Parallelizable and Authenticated Online Ciphers*

Elena Andreeva[1,2], Andrey Bogdanov[3], Atul Luykx[1,2], Bart Mennink[1,2], Elmar Tischhauser[1,2], and Kan Yasuda[1,4]

[1] Department of Electrical Engineering, ESAT/COSIC, KU Leuven, Belgium.
[2] iMinds, Belgium.
[3] Department of Mathematics, Technical University of Denmark, Denmark.
[4] NTT Secure Platform Laboratories, Japan.

Abstract. Online ciphers encrypt an arbitrary number of plaintext blocks and output ciphertext blocks which only depend on the preceding plaintext blocks. All online ciphers proposed so far are essentially serial, which significantly limits their performance on parallel architectures such as modern general-purpose CPUs or dedicated hardware. We propose the first parallelizable online cipher, COPE. It performs two calls to the underlying block cipher per plaintext block and is fully parallelizable in both encryption and decryption. COPE is proven secure against chosen-plaintext attacks assuming the underlying block cipher is a strong PRP. We then extend COPE to create COPA, the first parallelizable, online authenticated cipher with nonce-misuse resistance. COPA only requires two extra block cipher calls to provide integrity. The privacy and integrity of the scheme is proven secure assuming the underlying block cipher is a strong PRP. Our implementation with Intel AES-NI on a Sandy Bridge CPU architecture shows that both COPE and COPA are about *5 times faster* than their closest competition: TC1, TC3, and McOE-G. This high factor of advantage emphasizes the paramount role of parallelizability on up-to-date computing platforms.

Keywords: Block cipher, tweakable cipher, online cipher, authenticated encryption, nonce-misuse resistance, parallelizability, AES

1 Introduction

Online ciphers. A cipher which takes input of arbitrary length is said to be an *online cipher* if it can output ciphertext blocks as it is receiving the plaintext blocks. Specifically, the ith ciphertext block should only depend on the key and the first i plaintext blocks. This desirable functionality known more generally as online data processing is characteristic for other cryptographic primitives such as standard encryption schemes like CTR, CBC, OFB, and CFB.

The first theoretical treatment of online ciphers was put forward by Bellare, Boldyreva, Knudsen, and Namprempre [3]. They introduce the online ciphers HCBC1 and HCBC2, both of which require the use of two keys, one for the underlying block cipher and the other for the almost-xor-universal hash family [32]. Subsequently, Nandi [29] proposed two more efficient online ciphers MHCBC and MCBC. MHCBC improves upon HCBC2 by using a smaller hashing key with a finite field multiplication as universal hash function, whereas MCBC does not even require a universal hash function, thus needing only one block cipher key and calling the block cipher twice per plaintext block. Rogaway and Zhang in [35] recast the formalism of Bellare et al. [3] in terms of tweakable block ciphers [22] and provide three generalizations of the previous online ciphers: TC1, TC2, and TC3.

* An extended abstract of this paper will appear at ASIACRYPT 2013.

Authenticated encryption from online ciphers. While online AE schemes are not a novelty[5], presently we are aware of only one family of online *and* misuse-resistant AE schemes, McOE [11]. McOE makes use of the online cipher TC3 [35] to build its general structure and adds two calls to the tweakable cipher to achieve authenticity. To process messages of arbitrary lengths, McOE applies a tag splitting method, similar to the ciphertext stealing method [9].

Bellare et al. [3] give a few generic transformations to turn an online cipher into a secure authenticated encryption scheme.

Problem statement. All existing online ciphers are highly sequential and none of them offer any possibility for parallelizing the computation between distinct block cipher calls. The only exception can be seen in TC1, which allows parallelization only in decryption but not in encryption. As a consequence, the McOE AE schemes are not parallelizable either, due to the fact that they are based on existing online ciphers.

At the same time, in the overwhelming majority of cases in practice, the underlying cipher is AES which is very well parallelizable on many platforms. Parallelization is a rather inherent feature of hardware implementations, both in ASIC and FPGA. Also in general-purpose software, parallelizable encryption algorithms have profited in terms of performance due to the bitslice approach for a long time already [26,5,18]. However, with the introduction of the hardware supported AES by Intel in general-purpose x86 CPUs as an instruction set AES-NI in Intel Westmere, Sandy Bridge, and Ivy Bridge — followed by the AMD adoption of AES-NI in AMD Bulldozer and Piledriver — the parallelizability of the AES modes of operation has become of truly paramount importance. With AES-NI, using a parallelizable mode of operation enables performance advantages of a factor 3 and higher — see, for instance, the case of the (serial) CBC encryption vs (parallel) CBC decryption [1].

Our contributions. We present the first parallelizable online cipher, COPE, and the first parallelizable online authenticated encryption scheme with nonce-misuse resistance, COPA.

COPE: Our novel design is illustrated in Fig. 1. To process a single plaintext block two block cipher calls are required. A secret mask (*tweak*) is applied to the plaintext block and used as input to the first block cipher call. Then the output of the second block cipher call is masked again to produce the ciphertext block.
By introducing dummy masks, each block cipher call can be viewed as an instance of the XEX construction [33], which uses the so-called "doubling" mask generation. Our basic design only deals with message lengths that are a multiple of the block length. In order to handle messages of arbitrary lengths we use the technique prescribed in the XLS domain extender by Ristenpart et al. in [31]. In contrast with previous designs, our scheme only uses a *single key* and a *single cryptographic primitive*, namely a block cipher.
COPE is proven IND-CPA up to the birthday bound of $n/2$-bit security, where n denotes the block size of the underlying block cipher.

COPA: We transform COPE to support authentication, while maintaining parallelizability. The modifications are limited to computing an XOR sum of the plaintext data and using two extra block cipher calls; these can be seen in Fig. 2. The scheme also

[5] Examples of online AE schemes are abundant, including CCFB [24], CHM [15], CIP [16], CWC [19], EAX [4], GCM [27], IACBC [17], IAPM [17], XCBC [12], RPC [7], TAE [23], and OCB1-3 [34,33,21].

supports associated data in a way similar to how PMAC1 [33] operates. The privacy and integrity of COPA are proven up to the birthday bound.

To illustrate the impact of the parallelizability of our online schemes, we implement them with AES-NI on an Intel Sandy Brigde processor. We systematically compare the performance we attain with the online ciphers TC1, TC3, and MCBC as well as the online AE scheme McOE-G when instantiated with the AES. When compared to these closest online competitors, which are all explicitly not parallelizable, our modes provide performance improvements between a factor of 4.5 and 5, being below 2 cycles per byte on a single core. We expect almost a linear speed-up when several cores are available.

Organization of the paper. The remainder of the paper is organized as follows. We recall the necessary background on block ciphers in Section 2. Section 3 provides the specification of our new parallel modes. Sections 4 and 5 deal with the security proofs. Section 6 gives AES-NI implementations of our modes along with a systematic comparison to the state-of-the-art schemes. We conclude in Section 7.

2 Preliminaries

In this section we give syntax definitions and security notions of block ciphers and tweakable ciphers. In particular, we review the XE and XEX constructions by Rogaway [33], which provides us with an efficient way of making tweakable ciphers from an ordinary block cipher. These constructions of tweakable ciphers form the basis for COPE and COPA.

2.1 Block Ciphers

A block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a function that takes as input a key $k \in \mathcal{K}$ and a plaintext $M \in \{0,1\}^n$, and produces a ciphertext $C = E(k, M)$. We sometimes write $E_k(\cdot) = E(k, \cdot)$. For a fixed key k, a block cipher is a permutation on n bits, and we denote the inverse permutation (decryption function) by E_k^{-1}.

Let $\mathrm{Perm}(n)$ be the set of all permutations on n bits. When writing $x \xleftarrow{\$} X$ for some finite set X we mean that x is sampled uniformly from X. We write $\Pr\left[\mathbf{A} \;\middle|\; \mathbf{B}\right]$ to denote the probability of event $\mathbf{A}$ given $\mathbf{B}$.

Definition 1. *Let E be a block cipher. The* $\mathrm{prp}\pm1$ *advantage of a distinguisher $\mathcal{D}$ is defined as*

$$\mathbf{Adv}_E^{\mathrm{prp}\pm1}(\mathcal{D}) = \left| \Pr_k\left[\mathcal{D}^{E_k, E_k^{-1}} = 1\right] - \Pr_\pi\left[\mathcal{D}^{\pi, \pi^{-1}} = 1\right] \right| .$$

Here, $\mathcal{D}$ is a distinguisher with oracle access to either (E_k, E_k^{-1}) or (π, π^{-1}). The probabilities are taken over $k \xleftarrow{\$} \mathcal{K}$, $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ and random coins of $\mathcal{D}$, if any. By $\mathbf{Adv}_E^{\mathrm{prp}\pm1}(t, q)$ we denote the maximum advantage taken over all distinguishers that run in time t and make q queries.

We shall also write $E_k^{\pm1}$ for (E_k, E_k^{-1}). Similarly, $\pi^{\pm1}$ means (π, π^{-1}), and so on.

