2 when $j \in \mathcal{J}_2$, and K_3 when $j \in \mathcal{J}_3$. The input value of R when $j \in \mathcal{J}_4$ is directly determined by the input value of the function itself. The absence of the bad event implies that the responses that $\mathcal{B}$ receives from the oracles are uniform and independent random bit strings, since the output values of R are all independent. Therefore, we have

$$\mathbf{Adv}^{\mathrm{ind}}_{Q}(\mathcal{B}) \leq \Pr\left[\mathcal{B}^{Q_1(\cdot),\ldots,Q_{26}(\cdot)} \text{ sets } \mathsf{bad}\right]. \qquad (8)$$

We also see that, from the argument above, the adaptivity does not help in increasing the probability of the bad event. Therefore, we may fix all queries $(j_1,X_1),\ldots,(j_q,X_q)$ made by $\mathcal{B}$, and evaluate the right hand side of (8) based on the randomness of K_1, K_2, and K_3. Let $(j,X),(j',X') \in \{(j_1,X_1),\ldots,(j_q,X_q)\}$ be two distinct queries. If $j=j'$, then we have $X \neq X'$, and hence we never have $I(j,X)=I(j',X')$ from the invertibility of f_1, f_2, g_1, g_2, and h. In what follows, suppose that $1 \leq j < j' \leq 26$, and we evaluate $\Pr[I(j,X)=I(j',X')]$.

$j \in \mathcal{J}_1$	$j \in \mathcal{J}_2$	$j \in \mathcal{J}_3$	$j \in \mathcal{J}_4$
$I(2,X) = X \oplus K_1$	$I(4,X) = X \oplus K_2$	$I(24,X) = X \oplus K_3$	$I(1,X) = \mathsf{fix0}(X)$
$I(3,X) = \mathsf{h}(X \oplus K_1)$	$I(9,X) = \mathsf{g}_1(\mathsf{f}_1(X \oplus K_2))$	$I(25,X) = \mathsf{f}_1(X \oplus K_3)$	$I(17,X) = \mathsf{fix1}(X)$
$I(5,X) = \mathsf{g}_1(\mathsf{f}_1(X \oplus K_1))$	$I(10,X) = \mathsf{g}_1(\mathsf{f}_2(X \oplus K_2))$	$I(26,X) = \mathsf{f}_2(X \oplus K_3)$	
$I(6,X) = \mathsf{g}_1(\mathsf{f}_1(\mathsf{h}(X \oplus K_1)))$	$I(15,X) = \mathsf{f}_1(X \oplus K_2)$		
$I(7,X) = \mathsf{g}_1(\mathsf{f}_2(X \oplus K_1))$	$I(16,X) = \mathsf{f}_2(X \oplus K_2)$		
$I(8,X) = \mathsf{g}_1(\mathsf{f}_2(\mathsf{h}(X \oplus K_1)))$	$I(22,X) = \mathsf{g}_2(\mathsf{f}_1(X \oplus K_2))$		
$I(11,X) = \mathsf{f}_1(X \oplus K_1)$	$I(23,X) = \mathsf{g}_2(\mathsf{f}_2(X \oplus K_2))$		
$I(12,X) = \mathsf{f}_1(\mathsf{h}(X \oplus K_1))$			
$I(13,X) = \mathsf{f}_2(X \oplus K_1)$			
$I(14,X) = \mathsf{f}_2(\mathsf{h}(X \oplus K_1))$			
$I(18,X) = \mathsf{g}_2(\mathsf{f}_1(X \oplus K_1))$			
$I(19,X) = \mathsf{g}_2(\mathsf{f}_1(\mathsf{h}(X \oplus K_1)))$			
$I(20,X) = \mathsf{g}_2(\mathsf{f}_2(X \oplus K_1))$			
$I(21,X) = \mathsf{g}_2(\mathsf{f}_2(\mathsf{h}(X \oplus K_1)))$			

Fig. 19. Descriptions of $I(1,X), \ldots, I(26,X)$

j'	$j = 2$	$j = 3$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
3	$\mathsf{i} \oplus \mathsf{h}$					
5	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_1$	$\mathsf{h} \oplus \mathsf{g}_1\mathsf{f}_1$				
6	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_1\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_1$	$\mathsf{i} \oplus \mathsf{h}$			
7	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_2$	$\mathsf{h} \oplus \mathsf{g}_1\mathsf{f}_2$	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{f}_1\mathsf{h} \oplus \mathsf{f}_2$		
8	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_2\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_1\mathsf{f}_2$	$\mathsf{f}_1 \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{i} \oplus \mathsf{h}$	
11	$\mathsf{i} \oplus \mathsf{f}_1$	$\mathsf{h} \oplus \mathsf{f}_1$	$\mathsf{g}_1 \oplus \mathsf{i}$	$\mathsf{g}_1\mathsf{f}_1\mathsf{h} \oplus \mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_2\mathsf{h} \oplus \mathsf{f}_1$
12	$\mathsf{i} \oplus \mathsf{f}_1\mathsf{h}$	$\mathsf{i} \oplus \mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{f}_1\mathsf{h}$	$\mathsf{g}_1 \oplus \mathsf{i}$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{f}_1\mathsf{h}$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{f}_1$
13	$\mathsf{i} \oplus \mathsf{f}_2$	$\mathsf{h} \oplus \mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1\mathsf{h} \oplus \mathsf{f}_2$	$\mathsf{g}_1 \oplus \mathsf{i}$	$\mathsf{g}_1\mathsf{f}_2\mathsf{h} \oplus \mathsf{f}_2$
14	$\mathsf{i} \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{i} \oplus \mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{g}_1 \oplus \mathsf{i}$
18	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{g}_1 \oplus \mathsf{g}_2$	$\mathsf{g}_1\mathsf{f}_1\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_2\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_1$
19	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_1\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_1\mathsf{h}$	$\mathsf{g}_1 \oplus \mathsf{g}_2$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1\mathsf{h}$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1$
20	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{g}_1 \oplus \mathsf{g}_2$	$\mathsf{g}_1\mathsf{f}_2\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_2$
21	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_2\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2\mathsf{h}$	$\mathsf{g}_1\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{g}_1\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_2\mathsf{h}$	$\mathsf{g}_1 \oplus \mathsf{g}_2$

