Submission to the CAESAR competition

AES-COBRA v1

Designers/Submitters:
Elena Andreeva[1,2], Andrey Bogdanov[3], Martin M. Lauridsen[3],
Atul Luykx[1,2], Bart Mennink[1,2], Elmar Tischhauser[3], and Kan Yasuda[1,4]

Affiliation:
[1] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, Belgium.
[2] iMinds, Belgium.
[3] DTU Compute, Technical University of Denmark, Denmark.
[4] NTT Secure Platform Laboratories, Japan.

cobra@esat.kuleuven.be

March 16, 2014

1 Specification

1.1 Parameters

AES-COBRA has three parameters: the key length κ, the nonce length ν, and the tag length τ. The key length can be either 16 bytes (128 bits), 24 bytes (192 bits), or 32 bytes (256 bits). The tag length is between 8 bytes (64 bits) and 16 bytes (128 bits). The nonce, also called the public message number, is an optional input of length between 0 and 16 bytes (128 bits). AES-COBRA does not support a secret message number. Each key size of AES-COBRA corresponds to a key size of AES. AES-COBRA supports variable length associated data and plaintexts. To comply with our security claims from Sect. 2, the length of the associated data together with the plaintext data is at most $\approx 2^{64} \cdot 16$ bytes.

Recommended parameter set: 16 byte (128 bits) key length, 16 byte (128 bits) tag length, 16 byte (128 bits) nonce length.

1.2 Notation

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. Throughout the document E denotes the block cipher AES-128 and n denotes its block size (128 bits). Strings of length n are called blocks and strings of length $2n$ are called fragments.

By $\{0,1\}^*$ we denote the set of all strings, and by $\{0,1\}^+$ the set of all non-empty strings. Given two strings A and B, we use $A \parallel B$ and AB interchangeably to denote the concatenation of A and B. For $A \in \{0,1\}^*$, by $A10^*$ we denote the string with a 1 appended, and then padded with zeros until its length is a multiple of n. If X is a string with length a multiple of n, by $X[i]$ we denote the ith n-bit block of X. The length of a string X is denoted by $|X|$. By $\lfloor X \rfloor_j$ we denote the j most significant bits of X.

We can view the set $\{0,1\}^n$ of bit strings as the finite field $\mathrm{GF}(2^n)$ consisting of 2^n elements. To this end, we represent an element of $\mathrm{GF}(2^n)$ as a polynomial over the field $\mathrm{GF}(2)$ of degree less than n, and a string $a_{n-1}a_{n-2}\cdots a_1a_0 \in \{0,1\}^n$ corresponds to the polynomial $a_{n-1}\mathtt{x}^{n-1} + a_{n-2}\mathtt{x}^{n-2} + \cdots + a_1\mathtt{x} + a_0 \in \mathrm{GF}(2^n)$. Addition in the field is addition of polynomials over $\mathrm{GF}(2)$ (i.e. bitwise XOR, denoted by $\oplus$). To define multiplication in the field, we fix an irreducible polynomial $f(\mathtt{x}) := \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$ over the field $\mathrm{GF}(2)$. For $a(\mathtt{x}), b(\mathtt{x}) \in \mathrm{GF}(2^n)$, 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 $\mathrm{GF}(2^n)$, and denote the multiplication by $\otimes$.

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

1.3 Authenticated Encryption

The encryption $\mathcal{E}$ and decryption $\mathcal{D}$ functions of AES-COBRA have the following interfaces:

$$\mathcal{E} : \{0,1\}^\kappa \times \{0,1\}^\nu \times \{0,1\}^* \times \{0,1\}^+ \to \{0,1\}^+ \times \{0,1\}^\tau,$$
$$\mathcal{D} : \{0,1\}^\kappa \times \{0,1\}^\nu \times \{0,1\}^* \times \{0,1\}^+ \times \{0,1\}^\tau \to \{0,1\}^+ \cup \{\perp\}.$$