2.2 Binary Fields

The set $\{0,1\}^n$ of bit strings can be considered as the finite field $\mathrm{GF}(2^n)$ consisting of 2^n elements. To do this, we represent an element of $\mathrm{GF}(2^n)$ as a polynomial over the field $\mathrm{GF}(2)$ of degree less than n. A string $a_{n-1}a_{n-2}\cdots a_1a_0 \in \{0,1\}^n$ corresponds to the polynomial $a_{n-1}\mathtt{x}^{n-1} + a_{n-2}\mathtt{x}^{n-2} + \cdots + a_1\mathtt{x} + a_0 \in \mathrm{GF}(2^n)$. The addition in the field is just the addition of polynomials over $\mathrm{GF}(2)$ (that is, bitwise XOR, denoted by $\oplus$). To define multiplication in the field, we fix an irreducible polynomial $f(\mathtt{x})$ of degree n over the field $\mathrm{GF}(2)$. Given two elements $a(\mathtt{x}), b(\mathtt{x}) \in \mathrm{GF}(2^n)$, their product is defined as $a(\mathtt{x})b(\mathtt{x}) \bmod f(\mathtt{x})$—polynomial multiplication over the field $\mathrm{GF}(2)$ reduced modulo $f(\mathtt{x})$. We simply write $a(\mathtt{x})b(\mathtt{x})$ and $a(\mathtt{x}) \cdot b(\mathtt{x})$ to mean the product in the field $\mathrm{GF}(2^n)$.

The set $\{0,1\}^n$ can be also regarded as a set of integers ranging from 0 through $2^n - 1$. A string $a_{n-1}a_{n-2}\cdots a_1a_0 \in \{0,1\}^n$ corresponds to the integer $a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots + a_1 2 + a_0 \in [0, 2^n - 1]$. We often write elements of $\mathrm{GF}(2^n)$ as integers, based on these conversions. So, for example, "2" means $\mathtt{x}$, "3" means $\mathtt{x}+1$, and "7" means $\mathtt{x}^2+\mathtt{x}+1$. When we write multiplications such as $2 \cdot 3$ and 7^2, we mean those in the field $\mathrm{GF}(2^n)$.

2.3 XE and XEX Constructions of Tweakable Ciphers

Given a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a secret mask $\Delta \in \{0,1\}^n$, the ciphers

$$E'_{k,\Delta}(x) \stackrel{\text{def}}{=} E_k(x \oplus \Delta) \text{ or } \boldsymbol{E}'_{k,\Delta}(x) \stackrel{\text{def}}{=} E_k(x \oplus \Delta) \oplus \Delta$$

behave like another block cipher independent of E_k (up to some bound). In the case of $E'_{k,\Delta}$, adversaries are allowed to make only forward queries, whereas $\boldsymbol{E}'_{k,\Delta}$ accepts both encryption and decryption queries. Now consider a set of secret masks $\{\Delta_i\}_{i\in\mathcal{T}}$, where Δ_i and Δ_j may not be necessarily independent. An index $i \in \mathcal{T}$ is called a tweak, which is not secret. We obtain a tweakable cipher $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ by defining $\widetilde{E}_{k,i} \stackrel{\text{def}}{=} E'_{k,\Delta_i}$, and similarly $\widetilde{\boldsymbol{E}}_{k,i}$. We consider $\widetilde{E}_{k,i}$ and $\widetilde{\boldsymbol{E}}_{k,j}$ together, where $i \in \mathcal{T}_0$, $j \in \mathcal{T}_1$ and $\mathcal{T}_0 \cap \mathcal{T}_1 = \varnothing$.

Definition 2. *Let $\widetilde{E}, \widetilde{\boldsymbol{E}}$ be tweakable ciphers. The* twk *advantage of a distinguisher $\mathcal{D}$ is defined as*

$$\mathbf{Adv}^{\mathrm{twk}}_{\widetilde{E},\widetilde{\boldsymbol{E}}}(\mathcal{D}) = \left|\Pr_k\left[\mathcal{D}^{\widetilde{E}_{k,i},\widetilde{\boldsymbol{E}}^{\pm 1}_{k,j}} = 1\right] - \Pr_{\pi_i,\boldsymbol{\pi}_j}\left[\mathcal{D}^{\pi_i,\boldsymbol{\pi}^{\pm 1}_j} = 1\right]\right|.$$

Here, $\mathcal{D}$ is a distinguisher with oracle access to a series of permutations. The tweaks run over $i \in \mathcal{T}_0$ and $j \in \mathcal{T}_1$ where $\mathcal{T}_0 \cap \mathcal{T}_1 = \varnothing$. By $\mathbf{Adv}^{\mathrm{twk}}_{\widetilde{E},\widetilde{\boldsymbol{E}}}(t,q)$ we denote the maximum advantage taken over all distinguishers that run in time t and make q queries in total.

The doubling method [33] enables us to produce many different values of the mask Δ from just one secret value $L \stackrel{\text{def}}{=} E_k(0)$. Namely, the masks are produced as $\Delta = 2^\alpha 3^\beta 7^\gamma L$ for varying indices of α, β and γ. To do this, we need to choose our irreducible polynomial $f(\mathtt{x})$ carefully. First, $f(\mathtt{x})$ needs to be primitive, meaning that 2 generates the whole multiplicative group. Second, we make sure that $\log_2 3$ and $\log_2 7$ are both "huge." Third, we check if $\log_2 3$ and $\log_2 7$ are "apart enough" (modulo $2^n - 1$). We impose these conditions to ensure that values $2^\alpha 3^\beta 7^\gamma$ do not collide or become equal to 1. For example, when $n = 128$, the irreducible polynomial $f(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$ satisfies these requirements, making values $2^\alpha 3^\beta 7^\gamma$ all distinct and not equal to 1 for $\alpha \in [-2^{108}, 2^{108}]$ and $\beta, \gamma \in [-2^7, 2^7]$ [33], except for $(\alpha, \beta, \gamma) = (0, 0, 0)$. So we obtain tweakable ciphers $\widetilde{E}_{k,\alpha\beta\gamma}$ and $\widetilde{\boldsymbol{E}}_{k,\alpha\beta\gamma}$.

Lemma 1 (XE and XEX [33]). *Let $\mathcal{T}_0, \mathcal{T}_1 = \{(\alpha, \beta, \gamma)\}$ be two sets of integer triples such that $2^\alpha 3^\beta 7^\gamma$ are all distinct and not equal to 1, in particular $\mathcal{T}_0 \cap \mathcal{T}_1 = \varnothing$. Then the permutations $\{\widetilde{E}_{k,\alpha\beta\gamma}\}_{\mathcal{T}_0} \cup \{\widetilde{\boldsymbol{E}}^{\pm 1}_{k,\alpha\beta\gamma}\}_{\mathcal{T}_1}$ are indistinguishable from independently random permutations $\{\pi_{\alpha\beta\gamma}\}_{\mathcal{T}_0} \cup \{\boldsymbol{\pi}^{\pm 1}_{\alpha\beta\gamma}\}_{\mathcal{T}_1}$. Specifically, for given t, q, there exists a $t' \approx t$ such that*

$$\mathbf{Adv}^{\mathrm{twk}}_{\widetilde{E},\widetilde{\boldsymbol{E}}}(t, q) \le \frac{9.5q^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm 1}_{E}(t', 2q).$$

3 COPE and COPA: Design and Specification

We define COPE and COPA. COPE is an online cipher secure against chosen plaintext attacks. COPE makes two calls to the underlying block cipher per message block. COPA is an authenticated online cipher that builds on COPE. The additional cost of producing a tag is kept minimal—a message checksum and two extra block cipher calls. COPA accepts associated data input.

In this section we assume that the message length is a positive multiple of n. The length of associated data can be fractional. In App. A we show how to handle fractional messages with COPE and COPA. At the end of this section we give the design rationale for our constructions, explaining our choice of operations.

3.1 COPE Definition

Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be an n-bit block cipher, and denote $L \stackrel{\mathrm{def}}{=} E_k(0)$. The encryption and decryption procedures of the COPE online cipher on a message $M[1]M[2]\cdots M[d]$ of d n-bit blocks and on a ciphertext $C[1]C[2]\cdots C[d]$ are then defined as:

COPE-Encrypt $\mathcal{E}[E]$:
$V[0] \leftarrow L, \Delta_0 \leftarrow 3L, \Delta_1 \leftarrow 2L$
for $i = 1, \ldots, d$ **do**
  $V[i] \leftarrow E_k\big(M[i] \oplus \Delta_0\big) \oplus V[i-1]$
  $C[i] \leftarrow E_k\big(V[i]\big) \oplus \Delta_1$
  $\Delta_0 \leftarrow 2\Delta_0, \Delta_1 \leftarrow 2\Delta_1$
end for

COPE-Decrypt $\mathcal{E}^{-1}[E]$:
$V[0] \leftarrow L, \Delta_0 \leftarrow 3L, \Delta_1 \leftarrow 2L$
for $i = 1, \ldots, d$ **do**
  $V[i] \leftarrow E_k^{-1}\big(C[i] \oplus \Delta_1\big)$
  $M[i] \leftarrow E_k^{-1}\big(V[i] \oplus V[i-1]\big) \oplus \Delta_0$
  $\Delta_0 \leftarrow 2\Delta_0, \Delta_1 \leftarrow 2\Delta_1$
end for

In words, each message block $M[i]$ is XOR-ed with the mask $2^{i-1}3L$ and processed by a call to E_k. The encrypted value is then chained by XOR with the previous $V[i-1]$, resp. with L for the first block. The ciphertexts are then formed by processing the intermediate values through a second call to E_k and XOR with the mask $2^i L$. The encryption operation is illustrated in Fig. 1.