Fig. 20. Analysis of Case $j, j' \in \mathcal{J}_1$ ($j \in \{2, 3, 5, 6, 7, 8\}$)

Case $j, j' \in \mathcal{J}_1$. There are 14 elements in $\mathcal{J}_1$, and hence we have 91 combinations of (j, j') with $j < j'$. See Fig. 20 and Fig. 21 for the analysis. In the figures, we use the same notation as in Fig. 9. That is, if z is a function in Fig. 20 or Fig. 21 which is of the form $\mathsf{z} = \mathsf{z}' \oplus \mathsf{z}''$, then this stands for a function $\mathsf{z}'(K_1) \oplus \mathsf{z}''(K_1)$. When z is of the form $\mathsf{z} = \mathsf{z}'\mathsf{z}''$, then this stands for a function $\mathsf{z}'(\mathsf{z}''(K_1))$. Recall that we define i as $\mathsf{i}(K_1) = K_1$. As an example, consider the case $j = 3$ and $j' = 6$. In this case we are interested in the event $\mathsf{h}(X \oplus K_1) = \mathsf{g}_1(\mathsf{f}_1(\mathsf{h}(X' \oplus K_1)))$, where X and X' are n-bit constants. We see that the event is equivalent to $K_1 \oplus \mathsf{g}_1(\mathsf{f}_1(K_1)) = Y$ for some constant Y, and Fig. 20 and Fig. 21 show the left hand side of the event. From the discussions in Sect. 4, the probability of this event is at most $1/2^n$. One can verify that all these events, $I(j, X) = I(j', X')$, are covered in Fig. 9, and we thus have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$ in this case.

Case $j \in \mathcal{J}_1$ and $j' \in \mathcal{J}_2 \cup \mathcal{J}_3 \cup \mathcal{J}_4$. We have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$ from the randomness of K_1 and the invertibility of f_1, f_2, g_1, g_2, and h.

Case $j, j' \in \mathcal{J}_2$. We have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$ with exactly the same reasoning in the analysis of Case $j, j' \in \mathcal{J}_1$.

Case $j \in \mathcal{J}_2$ and $j' \in \mathcal{J}_3 \cup \mathcal{J}_4$. We have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$ from the randomness of K_2 and the invertibility of f_1, f_2, g_1, and g_2.

Case $j, j' \in \mathcal{J}_3$. We have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$ form the analysis of Case $j, j' \in \mathcal{J}_1$.

j'	$j=11$	$j=12$	$j=13$	$j=14$	$j=18$	$j=19$	$j=20$
12	$\mathsf{i} \oplus \mathsf{h}$						
13	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{f}_1\mathsf{h} \oplus \mathsf{f}_2$					
14	$\mathsf{f}_1 \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{i} \oplus \mathsf{h}$				
18	$\mathsf{i} \oplus \mathsf{g}_2$	$\mathsf{f}_1\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{f}_2\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_1$			
19	$\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_1\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_2$	$\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1\mathsf{h}$	$\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_1$	$\mathsf{i} \oplus \mathsf{h}$		
20	$\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{f}_1\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{i} \oplus \mathsf{g}_2$	$\mathsf{f}_2\mathsf{h} \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{f}_1\mathsf{h} \oplus \mathsf{f}_2$	
21	$\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2\mathsf{h}$	$\mathsf{f}_1 \oplus \mathsf{g}_2\mathsf{f}_2$	$\mathsf{f}_2 \oplus \mathsf{g}_2\mathsf{f}_2\mathsf{h}$	$\mathsf{i} \oplus \mathsf{g}_2$	$\mathsf{f}_1 \oplus \mathsf{f}_2\mathsf{h}$	$\mathsf{f}_1 \oplus \mathsf{f}_2$	$\mathsf{i} \oplus \mathsf{h}$

Fig. 21. Analysis of Case $j, j' \in \mathcal{J}_1$ ($j \in \{11, 12, 13, 14, 18, 19, 20\}$)

```
Algorithm HASH3*_{F_1,...,F_4}(N, A)

1.  $(A[1], \ldots, A[a]) \xleftarrow{n} A$
2.  $S_{\mathsf{H}}[1] \leftarrow F_1(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
3.  if $0 \le |A[1]| \le n$ then
4.      if $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 0$ then
5.          $Y \leftarrow S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(N)$
6.      else                          // $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$
7.          $Y \leftarrow S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N))$
8.  if $|A| \ge n + 1$ then
9.      if $\mathsf{msb}_1(A[1]) = 0$ then
10.         $S_{\mathsf{H}}[2] \leftarrow F_2(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$
11.     else                          // $\mathsf{msb}_1(A[1]) = 1$
12.         $S_{\mathsf{H}}[2] \leftarrow F_3(S_{\mathsf{H}}[1] \oplus \mathsf{h}^{-1}(\mathsf{ozp}(A[2])))$
13.     if $a \ge 3$ then
14.         for $i \leftarrow 3$ to $a - 1$ do  // only for $a \ge 4$
15.             $S_{\mathsf{H}}[i] \leftarrow F_4(S_{\mathsf{H}}[i-1] \oplus A[i])$
16.         $S_{\mathsf{H}}[a] \leftarrow F_4(S_{\mathsf{H}}[a-1] \oplus \mathsf{ozp}(A[a]))$
17.     $Y \leftarrow S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N)$
18. return $Y$
```

Fig. 22. Definition of $\mathsf{HASH3}^*$ used in the proof of Lemma 2

Case $j \in \mathcal{J}_3$ and $j' \in \mathcal{J}_4$. We have $\Pr[I(j, X) = I(j', X')] = 1/2^n$ from the randomness of K_3 and the invertibility of f_1 and f_2.