3.2 COPA Definition

The core of the authenticated online cipher COPA is identical to COPE. The only differences are that first, an authentication tag T is generated after the COPE cipher invocation, and second, that associated data (if any) is processed before the cipher iteration to produce a value V that is XOR-ed into the first intermediate block chaining (see Fig. 1): $V[0] \leftarrow V \oplus L$. If there is no associated data, then we set $V \stackrel{\mathrm{def}}{=} 0$. The tag T is computed by keeping a XOR checksum of the message blocks $\Sigma \stackrel{\mathrm{def}}{=} M[1] \oplus \cdots \oplus M[d]$ and computing

$$T \leftarrow E_k\big(E_k(\Sigma \oplus 2^{d-1}3^2 L) \oplus S\big) \oplus 2^{d-1}7L,$$

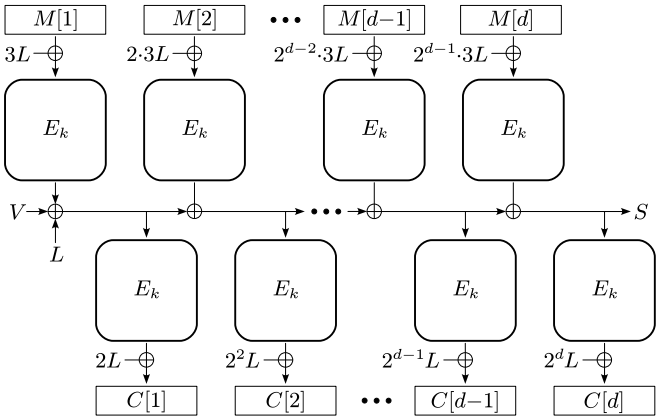
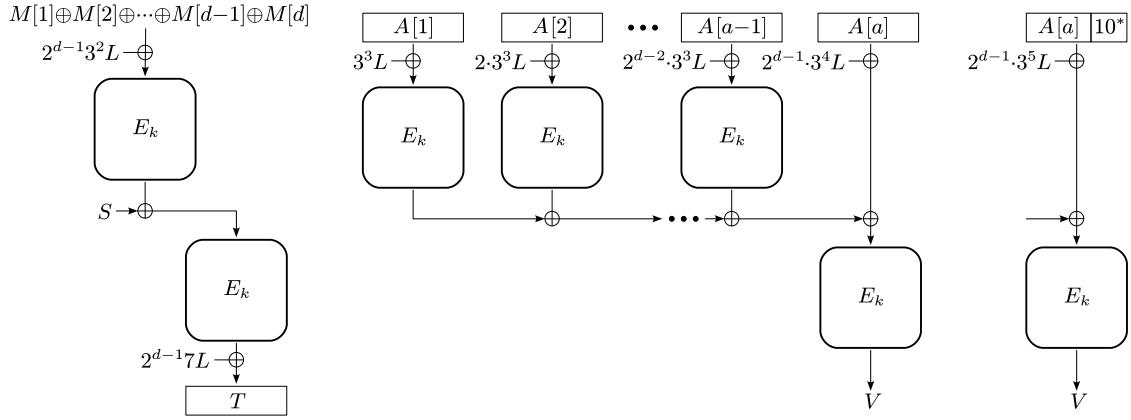


Fig. 1: Online cipher COPE. Set $V \stackrel{\text{def}}{=} 0$ for COPE. Variable S will be used later by COPA.



(a) Tag generation

(b) Processing of associated data

Fig. 2: Authenticated online cipher COPA: tag generation and processing of associated data.

with $S \stackrel{\text{def}}{=} V[d]$ denoting the last intermediate value in COPE's block chaining, as in Fig. 1. The tag computation is illustrated in Fig. 2a. The value V is generated as follows: any associated data $A[1], \ldots, A[a]$ is padded (if not a multiple of n bits) by a one and as many zeroes as necessary to obtain a multiple of the block size n. These blocks are then processed by a PMAC1-like [33] iteration as illustrated in Fig. 2b. Here, the block "$A[a]10^*$" replaces the block "$A[a]$" if $A[a]$ itself is not n bits. Tag verification occurs by checking if

$$S \oplus E_k(\Sigma \oplus 2^{d-1}3^2L) = E_k^{-1}(T \oplus 2^{d-1}7L),$$

where the tag is rejected if the equality is not true.

3.3 COPE and COPA for Arbitrary-Length Messages

We explain how to extend our schemes to accept "fractional" messages M in App. A. Here the length $|M|$ is not necessarily a positive multiple of the block size n. Note that simply using 10^* padding to M would result in ciphertext expansion. The methods described in App. A avoid such expansion with minimal loss of efficiency.

3.4 Design Rationale

We explain the choices we made while designing COPE and COPA. Namely, we explain why COPE and COPA are entirely based on block ciphers with doubling masks.

One could combine universal hashing with a block cipher to design an AE scheme. Indeed, McOE-G [11] follows this approach. However, we decided to avoid the use of universal hashing, for three reasons. First, the use of universal hashing would result in additional implementation cost, in particular with hardware. Second, recent study shows that there is an issue of weak keys with polynomial-based hashing [30]. Third, on the latest Intel CPUs, one call of AES is faster than one multiplication over the finite field $\mathrm{GF}(2^{128})$, which is an operation used for polynomial-based hashing.

There has been discussion of whether one should use the doubling method or Gray code to produce tweak masks. We decided to use doubling, for three reasons. First, doubling provides us with the framework of XE and XEX constructions, which makes our constructions and proofs simple and easy to analyze. Neither our constructions nor our proofs can be directly translated into a Gray code version, as it is not immediately clear which masks we should use for the construction to make the proof work. Second, although it was reported that Gray code performs better than doubling on Intel CPUs [21], recent study shows that the doubling method can be implemented equally efficiently [2]. Third, the speedup of Gray code mask generation requires precomputation and memory, whereas doubling does not.

4 Privacy Proof of COPE

This section is devoted to proving the security of COPE. We prove that COPE is secure against chosen-plaintext attacks with respect to privacy (IND-CPA).

4.1 Security Definition of Online Ciphers

We use the security definition of online ciphers from Rogaway and Zhang [35]. Let $(\{0,1\}^n)^+$ denote the set of strings whose length is a positive multiple of n bits and is at most $2^n \cdot n$ bits. An online cipher $\mathcal{E} : \mathcal{K} \times (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$ is a function such that it is a permutation on every block of n bits, having the additional feature that the outputs are the same for a common prefix. In other words, the first $|M|$ bits of $\mathcal{E}_k\big(M\|N\big)$ and $\mathcal{E}_k\big(M\|N'\big)$ are the same for any $M, N, N' \in (\{0,1\}^n)^+$. So an online cipher $\mathcal{E}_k$ yields a permutation of i-th blocks, where the permutation is determined by the prefix (i.e. the first $i-1$ blocks). Let $\mathrm{OPerm}(n)$ be the set of all such permutations $\pi : (\{0,1\}^n)^+ \to (\{0,1\}^n)^+$.

Definition 3. *Let $\mathcal{E}$ be an online cipher. The IND-CPA advantage of a distinguisher $\mathcal{D}$ is defined as*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(\mathcal{D}) = \left| \Pr_{k}\big[\mathcal{D}^{\mathcal{E}_k} = 1\big] - \Pr_{\pi}\big[\mathcal{D}^{\pi} = 1\big] \right| .$$

Here, $\mathcal{D}$ is a distinguisher with oracle access to either $\mathcal{E}_k$ or π. The probabilities are taken over $k \xleftarrow{\$} \mathcal{K}$, $\pi \xleftarrow{\$} \mathrm{OPerm}(n)$ and random coins of $\mathcal{D}$, if any. By $\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}}(t, q, \sigma, \ell)$ we denote the maximum advantage taken over all distinguishers that run in time t and make q queries, each of length at most ℓ blocks, and of total length at most σ blocks.

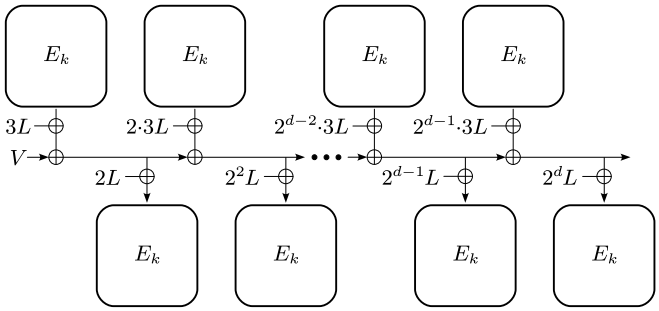


Fig. 3: IND-CPA proofs of COPE: introducing dummy masks rewriting the scheme in terms of XEX.

4.2 IND-CPA Proof of COPE

We now prove the IND-CPA security of COPE.

Theorem 1. *Let $\mathcal{E}[E]$ denote COPE, where E is the underlying block cipher. We have*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}[E]}(t,q,\sigma,\ell) \le \frac{38\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm1}_{E}(t',4\sigma) + \frac{(\ell+1)(q-1)^2}{2^n},$$

where $t' \approx t$.

The proof consists of two steps. First, we rewrite COPE in terms of XEX constructions.[6] Then by Lem. 1 we can replace our block cipher calls with random permutations. Second, we show that COPE (now calling random permutations) behaves exactly the same as the ideal functionality, as long as certain "bad" events do not occur. These events are collisions of state values, and the proof amounts to evaluating the probabilities of these events.

Rewriting with Tweakable Ciphers. We introduce dummy masks to the state values, as shown in Fig. 3. In this way we have rewritten COPE in terms of XEX construction. Namely, the block cipher calls in the upper layer are now $\widetilde{\boldsymbol{E}}_{k,\alpha-1,1,0}$ and those in the lower layer $\widetilde{\boldsymbol{E}}_{k,\alpha,0,0}$. Note that the "$L$" initially XORed to the state now disappears.