Case $j, j' \in \mathcal{J}_4$. This case corresponds to $(j, j') = (1, 17)$, and $\Pr[I(j, X) = I(j', X')] = 0$ holds.

Finally, we evaluate the probability of the bad event. For any two distinct queries (j, X), $(j', X') \in \{(j_1, X_1), \ldots, (j_q, X_q)\}$, we have $\Pr[I(j, X) = I(j', X')] \le 1/2^n$. Therefore, we obtain

$$\Pr\left[\mathcal{B}^{Q_1(\cdot), \ldots, Q_{26}(\cdot)} \text{ sets } \mathsf{bad}\right] \le \sum_{1 \le j < j' \le q} \frac{1}{2^n} \le \frac{0.5q^2}{2^n}$$

as claimed. □

C Proof of Lemma 2

We consider the following partition of $\mathcal{A}_{\text{CLOC}} = \{0,1\}^*$, where $(A[1], \ldots, A[a]) \xleftarrow{n} A$.

$$\begin{cases} \mathcal{A}_{\text{CLOC}}^{(1)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, 1 \le |A| \le n-1, \mathsf{msb}_1(\mathsf{ozp}(A[1])) = 0\} \\ \mathcal{A}_{\text{CLOC}}^{(2)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, 0 \le |A| \le n-1, \mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1\} \\ \mathcal{A}_{\text{CLOC}}^{(3)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, |A| = n, \mathsf{msb}_1(A[1]) = 0\} \\ \mathcal{A}_{\text{CLOC}}^{(4)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, |A| = n, \mathsf{msb}_1(A[1]) = 1\} \\ \mathcal{A}_{\text{CLOC}}^{(5)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, |A| \ge n+1, |A[a]| = n\} \\ \mathcal{A}_{\text{CLOC}}^{(6)} = \{A \mid A \in \mathcal{A}_{\text{CLOC}}, |A| \ge n+1, 1 \le |A[a]| \le n-1\} \end{cases}$$

Without loss of generality, assume that the adversary $\mathcal{B}$ makes exactly q queries. We write the q queries as $(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)$. Suppose that $\mathcal{B}$ interacts with $\mathsf{HASH3}$, $\mathsf{HASH3}'$, and $\mathsf{HASH3}''$, and we say that a bad event occurs and write $\mathcal{B}^{\mathsf{HASH3}(\cdot,\cdot),\mathsf{HASH3}'(\cdot,\cdot),\mathsf{HASH3}''(\cdot,\cdot)}$ sets bad, if there exist two distinct queries (j, N, A), $(j', N', A') \in \{(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)\}$ such that

- $j = j'$,
- $A, A' \in \mathcal{A}^{(\ell)}_{\mathrm{CLOC}}$ for some $1 \le \ell \le 6$, and
- $\mathsf{HASH3}^*(N, A) = \mathsf{HASH3}^*(N', A')$,

where $\mathsf{HASH3}^*$ is defined in Fig. 22. It takes N and A as input, and $\mathsf{HASH3}^*$ is designed to output the input value of the last invocation of the random function in $\mathsf{HASH3}$, $\mathsf{HASH3}'$, or $\mathsf{HASH3}''$. The absence of the bad event implies that the responses that $\mathcal{B}$ receives from the oracles are uniform and independent random bit strings. Therefore, we have

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{HASH3},\mathsf{HASH3}',\mathsf{HASH3}''}(\mathcal{B}) \le \Pr\left[\mathcal{B}^{\mathsf{HASH3}(\cdot,\cdot),\mathsf{HASH3}'(\cdot,\cdot),\mathsf{HASH3}''(\cdot,\cdot)} \text{ sets } \mathsf{bad}\right]. \tag{9}$$

We also see that, from the argument above, the adaptivity does not help in increasing the probability of the bad event. Therefore, we may fix all queries $(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)$ made by $\mathcal{B}$, and evaluate the right hand side of (9) based on the randomness of $F_1, \ldots, F_4$. Let $(j, N, A), (j', N', A') \in \{(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)\}$ be two distinct queries such that $j = j'$ and $A, A' \in \mathcal{A}^{(\ell)}_{\mathrm{CLOC}}$ for some $1 \le \ell \le 6$. We evaluate

$$\Pr[\mathsf{HASH3}^*(N, A) = \mathsf{HASH3}^*(N', A')] \tag{10}$$

in the following six cases depending on the value of $1 \le \ell \le 6$.

Case $A, A' \in \mathcal{A}^{(1)}_{\mathrm{CLOC}}$. If $A = A'$, then we have $\mathsf{ozp}(N) \ne \mathsf{ozp}(N')$ from $N \ne N'$. Therefore,

$$F_1(\mathsf{fix0}(\mathsf{ozp}(A[1]))) \oplus \mathsf{ozp}(N) \ne F_1(\mathsf{fix0}(\mathsf{ozp}(A'[1]))) \oplus \mathsf{ozp}(N')$$

holds, and thus (10) $= 0$. If $A \ne A'$, then we have

$$\Pr[F_1(\mathsf{fix0}(\mathsf{ozp}(A[1]))) \oplus \mathsf{ozp}(N) = F_1(\mathsf{fix0}(\mathsf{ozp}(A'[1]))) \oplus \mathsf{ozp}(N')] = \frac{1}{2^n}$$

from $\mathsf{fix0}(\mathsf{ozp}(A[1])) \ne \mathsf{fix0}(\mathsf{ozp}(A'[1]))$, implying that (10) $= 1/2^n$.

Case $A, A' \in \mathcal{A}^{(2)}_{\mathrm{CLOC}}$. If $A = A'$, then we have $\mathsf{h}^{-1}(\mathsf{ozp}(N)) \ne \mathsf{h}^{-1}(\mathsf{ozp}(N'))$ from $N \ne N'$. Therefore,

$$F_1(\mathsf{fix0}(\mathsf{ozp}(A[1]))) \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N)) \ne F_1(\mathsf{fix0}(\mathsf{ozp}(A'[1]))) \oplus \mathsf{h}^{-1}(\mathsf{ozp}(N')),$$

and we have (10) $= 0$. If $A \ne A'$, then we have (10) $= 1/2^n$ from $\mathsf{fix0}(\mathsf{ozp}(A[1])) \ne \mathsf{fix0}(\mathsf{ozp}(A'[1]))$.

Case $A, A' \in \mathcal{A}^{(3)}_{\mathrm{CLOC}}$. By following the analysis of Case $A, A' \in \mathcal{A}^{(1)}_{\mathrm{CLOC}}$, we have (10) $= 0$ if $A = A'$, and (10) $= 1/2^n$ if $A \ne A'$.

Case $A, A' \in \mathcal{A}^{(4)}_{\mathrm{CLOC}}$. Similarly, by following the analysis of Case $A, A' \in \mathcal{A}^{(2)}_{\mathrm{CLOC}}$, we have (10) $= 0$ if $A = A'$, and (10) $= 1/2^n$ if $A \ne A'$.

Case $A, A' \in \mathcal{A}^{(5)}_{\mathrm{CLOC}}$. This case is more involved to analyze, and we first introduce a lemma to analyze this case. Let $M, M' \in \{0,1\}^*$ be two distinct strings such that $|M| = mn$ and $|M'| = m'n$ for $m, m' \ge 1$. Let $(M[1], \ldots, M[m]) \xleftarrow{n} M$ and $(M'[1], \ldots, M'[m']) \xleftarrow{n} M'$ be the partition. Let $F \xleftarrow{\$} \mathrm{Rand}(n)$ be a random function. We define $\mathsf{CBC}_F(M)$ as $S[m]$, where $S[i] \leftarrow F(S[i-1] \oplus M[i])$ for $i = 1, \ldots, m$ and $S[0] = 0^n$. $\mathsf{CBC}_F(M')$ is defined analogously. Let $\mathsf{COLL}_F(M, M')$ be the event defined as $\mathsf{CBC}_F(M) = \mathsf{CBC}_F(M')$. We make use of the following lemma shown by Black and Rogaway [15].

Lemma 5 ([15]). $\Pr[\mathsf{COLL}_F(M, M')] \le mm'/2^n + \max\{m, m'\}/2^n$*, where the probability is taken over* $F \xleftarrow{\$} \mathrm{Rand}(n)$.

Recall that $(j, N, A), (j', N', A') \in \{(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)\}$ are the two distinct queries we are analyzing, and let $(A[1], \ldots, A[a]) \xleftarrow{n} A$ and $(A'[1], \ldots, A'[a']) \xleftarrow{n} A'$ be the partition, where $|A[a]| = |A'[a']| = n$. For N and $A = (A[1], \ldots, A[a])$, we consider $M = (M[1], \ldots, M[m])$ defined as

$$M \leftarrow \begin{cases} S_{\mathsf{H}}[2] \oplus \mathsf{ozp}(N) & \text{if } a = 2, \\ (S_{\mathsf{H}}[2] \oplus A[3], \mathsf{ozp}(N)) & \text{if } a = 3, \\ (S_{\mathsf{H}}[2] \oplus A[3], A[4], \ldots, A[a], \mathsf{ozp}(N)) & \text{if } a \geq 4, \end{cases}$$

where $S_{\mathsf{H}}[1] = F_1(\mathsf{fix0}(A[1]))$, $S_{\mathsf{H}}[2] = F_2(S_{\mathsf{H}}[1] \oplus A[2])$ if $\mathsf{msb}_1(A[1]) = 0$, and $S_{\mathsf{H}}[2] = F_3(S_{\mathsf{H}}[1] \oplus A[2])$ if $\mathsf{msb}_1(A[1]) = 1$. We note that $m = a - 1$ holds, and we define $M' = (M'[1], \ldots, M'[m'])$ from N' and $A' = (A'[1], \ldots, A'[a'])$ analogously. It is not hard to see that if $\mathsf{HASH3}^*(N, A) = \mathsf{HASH3}^*(N', A')$ holds, then we have $\mathsf{COLL}_F(M, M')$, which is $\mathsf{CBC}_F(M) = \mathsf{CBC}_F(M')$, by setting $F \leftarrow F_4$. However, the converse may not be true since we may have $\mathsf{COLL}_F(M, M')$ even if $\mathsf{HASH3}^*(N, A) \neq \mathsf{HASH3}^*(N', A')$. Now we evaluate (10) in two cases, Case $(A[1], A[2]) = (A'[1], A'[2])$, and Case $(A[1], A[2]) \neq (A'[1], A'[2])$.