We use Lem. 1 to replace the block cipher calls in the upper layer with random permutations $\boldsymbol{\pi}_{\alpha-1,1,0}$ and those in the lower layer with $\boldsymbol{\pi}_{\alpha,0,0}$ (for $\alpha = 1, 2, \ldots$). Such a replacement costs us

$$\frac{9.5\cdot(2\sigma)^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm1}_{E}(t', 2\cdot 2\sigma) = \frac{38\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm1}_{E}(t', 4\sigma),$$

by Lem. 1. We write $\mathcal{E}[\boldsymbol{\pi}]$ to denote the COPE scheme making calls to independent random permutations $\boldsymbol{\pi}_{\alpha\beta\gamma}$ rather than to a block cipher.

Bounding Collision Probabilities. We bound the indistinguishability advantage by a collision probability. To do this, define variables $V[\alpha]$ of state values as $V[\alpha] \stackrel{\mathrm{def}}{=} \bigoplus_{i=1}^{\alpha} \boldsymbol{\pi}_{i-1,1,0}\big(M[i]\big)$ which is also equal to $\boldsymbol{\pi}^{-1}_{\alpha,0,0}\big(C[\alpha]\big)$.

We look for *collisions* of these variables. Here by a "collision" roughly we mean the same value of $V[\alpha]$ coming from different prefixes $M[1]M[2]\cdots M[\alpha]$ and $M'[1]M'[2]\cdots M'[\alpha]$,

[6] The reason why our IND-CPA COPE is based on XEX constructions, and not on XEs, is because our COPA, which gives decryption oracle access to adversaries, builds upon COPE.

for some α. More precisely, we have a collision of $V[\alpha] = V'[\alpha]$ if we have $V[\alpha-1] \neq V'[\alpha-1]$ and $V[\alpha] = V'[\alpha]$, which implies we must have $M[\alpha] \neq M'[\alpha]$ and also $M[i] \neq M'[i]$ for some $i < \alpha$. Let C denote the event that a collision of $V[\alpha]$ occurs for some α.

Claim. Unless C occurs, $\mathcal{E}[\boldsymbol{\pi}]$ is indistinguishable from the ideal functionality.

Proof. Under the condition $\neg\mathsf{C}$, every time we have a new prefix $M[1]M[2]\cdots M[\alpha]$, we get a new value $V[\alpha]$, which gives us a fresh random value (up to permutation) $C[\alpha]$. This is exactly the behavior of the ideal oracle. □

So it remains to evaluate $\Pr[\mathsf{C}]$. Note that a technicality is involved here: in the h-th query ($1 \le h \le q$), the α-th value $V[\alpha]$ ($1 \le \alpha \le \ell$) may be constructed out of α "old" permutation evaluations only, and we cannot simply bound the probability C as $\ell\binom{q}{2}$ times the probability that a certain $V[\alpha]$ hits an older value. Instead, a slightly more involved approach is needed. Let $\mathcal{D}$ denote the distinguisher, making q queries, each query being at most ℓ blocks. We shall construct an adversary $\mathcal{D}'$ which makes two queries in a non-adaptive way and tries to set C, as follows: $\mathcal{D}'$ first picks at random a pair (i,j) such that $1 \le i < j \le q$. Then $\mathcal{D}'$ runs $\mathcal{D}$, answering its queries with random (up to prefix and up to permutation) strings. When $\mathcal{D}$ makes the j-th query M_j, $\mathcal{D}'$ stops, and makes two queries, M_i (the i-th query) and M_j.

Claim. We have $\Pr[\mathcal{D}^{\mathcal{E}[\boldsymbol{\pi}]} \text{ sets } \mathsf{C}] \le (q-1)^2 \Pr[\mathcal{D}'^{\mathcal{E}[\boldsymbol{\pi}]} \text{ sets } \mathsf{C}]$.

Proof. We divide C into disjoint events C_h for $h = 2, 3, \ldots, q$, where C_h is the event that at the h-th query a collision occurs for the first time. Under the event C_h and $j = h$, $\mathcal{D}'$ perfectly simulates the game for $\mathcal{D}$. Moreover, the event C_h is independent of the values returned by the oracle so far. Therefore,

$$\Pr[\mathcal{D}' \text{ sets } \mathsf{C}] = \sum_{h=2}^{q}\sum_{i^*=1}^{q} \Pr[(i=i^*) \wedge (j=h) \wedge (\mathcal{D} \text{ sets } \mathsf{C}_h \text{ with } i^*)] \tag{1}$$

$$= \sum_{h=2}^{q}\sum_{i^*=1}^{q} \Pr[(i=i^*) \wedge (j=h)] \cdot \Pr[\mathcal{D} \text{ sets } \mathsf{C}_h \text{ with } i^*] \tag{2}$$

$$\ge \sum_{h=2}^{q} \frac{1}{(h-1)(q-1)} \cdot \Pr[\mathcal{D} \text{ sets } \mathsf{C}_h] \tag{3}$$

$$\ge \frac{1}{(q-1)^2} \cdot \sum_{h=2}^{q} \Pr[\mathcal{D} \text{ sets } \mathsf{C}_h] \tag{4}$$

$$= \frac{1}{(q-1)^2} \cdot \Pr[\mathcal{D} \text{ sets } \mathsf{C}] \tag{5}$$

□

Claim. We have $\Pr[\mathcal{D}'^{\mathcal{E}[\boldsymbol{\pi}]} \text{ sets } \mathsf{C}] \le (\ell+1)/2^n$.

Proof. Recall that $\mathcal{D}'$ is non-adaptive. Let us denote the two queries by $M = M[1]M[2]\cdots$ and $M' = M'[1]M'[2]\cdots$. We perform lazy sampling of the random permutations $\boldsymbol{\pi}_{\alpha-1,1,0}$. We first sample the points $M[1], M[2], \ldots$. We then sample the points $M'[\alpha]$ where $M'[\alpha] \neq M[\alpha]$. There are at most ℓ such points, as the length of a message is at most ℓ blocks. Each time we sample a point $M'[\alpha]$, there is a chance that C gets set. The points are sampled from the set of $2^n - 1$ values (the remaining after having sampled $M[1], M[2], \ldots$). Therefore, $\Pr[\mathcal{D}' \text{ sets } \mathsf{C}] \le \ell \cdot 1/(2^n - 1) \le (\ell+1)/2^n$. □

5 Privacy and Integrity Proofs of COPA

COPA takes as inputs (optional) associated data $A \in \{0,1\}^*$ and a message $M \in (\{0,1\}^n)^+$ to return a pair made up of a ciphertext $C \in (\{0,1\}^n)^+$ and a tag $T \in \{0,1\}^n$, as $(C,T) \leftarrow \mathcal{E}_k(A,M)$ (the fractional case requires only syntactical modification of the respective domain and range sizes). The decryption/verification algorithm $\mathcal{D}$ takes as input associated data $A \in (\{0,1\}^n)^+$, a ciphertext $C \in (\{0,1\}^n)^+$, and a tag $T \in \{0,1\}^n$ to output either a message $M \in (\{0,1\}^n)^+$ or the reject symbol $\perp$, as $M/\perp \leftarrow \mathcal{D}_k(A,C,T)$. Correctness of decryption/verification has to be satisfied, or for a valid (A,M) encrypted/authenticated to $(C,T) \leftarrow \mathcal{E}_k(A,M)$, under the same k the decryption/verification always outputs the correct (A,M), and not $\perp$.

5.1 Security Definition of Authenticated Online Ciphers

Also for authenticated online ciphers, we use the IND-CPA security advantage of Def. 3, except that the ideal encryption oracle now has an additional random function that maps $\{0,1\}^* \times (\{0,1\}^n)^+$ to $\{0,1\}^n$, corresponding to $(A,M) \mapsto T$.

We use the notion of integrity of authenticated encryption schemes from Fleischmann et al. [11]. By $\perp$, we denote a function that returns $\perp$ on every input.

Definition 4. *Let $\mathcal{E}$ be an online cipher. The integrity advantage of a distinguisher $\mathcal{D}$ is defined as*

$$\mathbf{Adv}^{\mathrm{int}}_{\mathcal{E}}(\mathcal{D}) = \left| \Pr_k\left[\mathcal{D}^{\mathcal{E}_k^{\pm 1}} = 1\right] - \Pr_k\left[\mathcal{D}^{\mathcal{E}_k,\perp} = 1\right] \right| .$$

Here, $\mathcal{D}$ is a distinguisher with oracle access to either $(\mathcal{E}_k, \mathcal{E}_k^{-1})$ or $(\mathcal{E}_k, \perp)$. To avoid a trivial win, we assume that the distinguisher does not make a query (A,C,T) if it has made a query (A,M) to the encryption oracle and obtained (C,T) from the oracle. By $\mathbf{Adv}^{\mathrm{int}}_{\mathcal{E}}(t,q,\sigma,\ell)$ we denote the maximum advantage taken over all distinguishers that run in time t and make q queries, each of length at most ℓ blocks, and of total length at most σ blocks.

5.2 Privacy of COPA

We now prove the IND-CPA security of COPA.

Theorem 2. *Let $\mathcal{E}[E]$ denote COPA, where E is the underlying block cipher. We have*

$$\mathbf{Adv}^{\mathrm{cpa}}_{\mathcal{E}[E]}(t,q,\sigma,\ell) \leq \frac{39(\sigma+q)^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm 1}_{E}(t', 4(\sigma+q)) + \frac{(\ell+2)(q-1)^2}{2^n},$$

where $t' \approx t$.

The IND-CPA security analysis of COPE carries over, with only minor modifications. First, we introduce dummy masks in a similar way (to the encryption part), and replace all XE (in the associated-data part) and XEX constructions by random permutations. This replacement costs us

$$\frac{9.5 \cdot (2\sigma+2q)^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm 1}_{E}(t', 2 \cdot 2(\sigma+q)) = \frac{38(\sigma+q)^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm 1}_{E}(t', 4(\sigma+q)).$$

The difference lies in the fact that per query two more block cipher evaluations are made for the derivation of the tag. Write $\mathcal{E}[\pi,\boldsymbol{\pi}]$ to denote the COPA scheme calling random permutations instead of a block cipher.