Case $(A[1], A[2]) = (A'[1], A'[2])$. In this case, we arbitrarily fix F_1, F_2, and F_3. We have $M \neq M'$ since the last $n(a-1)$ bits of $(A[1], \ldots, A[a], \mathsf{ozp}(N))$ and the last $n(a'-1)$ bits of $(A'[1], \ldots, A'[a'], \mathsf{ozp}(N'))$ are distinct, and by using Lemma 5 with $F \leftarrow F_4$, we obtain

$$(10) \leq \Pr[\mathsf{COLL}_F(M, M')] \leq \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n}.$$

Case $(A[1], A[2]) \neq (A'[1], A'[2])$. We have

$$\begin{aligned}(10) &\leq \Pr[\mathsf{COLL}_F(M, M') \text{ and } M[1] = M'[1]] + \Pr[\mathsf{COLL}_F(M, M') \text{ and } M[1] \neq M'[1]] \\ &\leq \Pr[M[1] = M'[1]] + \Pr[\mathsf{COLL}_F(M, M') \mid M[1] \neq M'[1]],\end{aligned}$$

and we also have $\Pr[\mathsf{COLL}_F(M, M') \mid M[1] \neq M'[1]] \leq (a-1)(a'-1)/2^n + \max\{a-1, a'-1\}/2^n$ from Lemma 5. It remains to evaluate $\Pr[M[1] = M'[1]]$, and we evaluate the probability in three cases, Case $A[1] = A'[1]$ and $A[2] \neq A'[2]$, Case $A[1] \neq A'[1]$ and $\mathsf{msb}_1(A[1]) \neq \mathsf{msb}_1(A'[1])$, and Case $A[1] \neq A'[1]$ and $\mathsf{msb}_1(A[1]) = \mathsf{msb}_1(A'[1])$.

Case $A[1] = A'[1]$ *and* $A[2] \neq A'[2]$. We arbitrarily fix F_1, and from $S_{\mathsf{H}}[1] \oplus A[2] \neq S'_{\mathsf{H}}[2] \oplus A'[2]$, we have $\Pr[M[1] = M'[1]] = 1/2^n$.

Case $A[1] \neq A'[1]$ *and* $\mathsf{msb}_1(A[1]) \neq \mathsf{msb}_1(A'[1])$. We arbitrarily fix F_1, and since the random functions used to compute $M[1]$ and $M'[1]$ are independent, we have $\Pr[M[1] = M'[1]] = 1/2^n$.

Case $A[1] \neq A'[1]$ *and* $\mathsf{msb}_1(A[1]) = \mathsf{msb}_1(A'[1])$. We proceed as follows.

$$\begin{aligned}\Pr[M[1] = M'[1]] \leq\ &\Pr[M[1] = M'[1] \text{ and } S_{\mathsf{H}}[1] \oplus A[2] = S'_{\mathsf{H}}[1] \oplus A'[2]] &(11)\\ &+ \Pr[M[1] = M'[1] \text{ and } S_{\mathsf{H}}[1] \oplus A[2] \neq S'_{\mathsf{H}}[1] \oplus A'[2]] &(12)\end{aligned}$$

We see $(11) \leq \Pr[S_{\mathsf{H}}[1] \oplus A[2] = S'_{\mathsf{H}}[1] \oplus A'[2]] = 1/2^n$ from $\mathsf{fix0}(A[1]) \neq \mathsf{fix0}(A'[1])$. For (12), we have $(12) \leq \Pr[M[1] = M'[1] \mid S_{\mathsf{H}}[1] \oplus A[2] \neq S'_{\mathsf{H}}[1] \oplus A'[2]] = 1/2^n$, since the condition ensures that two independent output values of the random function are used to compute $M[1]$ and $M'[1]$. We have covered all cases, and in Case $A, A' \in \mathcal{A}^{(5)}_{\text{CLOC}}$, we have

$$(10) \leq \frac{2}{2^n} + \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n}.$$

Case $A, A' \in \mathcal{A}^{(6)}_{\text{CLOC}}$. Similarly to the case above, we define $M = (M[1], \ldots, M[m])$ from N and $A = (A[1], \ldots, A[a])$ as

$$M \leftarrow \begin{cases} S_{\mathsf{H}}[2] \oplus \mathsf{ozp}(N) & \text{if } a = 2, \\ (S_{\mathsf{H}}[2] \oplus \mathsf{ozp}(A[3]), \mathsf{ozp}(N)) & \text{if } a = 3, \\ (S_{\mathsf{H}}[2] \oplus A[3], A[4], \ldots, A[a-1], \mathsf{ozp}(A[a]), \mathsf{ozp}(N)) & \text{if } a \geq 4, \end{cases}$$

where $S_{\mathsf{H}}[1] = F_1(\mathsf{fix0}(A[1]))$, $S_{\mathsf{H}}[2] = F_2(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$ if $\mathsf{msb}_1(A[1]) = 0$, and $S_{\mathsf{H}}[2] = F_3(S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]))$ if $\mathsf{msb}_1(A[1]) = 1$. We define $M' = (M'[1], \ldots, M'[m'])$ from N' and $A' = (A'[1], \ldots, A'[a'])$ analogously, and we evaluate (10) in two cases, Case $(A[1], A[2]) = (A'[1], A'[2])$, and Case $(A[1], A[2]) \neq (A'[1], A'[2])$.

Case $(A[1], A[2]) = (A'[1], A'[2])$. We fix F_1, F_2, and F_3, and use Lemma 5 to obtain

$$(10) \leq \Pr[\mathsf{COLL}_F(M, M')] \leq \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n}.$$

Case $(A[1], A[2]) \neq (A'[1], A'[2])$. We further split the case into four cases, Case $A[1] = A'[1]$, $A[2] \neq A'[2]$, and $\mathsf{ozp}(A[2]) = \mathsf{ozp}(A'[2])$, Case $A[1] = A'[1]$, $A[2] \neq A'[2]$, and $\mathsf{ozp}(A[2]) \neq \mathsf{ozp}(A'[2])$, Case $A[1] \neq A'[1]$ and $\mathsf{msb}_1(A[1]) \neq \mathsf{msb}_1(A'[1])$, and Case $A[1] \neq A'[1]$ and $\mathsf{msb}_1(A[1]) = \mathsf{msb}_1(A'[1])$.