Next, we again use the collision event C, but introduce two more events. One is A, which is the event that we have a collision of V for two different associated data. Recall that for $A = \varnothing$, we have $V = 0$. The other is T, which is the event that we have a collision of tag values for messages of the same length (or more precisely, a collision of input values to a random permutation that produces tags).

Claim. Unless $\mathsf{A} \vee \mathsf{C} \vee \mathsf{T}$ occurs, $\mathcal{E}[\pi, \boldsymbol{\pi}]$ is indistinguishable from the ideal functionality.

Proof. Under the condition $\neg\mathsf{A} \wedge \neg\mathsf{C}$, every time we have a new prefix $AM[1]M[2] \cdots M[\alpha]$, we get a new value $V[\alpha]$, which gives us a fresh random value (up to permutation) $C[\alpha]$. This is exactly the behavior of the ideal oracle. Similarly, given $\neg\mathsf{T}$, also the tag values are always new, unless in the trivial case (A, M) has been queried before. □

Lemma 2 (PMAC1, [33]). *The function $H[\pi] : \{0,1\}^* \to \{0,1\}^n$ $(A \mapsto V)$ is indistinguishable from a random function $\Phi : \{0,1\}^* \to \{0,1\}^n$. Specifically, the distinguishing advantage (defined accordingly, only forward queries) is at most $\sigma^2/2^n$. Here, $\{0,1\}^*$ denotes the set of strings whose length is at most $2^n \cdot n$ bits.*

So now we replace XE and XEX constructions with random permutations and $H[\pi]$ with a random function Φ. Denote the scheme by $\mathcal{E}[\Phi, \boldsymbol{\pi}]$. Then we have the following.

Claim. We have $\Pr\big[\mathcal{D}^{\mathcal{E}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{A}\big] \le q^2/2^n$.

Proof. This is just a collision probability of a random function Φ plus the probability of Φ hitting 0, which is at most $0.5q(q-1)/2^n + q/2^n = 0.5q(q+1)/2^n \le q^2/2^n$. □

Claim. We have $\Pr\big[\mathcal{D}^{\mathcal{E}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{C} \vee \mathsf{T} \;\big|\; \neg\mathsf{A}\big] \le (\ell+2)(q-1)^2/2^n$.

Proof. The proof is exactly the same as in the proof of Thm. 1, except that the length of the message is now one block longer due to the tag generation. □

5.3 Integrity of COPA

The integrity proof of COPA is more involved than the privacy proofs. We prove the following theorem:

Theorem 3. *Let $\mathcal{E}[E]$ denote COPA, where E is the underlying block cipher. We have*

$$\mathbf{Adv}^{\mathrm{int}}_{\mathcal{E}[E]}(t, q, \sigma, \ell) \le \frac{39(\sigma+q)^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm1}_{E}(t', 4(\sigma+q)) + \frac{(\ell+2)(q-1)^2}{2^n} + \frac{2q}{2^n},$$

where $t' \approx t$.

Let F denote the event that the decryption oracle $\mathcal{E}_k^{-1}$ returns something other than $\perp$. Clearly the two games are the same as long as the event F does not occur, so we have

$$\Pr\big[\mathcal{D}^{\mathcal{E}_k^{\pm1}} = 1\big] - \Pr\big[\mathcal{D}^{\mathcal{E}_k,\perp} = 1\big] \le \Pr\big[\mathcal{D}^{\mathcal{E}_k^{\pm1}} \text{ sets } \mathsf{F}\big].$$

In the rest of this section we shall bound this probability. First, as usual, we replace block cipher calls with random permutations $\pi, \boldsymbol{\pi}$. Then we replace the PMAC1 part of processing associated data with a random function Φ. These all together cost us (cf. the proof of Thm. 2)

$$\frac{38(\sigma+q)^2}{2^n} + \frac{\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}\pm1}_{E}(t', 4(\sigma+q)).$$

Removing "Privacy Part." Define events A, C and T as we have done in the privacy proof of Thm. 2. Note that these events are defined in terms of variables $V[\alpha]$, where we also define $V[0] \stackrel{\text{def}}{=} V$ and $V'[\alpha+1]$ the input value to the block cipher that produces a tag. We define these values as being set only by the queries to the encryption oracle $\mathcal{E}$. We do not let queries to the decryption oracle $\mathcal{E}^{-1}$ affect variables $V[\cdot], V'[\cdot]$, whether or not it returns a message or $\perp$. Set $\mathsf{E} \stackrel{\text{def}}{=} \mathsf{A} \vee \mathsf{C} \vee \mathsf{T}$.

Next we define similar events A', C' and T'. These are exactly the same as the previous ones, except that now we consider only those events (i.e. collisions of $V[\cdot]$ or $V'[\cdot]$) that occur prior to a forgery (that is, under the condition $\neg\mathsf{F}$). Again, set $\mathsf{E}' \stackrel{\text{def}}{=} \mathsf{A}' \vee \mathsf{C}' \vee \mathsf{T}'$.

When we consider event F, we would like to assume that we are under the condition $\neg\mathsf{E}'$, meaning that the encryption oracle $\mathcal{E}$ has behaved ideally so far (till forgery). To do this, we use the inequality

$$\Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{F}\right] \le \Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right] + \Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{E}'\right].$$

We shall construct a distinguisher $\mathcal{D}'$ that breaks the privacy of the encryption oracle $\mathcal{E}$. The distinguisher $\mathcal{D}'$ uses $\mathcal{D}$, and the query complexity of $\mathcal{D}'$ is at most that of $\mathcal{D}$. Specifically, $\mathcal{D}'$ starts running $\mathcal{D}$, answering $\mathcal{E}$-queries using its $\mathcal{E}$ oracle, and whenever $\mathcal{D}$ makes a query to the decryption oracle $\mathcal{E}^{-1}$, $\mathcal{D}'$ replies with a $\perp$.

Claim. We have $\Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{E}'\right] \le q^2/2^n + (\ell+2)(q-1)^2/2^n$.

Proof. Note that if $\mathcal{D}^{\mathcal{E}^{\pm 1}}$ sets E', then till this event $\mathcal{D}'$ simulates the environment of $\mathcal{D}$ correctly. Hence we get $\Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}} \text{ sets } \mathsf{E}'\right] \le \Pr\left[\mathcal{D}'^{\mathcal{E}} \text{ sets } \mathsf{E}\right]$, which is less than $q^2/2^n + (\ell+2)(q-1)^2/2^n$ as shown in the privacy proof. □

Passing to a Single-Query Adversary. So it remains to evaluate the probability that $\mathcal{D}$ sets F under the condition $\neg\mathsf{E}'$. We shall construct a forger $\mathcal{D}_1$ from $\mathcal{D}$. The forger $\mathcal{D}_1$ makes multiple queries to the encryption oracle $\mathcal{E}$ but makes only one query to the decryption oracle $\mathcal{E}^{-1}$ at the end of its run. We define $\mathcal{D}_1$ as follows: it chooses a random index $i \in [1,q]$. It then runs $\mathcal{D}$, answering its $\mathcal{E}$-queries using the $\mathcal{E}$ oracle of $\mathcal{D}_1$ and answering the queries to the decryption oracle $\mathcal{E}^{-1}$ with $\perp$. When $\mathcal{D}$ makes the i-th query $(A^\star, C^\star, T^\star)$ to the decryption oracle, $\mathcal{D}_1$ outputs the query $(A^\star, C^\star, T^\star)$ and stops (or more precisely, makes that query to the decryption oracle $\mathcal{E}^{-1}$ and stops.)

Claim. We have $\Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right] \le q \Pr\left[\mathcal{D}_1^{\mathcal{E}^{\pm 1}[\Phi,\boldsymbol{\pi}]} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right]$.

Proof. Let F_h denote the event that at the h-th query the decryption oracle $\mathcal{E}^{-1}$ returns something other than $\perp$ for the first time; that is, the oracle has returned only $\perp$ so far. Clearly these are disjoint events, and we have $\mathsf{F} = \bigvee_{h=1}^{q} \mathsf{F}_h$. Then, under the events $\neg\mathsf{E}'$ and $i = h$, the forger $\mathcal{D}_1$ correctly simulates the game of $\mathcal{D}$. Therefore, we get $\Pr\left[\mathcal{D}_1^{\mathcal{E}^{\pm 1}} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right] \ge \Pr\left[(i=h) \wedge \mathcal{D}^{\mathcal{E}^{\pm 1}} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right] \ge (1/q) \Pr\left[\mathcal{D}^{\mathcal{E}^{\pm 1}} \text{ sets } \mathsf{F} \;\middle|\; \neg\mathsf{E}'\right]$. □

Evaluating Forgery Probabilities. Let $(A^\star, C^\star, T^\star)$ denote the (non-trivial) query made by $\mathcal{D}_1$ to the decryption oracle $\mathcal{E}^{-1}[\Phi, \boldsymbol{\pi}]$. We shall evaluate the probability that this would make $\mathcal{E}^{-1}$ return something other than $\perp$. To evaluate the probability, we shall consider the following cases.

– *$A^\star$ or $T^\star$ is new, or $C^\star$ contains a new block.*

- $A^\star$ *is new.* If $A^\star$ is new, then it means that it triggers the random function Φ and yields a fresh random value $V \leftarrow \Phi(A^\star)$. This value is XORed to the value that is input to the block cipher to produce the tag, which must be equal to $T^\star$. All other values XORed to the value are independent of V. Hence, regardless of the values $C^\star, T^\star$, the probability of such an event is at most $1/2^n$.
- $A^\star$ *is not new, but* $C^\star$ *contains a new block.* Let $C^\star[\alpha]$ be one of the new blocks. The decryption invokes $\boldsymbol{\pi}^{-1}_{\alpha,0,0}\big(C^\star[\alpha]\big)$, which is sampled from the set of at least $2^n - q$ points. Therefore, the probability of a forgery is at most $1/(2^n - q) \le 2/2^n$, assuming $q \le 2^{n-1}$.
- $A^\star$ *is not new,* $C^\star$ *does not contains a new block, but* $T^\star$ *is new.* This is similar to the previous case. This would trigger a fresh point of $\boldsymbol{\pi}^{-1}_{d^\star-1,0,1}(T^\star)$, where $d^\star$ denotes the number of blocks in the message $M^\star$. The point is sampled from the set of at least $2^n - q$ points. Therefore, the probability of a forgery is at most $1/(2^n - q) \le 2/2^n$.

To summarize this case, the probability is at most $\max\{1, 2, 2\}/2^n = 2/2^n$.

– $A^\star$ *and* $T^\star$ *are old, and* $C^\star$ *consists of old blocks.* To handle this case, we introduce some notation. For the query $(A^\star, C^\star, T^\star)$ in question, divide $C^\star$ into blocks as $C^\star[1]C^\star[2]\cdots C^\star[d^\star] \leftarrow C^\star$ and define $C^\star[0] \stackrel{\text{def}}{=} A^\star$ and $C^\star[d^\star+1] \stackrel{\text{def}}{=} T$. We then focus on a pair of adjacent "blocks" $\big(C^\star[\alpha-1], C^\star[\alpha]\big)$ for $\alpha = 1, 2, \ldots, d^\star + 1$. We call a pair *old* if it (as a pair) has already appeared in some previous query made to the encryption oracle $\mathcal{E}$ and in the corresponding value returned by the oracle. That is, if $\mathcal{D}$ has made a query (A', M') to the oracle and got (C', T') back, then we check if the pair in question $\big(C^\star[\alpha-1], C^\star[\alpha]\big)$ is contained in (A', C', T')—that is, we check if $\big(C^\star[\alpha-1], C^\star[\alpha]\big) = \big(C'[\alpha-1], C'[\alpha]\big)$ holds, where $C'[0]$ and $C'[d'+1]$ are defined similarly. We do this for all previous queries. We call the pair $\big(C^\star[\alpha-1], C^\star[\alpha]\big)$ *new* otherwise.

Claim. The query $(A^\star, C^\star, T^\star)$ always contains a new pair.

Proof. If $(A^\star, C^\star, T^\star)$ contains no new pairs, then, given the non-triviality of the query, we must have observed a collision, contradicting with the assumption $\neg\mathsf{E}'$. □

We now make a distinction among new pairs $\big(C^\star[\alpha-1], C^\star[\alpha]\big)$ based on the decrypted message block $M^\star[\alpha]$ from the two adjacent ciphertext blocks. We say that a pair is *collapsing* if there exists a previous query (A', M') made by $\mathcal{D}$ to the encryption oracle $\mathcal{E}$ such that $M'[\alpha] = M^\star[\alpha]$.

- *There exists a new pair* $\big(C^\star[\alpha-1], C^\star[\alpha]\big)$ *that is not collapsing.* This case means that we trigger a random sampling of $\boldsymbol{\pi}^{-1}_{\alpha,1,0}$ to compute $M^\star[\alpha]$. Then, note that the value $\Sigma^\star = M^\star[1] \oplus M^\star[2] \oplus \cdots \oplus M^\star[d^\star]$ is already uniquely determined by the values $C^\star[d^\star]$ and $T^\star$ (via Fig. 2a). There are at least $2^n - q$ possible values for $M^\star[\alpha]$, and the message blocks must sum up to this particular value $\Sigma^\star$, which happens with a probability at most $1/(2^n - q) \le 2/2^n$.
- *All new pairs in* $(A^\star, C^\star, T^\star)$ *are collapsing.* This final case is quite different from the previous ones above, as we do not have any fresh sampling of permutations $\boldsymbol{\pi}^{\pm 1}_{\alpha,\beta,\gamma}$ or the random function Φ in evaluating $\mathcal{E}^{-1}[\Phi, \boldsymbol{\pi}](A^\star, C^\star, T^\star)$. To tackle this case, we shall convert this forgery game into one where the adversary $\mathcal{D}^\circ$ tries to find multiple collisions by outputting the following set of values, without making any query to the oracles:
  1. $r \in [1, \ell]$,
  2. $1 \le \alpha_1 < \alpha_2 < \cdots < \alpha_r \le \ell + 1$,

3. (A_1, M_1), (A_2, M_2), ..., (A_r, M_r), and
4. (A'_1, M'_1), (A'_2, M'_2), ..., (A'_r, M'_r).

The adversary $\mathcal{D}^\circ$ "wins" if the submitted values form a multi-collision in the following sense: (A_i, M_i) and (A'_i, M'_i) collides at the α_i-th block, for all $i \in [1, r]$. The adversary $\mathcal{D}^\circ$ runs $\mathcal{D}_1$, simulating the $\mathcal{E}$ oracle with an ideal functionality. Note that this simulation is correct under the condition $\neg\mathsf{E}'$. When $\mathcal{D}_1$ outputs $(A^\star, C^\star, T^\star)$, $\mathcal{D}^\circ$ first checks for new pairs contained in it. Let $1 \le \alpha_1 < \alpha_2 < \cdots < \alpha_r \le \ell + 1$ be the positions of new pairs. Then $\mathcal{D}^\circ$ checks the history of values (C, T) that it returned. Note that under $\neg\mathsf{E}'$, a block $C^\star[\alpha]$ determines a unique prefix AM. We choose (A_i, M_i) to be the prefix determined by $C^\star[\alpha_i]$. To choose (A'_i, M'_i), let $A'_i M''$ be the prefix determined by $C^\star[\alpha_i - 1]$. Then $\mathcal{D}^\circ$ chooses randomly, from the previously queried values, a message block $M[\alpha] \neq M_i[\alpha]$. Set $M'_i \stackrel{\text{def}}{=} M''M[\alpha]$. The adversary $\mathcal{D}^\circ$ does this for $i = 1, 2, \ldots$ except for the last block.

* If $\alpha_r < d^\star + 1$, then we know the message checksum $\Sigma^\star = M^\star[1] \oplus \cdots \oplus M^\star[d^\star]$, so $\mathcal{D}^\circ$ does not have to guess the value of $M'_{\alpha_r}[\alpha_r]$.
* If $\alpha_r = d^\star + 1$, then we simply set the last input value to be the checksum of all previous (guessed) message blocks.

Now we observe that as long as all the guesses of the message blocks are correct, $\mathcal{D}^\circ$ wins if $\mathcal{D}_1$ succeeds in forgery of this type. It should be noted that the values returned by $\mathcal{E}$ are independent of the success probabilities in question, under the event $\neg\mathsf{E}'$. Therefore, for a fixed r,

$$\Pr\big[\mathcal{D}^\circ \text{ wins} \,\big|\, r\big] \ge \frac{1}{q-1} \cdot \cdots \cdot \frac{1}{q-1} \Pr\big[\mathcal{D}_1 \text{ forges} \,\big|\, r\big] = \frac{1}{(q-1)^{r-1}} \Pr\big[\mathcal{D}_1 \text{ forges} \,\big|\, r\big].$$

We then calculate $\Pr\big[\mathcal{D}^\circ \text{ wins} \,\big|\, r\big]$. We do this by lazy sampling of the permutations, and we see that, for a fixed r,

$$\Pr\big[\mathcal{D}^\circ \text{ wins} \,\big|\, r\big] \le \frac{1}{2^n - 1} \cdot \frac{1}{2^n - 1} \cdot \cdots \cdot \frac{1}{2^n - 1} = \frac{1}{(2^n - 1)^r}.$$

Hence by varying r we get in total

$$\Pr\big[\mathcal{D}_1 \text{ forges}\big] \le \sum_{i=1}^{\ell} \frac{(q-1)^{i-1}}{(2^n-1)^i} \Pr[i = r] \le \frac{1}{(2^n - 1)} \sum_{i=1}^{\ell} \Pr[i = r] = \frac{1}{(2^n - 1)} \le \frac{2}{2^n}.$$

Overall, the forgery probability of $\mathcal{D}_1$ is bounded by $\max\{2, 2, 2\}/2^n = 2/2^n$.

6 Efficient Parallel Implementation

6.1 The Setting

In this section, we discuss implementation characteristics of COPE and COPA. We present experimental measurements for our high-performance software implementations with AES-NI. We compare the performance of COPE and COPA to that of its closest competitors.

6.2 The setting

We compare our schemes to some prominent existing online ciphers: TC1 and TC3 [35] being the most efficient previous schemes; and MCBC [29] as a representative for a scheme relying only on block cipher invocations (as opposed to tweakable block ciphers or universal hashing). The modes HCBC1 and MHCBC are implicitly covered by TC1 and TC3, and HCBC2 has a performance inferior to TC3.

For the case of authenticated online ciphers, we exclude modes of operation and dedicated designs that are based on a nonce and rely on its non-reuse (e.g., GCM [27], OCB [21], ALE [6], and AEGIS [37]). Therefore, we compare our COPA design to the McOE family of authenticated encryption algorithms [11], which, to the best of our knowledge, is the only other online scheme not relying on the non-reuse of a nonce. We focus on the McOE-G instance, since McOE-X itself is not secure [28], featuring a key recovery with birthday complexity.