Case $A[1] = A'[1]$*,* $A[2] \neq A'[2]$*, and* $\mathsf{ozp}(A[2]) = \mathsf{ozp}(A'[2])$. In this case, we necessary have $a = 2$ and $a' \geq 3$, or $a \geq 3$ and $a' = 2$. We arbitrarily fix F_1, F_2, and F_3 to conclude

$$(10) \leq \Pr[\mathsf{COLL}_F(M, M')] \leq \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n}$$

from $M \neq M'$. The analyses of the remaining three cases are similar to the last three cases of Case $A, A' \in \mathcal{A}^{(5)}_{\mathrm{CLOC}}$. We evaluate $\Pr[M[1] = M'[1]]$, and use

$$(10) \leq \Pr[M[1] = M'[1]] + \Pr[\mathsf{COLL}_F(M, M') \mid M[1] \neq M'[1]]$$

and $\Pr[\mathsf{COLL}_F(M, M') \mid M[1] \neq M'[1]] \leq (a-1)(a'-1)/2^n + \max\{a-1, a'-1\}/2^n$ from Lemma 5.

Case $A[1] = A'[1]$*,* $A[2] \neq A'[2]$*, and* $\mathsf{ozp}(A[2]) \neq \mathsf{ozp}(A'[2])$. We arbitrarily fix F_1, and from $S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]) \neq S'_{\mathsf{H}}[2] \oplus \mathsf{ozp}(A'[2])$, we have $\Pr[M[1] = M'[1]] = 1/2^n$.

Case $A[1] \neq A'[1]$ *and* $\mathsf{msb}_1(A[1]) \neq \mathsf{msb}_1(A'[1])$. We arbitrarily fix F_1, and since the random functions used to compute $M[1]$ and $M'[1]$ are independent, we have $\Pr[M[1] = M'[1]] = 1/2^n$.

Case $A[1] \neq A'[1]$ *and* $\mathsf{msb}_1(A[1]) = \mathsf{msb}_1(A'[1])$. We have

$$\begin{aligned}\Pr[M[1] = M'[1]] \leq {} & \Pr[M[1] = M'[1] \text{ and } S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]) = S'_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A'[2])] && (13)\\ & + \Pr[M[1] = M'[1] \text{ and } S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]) \neq S'_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A'[2])], && (14)\end{aligned}$$

$(13) \leq \Pr[S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]) = S'_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A'[2])] = 1/2^n$ from $\mathsf{fix0}(A[1]) \neq \mathsf{fix0}(A'[1])$, and $(14) \leq \Pr[M[1] = M'[1] \mid S_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A[2]) \neq S'_{\mathsf{H}}[1] \oplus \mathsf{ozp}(A'[2])] = 1/2^n$ from the independence of the output values of the random function used to compute $M[1]$ and $M'[1]$. Therefore, in Case $A, A' \in \mathcal{A}^{(6)}_{\mathrm{CLOC}}$, we have

$$(10) \leq \frac{2}{2^n} + \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n}.$$

Finally, we evaluate the probability of the bad event. From the analyses above, for any two distinct queries $(j, N, A), (j', N', A') \in \{(j_1, N_1, A_1), \ldots, (j_q, N_q, A_q)\}$, we have $(10) \leq aa'/2^n + \max\{a, a'\}/2^n$ for all cases, since we have $1/2^n \leq aa'/2^n + \max\{a, a'\}/2^n$ when $a = a' = 1$, and

$$\frac{2}{2^n} + \frac{(a-1)(a'-1)}{2^n} + \frac{\max\{a-1, a'-1\}}{2^n} \leq \frac{aa'}{2^n} + \frac{\max\{a, a'\}}{2^n}$$

when $a, a' \geq 2$. By writing the partition of A_i as $(A_i[1], \ldots, A_i[a_i])$, we obtain

$$\Pr\left[\mathcal{B}^{\mathsf{HASH3}(\cdot,\cdot),\mathsf{HASH3}'(\cdot,\cdot),\mathsf{HASH3}''(\cdot,\cdot)} \text{ sets } \mathsf{bad}\right] \leq \sum_{1 \leq i < i' \leq q} \frac{a_i a_{i'}}{2^n} + \frac{\max\{a_i, a_{i'}\}}{2^n} \leq \frac{\sigma_A^2}{2^n},$$

where the last inequality follows from the proof of [15, Theorem 4]. $\square$

```
Algorithm $\mathsf{PRF4}_{\mathsf{HASH4}',\mathsf{HASH4}'',F_{24},F_{25},F_{26}}(N,A,C)$
 1. if $|C|=0$ then
 2.     $T \leftarrow \mathsf{HASH4}'(N,A)$
 3. else                                              // $|C| \ge 1$
 4.     $S_{\mathsf{P}}[0] \leftarrow \mathsf{HASH4}''(N,A)$
 5.     $T \leftarrow \mathsf{PRF4}'_{F_{24},F_{25},F_{26}}(S_{\mathsf{P}}[0],C)$
 6. return $T$
```

```
Algorithm $\mathsf{PRF4}'_{F_{24},F_{25},F_{26}}(S_{\mathsf{P}}[0],C)$        // $|C| \ge 1$
 1. $(C[1],\ldots,C[m]) \xleftarrow{n} C$
 2. for $i \leftarrow 1$ to $m-1$ do                 // only for $m \ge 2$
 3.     $S_{\mathsf{P}}[i] \leftarrow F_{24}(S_{\mathsf{P}}[i-1] \oplus C[i])$
 4. if $|C[m]|=n$ then
 5.     $S_{\mathsf{P}}[m] \leftarrow F_{25}(S_{\mathsf{P}}[m-1] \oplus C[m])$
 6. else                                      // $1 \le |C[m]| \le n-1$
 7.     $S_{\mathsf{P}}[m] \leftarrow F_{26}(S_{\mathsf{P}}[m-1] \oplus \mathsf{ozp}(C[m]))$
 8. $T \leftarrow \mathsf{msb}_{\tau}(S_{\mathsf{P}}[m])$
 9. return $T$
```

Fig. 23. Definition of $\mathsf{PRF4}$

D Proof of Lemma 3

For reference, we present the specification of $\mathsf{PRF4}$ in Fig. 23.