For the concrete instantiation of all schemes, we use the AES-128 [10] as the underlying block cipher, and multiplication in $\mathrm{GF}(2^{128})$ as an almost XOR-universal hash function [20]. As target platform for the implementations, we chose the recent generation of Intel microprocessors (Westmere or later) which support the AES-NI instruction set [13] and carryless multiplication [14].

6.3 Implementation characteristics of COPE and COPA

The online modes proposed in this paper can utilize parallelized execution of block cipher calls in two ways: for messages longer than one block, the encryptions of subsequent message blocks can be carried out independently of each other once the respective masks have been XORed. The same holds for the second series of block cipher calls, once the chaining XORs have been executed.

This parallelism can be exploited in a single-core scenario by pipelining the block cipher rounds for several consecutive block cipher invocations. Similarly, these invocations can be processed independently by multiple threads, with the recombination being the computation of the chaining. Note that both scenarios can be combined when multiple cores with pipelined block cipher calls are available, which is typically the case for Intel's AES-NI architecture.

On the recent Sandy and Ivy Bridge platforms, the AES round function can be computed at a latency of 8 cycles with a throughput of 1 cycle. Consequently, to fully utilize the pipeline, our implementation issues 8 AES round function evaluations on the next 8 consecutive blocks (independent data and same key). The tweak masks are computed using dedicated multiplication routines for 2^{α}, 3^{β} and $7^{\gamma} \in \mathrm{GF}(2^{128})$. By contrast, the general $\mathrm{GF}(2^{128})$ multiplication needed for TC1, TC3, and McOE-G is implemented using the `PCLMULQDQ` carryless multiplication instruction followed by modular reduction. The parallel implementation of the core part of our schemes' encryption routine is illustrated in App. C.

6.4 Performance measurements

We provide performance data for the (authenticated) encryption of messages of length $16 \cdot 2^b$ bytes, with $3 \leq b \leq 10$. The performance of AES-CTR is provided as a reference point. All measurements were taken on a single core of an Intel Core i5-2520M CPU at 2500 MHz, averaged over 500000 repetitions, processing one message at a time. Our

Table 1: Software performance of (authenticated) online ciphers based on the AES on the Intel Sandy Bridge platform (AES-NI). All numbers are given in cycles per byte (cpb).

	message length (bytes)						
Algorithm	128	256	512	1024	2048	4096	8192
CTR	1.74	1.27	1.05	0.93	0.86	0.83	0.82
TC1	9.00	8.75	8.65	8.60	8.56	8.56	8.56
TC3	9.08	8.82	8.72	8.67	8.63	8.63	8.62
MCBC	11.66	11.00	10.68	10.52	10.44	10.40	10.38
COPE	2.56	2.08	1.89	1.78	1.72	1.70	1.69
McOE-G	10.85	9.73	9.14	8.90	8.74	8.69	8.66
COPA	3.78	2.85	2.31	2.06	1.94	1.88	1.85

findings are summarized in Table 1 and illustrated in Fig. 4 in App. B. All numbers are given in cycles per byte (cpb).

One can observe that for all message lengths, the parallelizability of the proposed schemes results in speed-ups of factor $4.5 - 5$ in comparison to the existing modes, at least for somewhat longer messages. By fully utilizing the pipeline, our schemes are only marginally slower than two times AES-CTR, which implies that the overhead imposed by the computation of the masks and the chaining is kept at a minimum. The authenticated mode COPA carries the additional overhead of two more AES calls plus field arithmetic for finalization, but this quickly becomes insignificant as the message length increases. Note, however, that some constant overhead in comparison to the unauthenticated mode remains even for very long messages: this can be attributed to the fact that the computation of the checksum does not allow overwriting the message blocks, leading to increased register pressure. We also note that with the availability of carryless multiplication, TC1 and TC3 can be implemented more efficiently than the purely block cipher-based MCBC which was created with the goal to improve performance by avoiding field arithmetic.

The performance of our parallelizable schemes COPE and COPA can be further improved by utilizing multiple cores. Our implementation of multithreaded encryption confirms the intuition that one can expect a nearly linear speedup when using multiple cores for computing our schemes (i.e., the cost is < 1 cpb for two cores and so on).

7 Conclusion

By presenting COPE, our work provides the first solution for a parallelizable online cipher. Building on COPE, we go on to construct COPA, the first parallelizable and nonce-misuse resistant online authenticated encryption scheme. Our implementations of COPE and COPA with Intel AES-NI on a Sandy Bridge processor architecture benefit strongly from the parallelism, which gives us speed-ups of about factor 5 in comparison to existing (serial) online ciphers TC1, TC3, MCBC and the online AE scheme McOE-G.

Our designs additionally employ only a single key and use only a block cipher as a building block—as opposed to tweakable block ciphers or universal hash functions. We prove that our cipher COPE is an IND-CPA secure online permutation. The privacy result is also carried over to COPA. The integrity proof of COPA uses a technique of converting a forgery to a set of multiple collisions. It seems that the technique has not been used before by security proofs of parallelizable authenticated encryption mode or message authentication code. The technique may be applicable to other new types of parallelizable modes of operation. We leave it as an interesting open problem to construct a scheme with less primitive calls but with comparable security guarantees.

Acknowledgments. This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, in part by the Research Council KU Leuven: GOA TENSE, and in part by the Research Fund KU Leuven, OT/08/027. Elena Andreeva is supported by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO-Vlaanderen). Bart Mennink is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Akdemir, K., Dixon, M., Feghali, W., Fay, P., Gopal, V., Guilford, J., Erdinc Ozturk, G.W., Zohar, R.: Breakthrough AES Performance with Intel AES New Instructions. Intel white paper (January 2010)
2. Aoki, K., Iwata, T., Yasuda, K.: How Fast Can a Two-Pass Mode Go? A Parallel Deterministic Authenticated Encryption Mode for AES-NI (Extended Abstract of Work in Progress). Directions in Authenticated Ciphers (DIAC) (July 2012)
3. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online Ciphers and the Hash-CBC Construction. In: Kilian, J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2139, pp. 292–309. Springer (2001)
4. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy and Meier [36], pp. 389–407
5. Bernstein, D.J., Schwabe, P.: New AES Software Speed Records. In: Chowdhury et al. [8], pp. 322–336
6. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-Based Lightweight Authenticated Encryption. In: FSE'13. Lecture Notes in Computer Science, Springer (2013), to appear
7. Buonanno, E., Katz, J., Yung, M.: Incremental Unforgeable Encryption. In: Matsui [25], pp. 109–124
8. Chowdhury, D.R., Rijmen, V., Das, A. (eds.): Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings, Lecture Notes in Computer Science, vol. 5365. Springer (2008)
9. Daemen, J.: Hash Function and Cipher Design: Strategies Based on Linear and Differential Cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1995)
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
11. 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)
12. Gligor, V.D., Donescu, P.: Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In: Matsui [25], pp. 92–108
13. Gueron, S.: Intel Advanced Encryption Standard (AES) Instructions Set. Intel white paper (September 2012)
14. Gueron, S., Kounavis, M.: Intel Carry-Less Multiplication Instruction and its Usage for Computing the GCM mode. Intel white paper (September 2012)
15. Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: Robshaw, M.J.B. (ed.) FSE. Lecture Notes in Computer Science, vol. 4047, pp. 310–327. Springer (2006)
16. Iwata, T.: Authenticated Encryption Mode for Beyond the Birthday Bound Security. In: Vaudenay, S. (ed.) AFRICACRYPT. Lecture Notes in Computer Science, vol. 5023, pp. 125–142. Springer (2008)
17. Jutla, C.S.: Encryption Modes with Almost Free Message Integrity. J. Cryptology 21(4), 547–578 (2008)
18. Käsper, E., Schwabe, P.: Faster and Timing-Attack Resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES. Lecture Notes in Computer Science, vol. 5747, pp. 1–17. Springer (2009)
19. Kohno, T., Viega, J., Whiting, D.: CWC: A High-Performance Conventional Authenticated Encryption Mode. In: Roy and Meier [36], pp. 408–426
20. Krawczyk, H.: LFSR-based Hashing and Authentication. In: Desmedt, Y. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 839, pp. 129–139. Springer (1994)
21. 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)
22. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002)

23. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. J. Cryptology 24(3), 588–613 (2011)
24. 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)
25. Matsui, M. (ed.): Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2-4, 2001, Revised Papers, Lecture Notes in Computer Science, vol. 2355. Springer (2002)
26. Matsui, M., Nakajima, J.: On the Power of Bitslice Implementation on Intel Core2 Processor. In: Paillier, P., Verbauwhede, I. (eds.) CHES. Lecture Notes in Computer Science, vol. 4727, pp. 121–134. Springer (2007)
27. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer (2004)
28. Mendel, F., Mennink, B., Rijmen, V., Tischhauser, E.: A Simple Key-Recovery Attack on McOE-X. In: Pieprzyk, J., Sadeghi, A.R., Manulis, M. (eds.) Cryptology and Network Security. LNCS, vol. 7712, pp. 23 – 31. Springer (2012)
29. Nandi, M.: Two New Efficient CCA-Secure Online Ciphers: MHCBC and MCBC. In: Chowdhury et al. [8], pp. 350–362
30. Procter, G., Cid, C.: On Weak Keys and Forgery Attacks against Polynomial-based MAC Schemes. In: FSE 2013 (2013)
31. Ristenpart, T., Rogaway, P.: How to Enrich the Message Space of a Cipher. In: Biryukov, A. (ed.) FSE. Lecture Notes in Computer Science, vol. 4593, pp. 101–118. Springer (2007)
32. Rogaway, P.: Bucket Hashing and its Application to Fast Message Authentication. In: Coppersmith, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 963, pp. 29–42. Springer (1995)
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., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM Conference on Computer and Communications Security. pp. 196–205. ACM (2001)
35. Rogaway, P., Zhang, H.: Online Ciphers from Tweakable Blockciphers. In: CT-RSA 2011. Lecture Notes in Computer Science, vol. 6558, pp. 237–249. Springer, Heidelberg (2011)
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. Wu, H., Preneel, B.: AEGIS: A Fast Authenticated Encryption Algorithm. Directions in Authenticated Ciphers (July 2012)

A Handling Arbitrary-Length Messages

The problem of fractional messages with (authenticated) online ciphers has been treated by TC1-3 [35] and by McOE [11]. TC1-3 utilizes constructions of *variable-input-length* (VIL) tweakable ciphers. McOE does a trick they call "tag splitting." Our methods are similar to these and efficient but differ in detail.