Without loss of generality, we assume that $\mathcal{B}$ makes exactly q queries, and we write the queries as $(N_1,A_1,C_1),\ldots,(N_q,A_q,C_q)$. Consider the case where $\mathcal{B}$ interacts with $\mathsf{PRF4}$, and we say that a bad event occurs and write $\mathcal{B}^{\mathsf{PRF4}(\cdot,\cdot,\cdot)}$ sets bad, if there exist two distinct queries $(N,A,C),(N',A',C') \in \{(N_1,A_1,C_1),\ldots,(N_q,A_q,C_q)\}$ such that

- $|C[m]|=|C'[m']|=n$ and $I[m]=I'[m']$, or
- $1 \le |C[m]| \le n-1$, $1 \le |C'[m']| \le n-1$, and $I[m]=I'[m']$,

where $(C[1],\ldots,C[m]) \xleftarrow{n} C$, $(C'[1],\ldots,C'[m']) \xleftarrow{n} C'$, $I[m]=S_{\mathsf{P}}[m-1]\oplus C[m]$ (or $I[m]=S_{\mathsf{P}}[m-1]\oplus \mathsf{ozp}(C[m])$) is the input value of F_{25} or F_{26} for the query (N,A,C), and $I'[m']=S'_{\mathsf{P}}[m'-1]\oplus C'[m']$ (or $I'[m']=S'_{\mathsf{P}}[m'-1]\oplus \mathsf{ozp}(C'[m'])$) is the input value for (N',A',C'). The absence of the bad event implies that the responses that $\mathcal{A}$ receives are random bit strings, and we thus have

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{PRF4}}(\mathcal{B}) \le \Pr\left[\mathcal{B}^{\mathsf{PRF4}(\cdot,\cdot,\cdot)} \text{ sets } \mathsf{bad}\right]. \tag{15}$$

Since the adaptivity does not help in increasing the probability of the bad event, we fix all queries $(N_1,A_1,C_1),\ldots,(N_q,A_q,C_q)$ and evaluate $\Pr\left[\mathcal{B}^{\mathsf{PRF4}(\cdot,\cdot,\cdot)} \text{ sets } \mathsf{bad}\right]$. To evaluate the probability, we first focus on two distinct queries $(N,A,C),(N',A',C') \in \{(N_1,A_1,C_1),\ldots,(N_q,A_q,C_q)\}$, and evaluate $\Pr[I[m]=I[m']]$ in two cases, Case $|C[m]|=|C'[m']|=n$, and Case $1 \le |C[m]| \le n-1$ and $1 \le |C'[m']| \le n-1$.

Case $|C[m]|=|C'[m']|=n$. For (N,A,C), we consider $M=(M[1],\ldots,M[m])$ defined as $M \leftarrow (S_{\mathsf{P}}[0]\oplus C[1],C[2],\ldots,C[m])$, where $S_{\mathsf{P}}[0]=\mathsf{HASH4}''(N,A)$, and we also define $M'=(M'[1],\ldots,M'[m'])$ from (N',A',C') analogously. If $(N,A)=(N',A')$, then we have $C \ne C'$. We arbitrarily fix $\mathsf{HASH4}''$, and we see that $M \ne M'$ holds. We use Lemma 5 to obtain $\Pr[I[m]=I'[m']] \le mm'/2^n + \max\{m,m'\}/2^n$. If $(N,A) \ne (N',A')$, then we have

$$\begin{aligned}\Pr[I[m]=I'[m']] &= \Pr[I[m]=I'[m'] \text{ and } S_{\mathsf{P}}[0]\oplus C[1]=S'_{\mathsf{P}}[0]\oplus C'[1]] \qquad (16)\\ &\quad + \Pr[I[m]=I'[m'] \text{ and } S_{\mathsf{P}}[0]\oplus C[1]\ne S'_{\mathsf{P}}[0]\oplus C'[1]]. \qquad (17)\end{aligned}$$

We see that (16) $\le \Pr[S_{\mathsf{P}}[0]\oplus C[1]=S'_{\mathsf{P}}[0]\oplus C'[1]]=1/2^n$ since $\mathsf{HASH4}''$ takes two distinct input values, and (17) $\le \Pr[I[m]=I'[m'] \mid S_{\mathsf{P}}[0]\oplus C[1]\ne S'_{\mathsf{P}}[0]\oplus C'[1]] \le mm'/2^n + \max\{m,m'\}/2^n$ from Lemma 5.

Case $1 \le |C[m]| \le n-1$ *and* $1 \le |C'[m']| \le n-1$. We define $M=(M[1],\ldots,M[m])$ from (N,A,C) as

$$M \leftarrow \begin{cases} S_{\mathsf{P}}[0]\oplus \mathsf{ozp}(C[1]) & \text{if } m=1,\\ (S_{\mathsf{P}}[0]\oplus C[1],C[2],\ldots,C[m-1],\mathsf{ozp}(C[m])) & \text{if } m\ge 2,\end{cases}$$

where $S_{\mathsf{P}}[0] = \mathsf{HASH4}''(N, A)$, and we also define $M' = (M'[1], \ldots, M'[m'])$ from (N', A', C'). If $(N, A) = (N', A')$, then we arbitrarily fix $\mathsf{HASH4}''$, and we see that $M \neq M'$ holds from $C \neq C'$. We use Lemma 5 to obtain $\Pr[I[m] = I'[m']] \leq mm'/2^n + \max\{m, m'\}/2^n$. If $(N, A) \neq (N', A')$, then we use

$$\begin{aligned}\Pr[I[m] = I'[m']] \leq \Pr[S_{\mathsf{P}}[0] \oplus \mathsf{ozp}(C[1]) = S'_{\mathsf{P}}[0] \oplus \mathsf{ozp}(C'[1])] \quad (18)\\ + \Pr[I[m] = I'[m'] \mid S_{\mathsf{P}}[0] \oplus \mathsf{ozp}(C[1]) \neq S'_{\mathsf{P}}[0] \oplus \mathsf{ozp}(C'[1])]\,. \quad (19)\end{aligned}$$

We have (18) $= 1/2^n$ from the randomness of $\mathsf{HASH4}''$, and (19) $\leq mm'/2^n + \max\{m, m'\}/2^n$ from Lemma 5.

We now evaluate the probability of the bad event. For any two distinct queries $(N, A, C), (N', A', C') \in \{(N_1, A_1, C_1), \ldots, (N_q, A_q, C_q)\}$, we have $\Pr[I[m] = I'[m']] \leq 1/2^n + mm'/2^n + \max\{m, m'\}/2^n$. By writing the partition of C_i as $(C_i[1], \ldots, C_i[m_i])$, we obtain

$$\Pr\left[\mathcal{B}^{\mathsf{PRF4}(\cdot,\cdot,\cdot)} \text{ sets } \mathsf{bad}\right] \leq \sum_{1 \leq i < i' \leq q} \frac{1}{2^n} + \frac{m_i m_{i'}}{2^n} + \frac{\max\{m_i, m_{i'}\}}{2^n} \leq \frac{0.5q^2}{2^n} + \frac{\sigma_C^2}{2^n}.$$

We note that the last inequality follows from [15, Theorem 4]. □

E Proof of Lemma 4

Privacy of CLOC5. We first analyze the privacy of CLOC5. Consider the case where $\mathcal{A}$ interacts with CLOC5-$\mathcal{E}$, and let (N_i, A_i, M_i) be the i-th query, and (C_i, T_i) be the response, where $(C_i[1], \ldots, C_i[m_i]) \xleftarrow{n} C_i$. Let $\mathcal{I}_i = \{\mathsf{fix1}(C_i[1]), \ldots, \mathsf{fix1}(C_i[m_i - 1])\}$, i.e., $\mathcal{I}_i$ is the set of input values of F_{17} for the i-th query. We say that a bad event occurs and write $\mathcal{A}^{\text{CLOC5-}\mathcal{E}(\cdot,\cdot,\cdot)}$ sets bad if, for some $1 \leq i \leq q$ and $1 \leq j \leq m_i - 1$, we have

$$\mathsf{fix1}(C_i[j]) \in \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{i-1} \cup \{\mathsf{fix1}(C_i[1]), \ldots, \mathsf{fix1}(C_i[j-1])\}. \quad (20)$$

If (20) holds, then we say that $\mathsf{fix1}(C_i[j])$ causes the bad event. That is, the bad event occurs if $\mathsf{fix1}(C_i[j])$ collides with a previously used input value of F_{17}. We see that the absence of the bad event implies that the responses that $\mathcal{A}$ receives are uniform random bit strings, and we thus have

$$\mathbf{Adv}^{\text{priv}}_{\text{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \leq \Pr\left[\mathcal{A}^{\text{CLOC5-}\mathcal{E}(\cdot,\cdot,\cdot)} \text{ sets } \mathsf{bad}\right].$$

Assuming that $C_1[1], \ldots, C_1[m_1 - 1], \ldots, C_{i-1}[1], \ldots, C_{i-1}[m_{i-1} - 1], C_i[1], \ldots, C_i[j-1]$ do not cause the bad event, we see that $\mathsf{fix1}(C_i[j])$ is a uniform random string of $(n-1)$ bits. In particular, $C_1[1], \ldots, C_q[1]$ are all random bits from the nonce-respecting assumption on $\mathcal{A}$, and other values are random bits from the randomness of F_{17}. Therefore, we obtain the upper bound on $\Pr\left[\mathcal{A}^{\text{CLOC5-}\mathcal{E}(\cdot,\cdot,\cdot)} \text{ sets } \mathsf{bad}\right]$ as

$$\sum_{1 \leq i \leq q} \sum_{1 \leq j \leq m_i - 1} \frac{m_1 - 1}{2^{n-1}} + \cdots + \frac{m_{i-1} - 1}{2^{n-1}} + \frac{j-1}{2^{n-1}} \leq \sum_{0 \leq \ell \leq \sigma_M - 1} \frac{\ell}{2^{n-1}} \leq \frac{\sigma_M^2}{2^n},$$

and therefore, we obtain $\mathbf{Adv}^{\text{priv}}_{\text{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \leq \sigma_M^2/2^n$.

Authenticity of CLOC5. Finally, we analyze the authenticity of CLOC5. Consider the j-th decryption query (N'_j, A'_j, C'_j, T'_j), and suppose that, prior to this decryption query, $\mathcal{A}$ made i encryption queries and obtained the responses. Let $(N_1, A_1, M_1, C_1, T_1), \ldots, (N_i, A_i, M_i, C_i, T_i)$ be the list of the queries and the responses. Now $(N'_j, A'_j, C'_j) \notin \{(N_1, A_1, C_1), \ldots, (N_i, A_i, C_i)\}$ holds, since otherwise $\mathcal{A}$ does not succeed. This implies that, each time $\mathcal{A}$ makes a decryption query, $\mathcal{A}$ has to guess the output value of $\mathsf{PRF5}$ for a new input value. Since $\mathcal{A}$ makes at most q' decryption queries, we have $\mathbf{Adv}^{\text{auth}}_{\text{CLOC5}[\ell_N,\tau]}(\mathcal{A}) \leq q'/2^\tau$. □