A.1 COPE for Arbitrary-Length Messages

Our solution relies on the XLS construction [31] of VIL tweakable ciphers. XLS makes only three block-cipher calls and requires only simple bit operations outside block-cipher calls.

Let $\widetilde{\boldsymbol{E}} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable cipher and $E : \mathcal{K}' \times \{0,1\}^n \to \{0,1\}^n$ a block cipher. Then $\mathrm{XLS}[\widetilde{\boldsymbol{E}}, E]$ yields a VIL permutation on $\{0,1\}^{n+*}$, the set of string whose length is between n bits and $2n-1$ bits. Specifically, we get $\mathrm{XLS}[\widetilde{\boldsymbol{E}}, E] : \mathcal{K} \times \mathcal{K}' \times \mathcal{T} \times \{0,1\}^{n+*} \to \{0,1\}^{n+*}$. Using appropriate choice of (α, β, γ), we can realize the ciphers used in XLS by the underlying block cipher in COPE encryption scheme $\mathcal{E}$, dependent on the message length d. So we write $\mathrm{XLS}_{k,d}$ to denote the XLS invocation in COPE.

Let M be a message of at least n bits. Divide it into blocks as $M[1]M[2]\cdots M[d-1]M[d] \leftarrow M$, and assume that we have $1 \le \big|M[d]\big| \le n-1$. Then we can define $C \leftarrow \mathcal{E}_k(M)$ as

$$\begin{aligned} C[1]C[2]\cdots C[d-2], S &\leftarrow \mathcal{E}_k\big(M[1]M[2]\cdots M[d-2]\big) \text{ (let } \mathcal{E}_k \text{ output } S \text{ for now)} \\ C[d-1]C[d] &\leftarrow \mathrm{XLS}_{k,d}\big((M[d-1] \oplus S) \| M[d]\big) \\ C &\leftarrow C[1]C[2]\cdots C[d]. \end{aligned}$$

The IND-CPA proof of COPE carries over with minor modifications. Note that we have to "wait" the processing of $M[d-1]$ till receiving $M[d]$ (or "redo" after receiving), making the scheme less online. Yet, we make only three calls to the block cipher to process these two blocks.

We require $|M| \ge n$. As pointed out by [35], it seems a challenging problem to handle the case $|M| < n$ with encryption-only online ciphers in a secure manner.

A.2 COPA for Arbitrary-Length Messages

There are solutions of arbitrary-length messages for COPA also. This time we can take the advantage of the tag to handle even the case $|M| < n$. The solution for the case $|M| > n$ also becomes more efficient owing to the presence of tags.

Tag Splitting for $|M| < n$. We can do a trick similar to tag splitting [11] if $|M| < n$. We first choose appropriate parameters (α, β, γ) to make it independent of the ordinary COPA encryption algorithm $\mathcal{E}$. Write it $\mathcal{E}^*_k$ (which will be used only for fractional one-block messages). Given M such that $|M| = s < n$, we can define $(C, T) \leftarrow \mathcal{E}_k(M)$ as

$$\begin{aligned} (C', T') &\leftarrow \mathcal{E}^*_k\big(M10^*\big) \\ C &\leftarrow \lceil C' \rceil_s \text{ (leftmost } s \text{ bits)} \\ T &\leftarrow \lfloor C' \rfloor_{n-s} \lceil T' \rceil_s. \end{aligned}$$

One can directly verify the security of this extension. Note that the integrity relies on the 10^* padding as well as on the "partial" tag $\lceil T' \rceil_s$.

XLS for $|M| > n$. Our solution for this case is similar to that of COPE but is more efficient, in that COPA still remains fully online. Again, let M be a message whose length is more than n bits. Divide it into blocks as $M[1]M[2]\cdots M[d-1]M[d] \leftarrow M$, and assume that we have $1 \leq |M[d]| \leq n-1$. Then we can define $(C,T) \leftarrow \mathcal{E}_k(M)$ as

$$\begin{aligned}(C', T') &\leftarrow \mathcal{E}_k\big(M[1]M[2]\cdots M[d-1]\big)\\ C[d]T &\leftarrow \mathrm{XLS}_{k,d}\big(M[d]T'\big)\\ C &\leftarrow C'C[d],\end{aligned}$$

where $\mathrm{XLS}_{k,d}$ is defined similarly to the case of COPE. Given the security of COPA and XLS, it is straightforward to verify that this extension is also secure.

B Illustration of software performance

The software performance of the proposed schemes in comparison to other (authenticated) online ciphers is illustrated in Fig. 4.

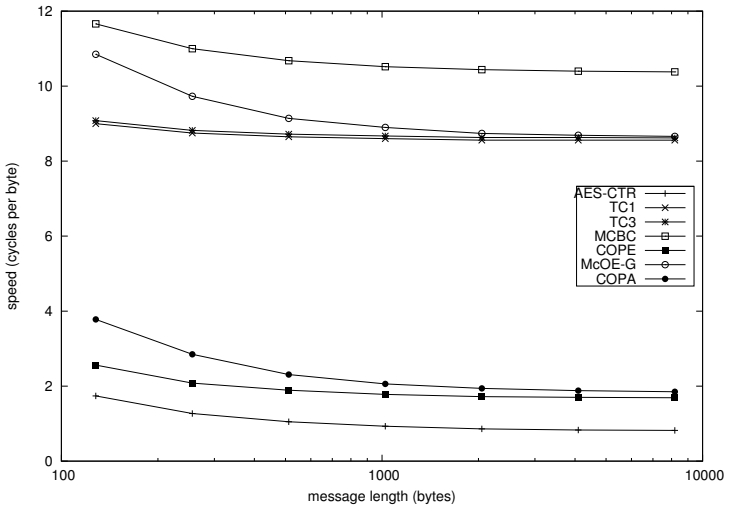


Fig. 4: Software performance of the proposed COPE and COPA and other (authenticated) online ciphers for different message lengths on Intel Sandy Bridge (AES-NI).

C Parallel implementation on Intel Sandy Bridge

The core of the encryption routine of both proposed schemes consists of processing message blocks with AES invocations and calculating the tweak masks and the chaining. This is done in an 8x parallel fashion in order to optimally utilize the pipeline. Note also that we use the AVX instruction set which in particular allows two source operands per instruction. See the following pseudocode (in two-column format):

```
# initialize Lup with 3*L
# initialize Ldown with 2*L
loop:
  # ... load next 8 blocks into xmm0, xmm7
  # prepare next 8 upper masks:
  vmovdqa     Lup, xmm8
  gf128_mul2  xmm8 xmm9
  gf128_mul2  xmm9 xmm10
  gf128_mul2  xmm10 xmm11
  gf128_mul2  xmm11 xmm12
  gf128_mul2  xmm12 xmm13
  gf128_mul2  xmm13 xmm14
  gf128_mul2  xmm14 xmm15
  # xor mask and first round key:
  vpxor       expkey, xmm8, xmm0
  vpxor       expkey, xmm9, xmm1
  vpxor       expkey, xmm10, xmm2
  vpxor       expkey, xmm11, xmm3
  vpxor       expkey, xmm12, xmm4
  vpxor       expkey, xmm13, xmm5
  vpxor       expkey, xmm14, xmm6
  vpxor       expkey, xmm15, xmm7
  # AES rounds:
  vpmovdqa    expkey+16, xmm10
  vaesenc     xmm10, xmm0, xmm0
  vaesenc     xmm10, xmm1, xmm1
  vaesenc     xmm10, xmm2, xmm2
  vaesenc     xmm10, xmm3, xmm3
  vaesenc     xmm10, xmm4, xmm4
  vaesenc     xmm10, xmm5, xmm5
  vaesenc     xmm10, xmm6, xmm6
  vaesenc     xmm10, xmm7, xmm7
  # ... repeat the above 8 times  ...
  # ... last AES round analogously ...
  # prepare next upper mask:
  gf128_mul2  xmm15 Lup
  # do the block chaining:
  vpxor       lastblock, xmm0, xmm0
  vpxor       xmm0, xmm1, xmm1
  vpxor       xmm1, xmm2, xmm2
  vpxor       xmm2, xmm3, xmm3
  vpxor       xmm3, xmm4, xmm4
  vpxor       xmm4, xmm5, xmm5
  vpxor       xmm5, xmm6, xmm6
  vpxor       xmm6, xmm7, xmm7
  vmovdqa     xmm7, lastblock
  # ... repeat the above
  #     for the second block cipher call ...
  # ... store xmm0..xmm7 to memory
  jmp         loop

subroutine gf128_mul2(in, out)
  vpslldq     $8, in, out
  vpsllq      $1, in, tmp
  vpsrlq      $63, out, out
  vpor        out, tmp, out
  vpextrq     $1, in, r15
  testq       r15, r15
  jns          .exit
  vpxor       REDPOLY, out, out
.exit:
  ret
```