The function $\mathcal{E}$ takes as input a public message number $N \in \{0,1\}^\nu$, associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^+$. It returns a ciphertext $C \in \{0,1\}^+$, where $|C| = |M|$, and tag $T \in \{0,1\}^\tau$: $(C,T) \leftarrow \mathcal{E}(N,A,M)$. The decryption function $\mathcal{D}$ takes as input a public message number $N \in \{0,1\}^\nu$, associated data $A \in \{0,1\}^*$, ciphertext $C \in \{0,1\}^+$, and tag $T \in \{0,1\}^\tau$. The algorithm $\mathcal{D}$ outputs $M \in \{0,1\}^+$ if the tag is correct and $\perp$ otherwise, which we denote as $M/\perp \leftarrow \mathcal{D}(N,A,C,T)$. The encryption and decryption functions are described in Fig. 1 for messages whose length is a positive multiple of $2n$. For the case the associated data A is of length at most $4n$ and the message is of length $6n$, the function $\mathcal{E}$ is depicted in Figs. 2-3. How AES-COBRA deals with fractional messages is described in Sect. 1.4.

1.4 Fractional Data

We use ciphertext stealing [8] in order to deal with messages of arbitrary length. Let M be a message where $M[1]M[2]\cdots M[2\ell-1]M[2\ell] = M$ and $|M[i]| = n$ for $1 \le i < 2\ell - 1$.

1.4.1 Case 1: $\ell > 1$, $|M[2\ell-1]| = n$, and $0 < |M[2\ell]| < n$

We start by computing the ciphertext of $M[1]\cdots M[2\ell-2]$ as is usually done in COBRA, resulting in $C[1]\cdots C[2\ell-2]$. Let M^* denote the rightmost $n - |M[2\ell]|$ bits of $C[2\ell-2]$, and we write $C[2\ell-2] = C'[2\ell-2]M^*$. Then we compute the final ciphertext fragment $C[2\ell-1]C[2\ell]$ using $M[2\ell-1]M[2\ell]M^*$ as our "new" final message fragment (see Fig. 4) Note that the final block cipher calls use different tweaks as well: $7\cdot 2^\ell L'$ and $7\cdot(2^\ell L' \oplus L)$ instead of $2^\ell L'$ and $2^\ell L' \oplus L$. The resulting ciphertext is

$$C[1]\cdots C[2\ell-3]C'[2\ell-2]C[2\ell-1]C[2\ell]. \tag{1}$$

Fig. 5 shows a diagram of the process. The tag is computed as usual.

We can recover M^* with just knowledge of $C[2\ell-1]$ and $C[2\ell]$:

$$M[2\ell]M^* = \Big[C[2\ell] \oplus E_{k,\eta_2}(C[2\ell-1])\Big] \oplus \Big(\Big[E_{k,\eta_1}\big(C[2\ell] \oplus E_{k,\eta_2}(C[2\ell-1])\big) \oplus C[2\ell-1]\Big] \otimes L\Big),$$

where $E_{k,\eta_1}(x) := E_k(x \oplus 7\cdot 2^\ell L')$ and $E_{k,\eta_2}(x) := E_k(x \oplus 7\cdot 2^\ell(L' \oplus L))$.

COBRA-ENCRYPT $\mathcal{E}(N, A, M)$:

$L \leftarrow E_k(1), \Sigma \leftarrow 0$
$N \leftarrow N10^*$
$\eta \leftarrow 4L, V \leftarrow L^2 \oplus (N \otimes L)$
for $i = 1, \ldots, d$ **do**
  $V \leftarrow V \oplus M[2i-1]$
  $C[2i-1] \leftarrow V$
  $V \leftarrow (V \otimes L) \oplus M[2i]$
  $C[2i] \leftarrow V$
  $\rho \leftarrow E_k(\eta \oplus C[2i])$
  $\Sigma \leftarrow \Sigma \oplus \rho$
  $C[2i-1] \leftarrow \rho \oplus C[2i-1]$
  $\sigma \leftarrow E_k(\eta \oplus L \oplus C[2i-1])$
  $\Sigma \leftarrow \Sigma \oplus \sigma$
  $C[2i] \leftarrow \sigma \oplus C[2i]$
  if $i < d$ **then**
    $V \leftarrow V \otimes L$
    $\eta \leftarrow 2\eta$
  end if
end for

$U \leftarrow$PROCESSAD(A)
$T \leftarrow$COMPUTETAG(L, η, Σ, N, U)
return $(C, \lfloor T \rfloor_\tau)$

COBRA-DECRYPT $\mathcal{D}(N, A, C, T)$:

$L \leftarrow E_k(1), \Sigma \leftarrow 0$
$N \leftarrow N10^*$
$\eta \leftarrow 4L, V \leftarrow L^2 \oplus (N \otimes L)$
for $i = 1, \ldots, d$ **do**
  $\sigma \leftarrow E_k(\eta \oplus L \oplus C[2i-1])$
  $\Sigma \leftarrow \Sigma \oplus \sigma$
  $M[2i] \leftarrow \sigma \oplus C[2i]$
  $\rho \leftarrow E_k(\eta \oplus M[2i])$
  $\Sigma \leftarrow \Sigma \oplus \rho$
  $V' \leftarrow \rho \oplus C[2i-1]$
  $M[2i-1] \leftarrow V' \oplus V$
  $V' \leftarrow V' \otimes L$
  $V \leftarrow M[2i]$
  $M[2i] \leftarrow V' \oplus M[2i]$
  if $i < d$ **then**
    $V \leftarrow V \otimes L$
    $\eta \leftarrow 2\eta$
  end if
end for

$U \leftarrow$PROCESSAD(A)
$T' \leftarrow$COMPUTETAG(L, η, Σ, N, U)
return $T = \lfloor T' \rfloor_\tau \ ? \ M : \perp$

PROCESSAD(A):

$X \leftarrow A \parallel 10^*$
$J \leftarrow E_k(0)$
$U \leftarrow J$
for $i = 1, \ldots, |X|/n - 1$ **do**
  $U \leftarrow (U \oplus X[i]) \otimes J$
end for
$U \leftarrow E_k(2J \oplus U \oplus X[c])$
return U

COMPUTETAG(L, η, Σ, N, U):

$\eta \leftarrow 3(\eta \oplus L)$
$T \leftarrow E_k(\eta \oplus \Sigma)$
$\eta \leftarrow 3\eta$
$T \leftarrow E_k(\eta \oplus T \oplus N \oplus U)$
return T

Figure 1: COBRA for integral data (message length is a multiple of $2n$ bits).

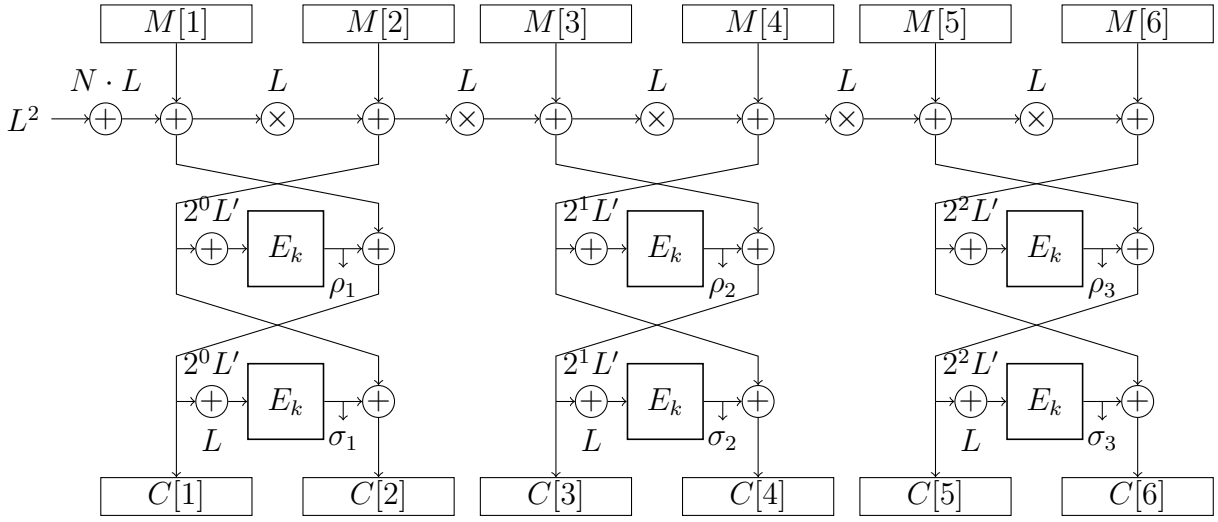

Figure 2: Processing plaintext (for message lengths up to $6n$ bits). Note that L and L' are defined in Fig. 3 below.

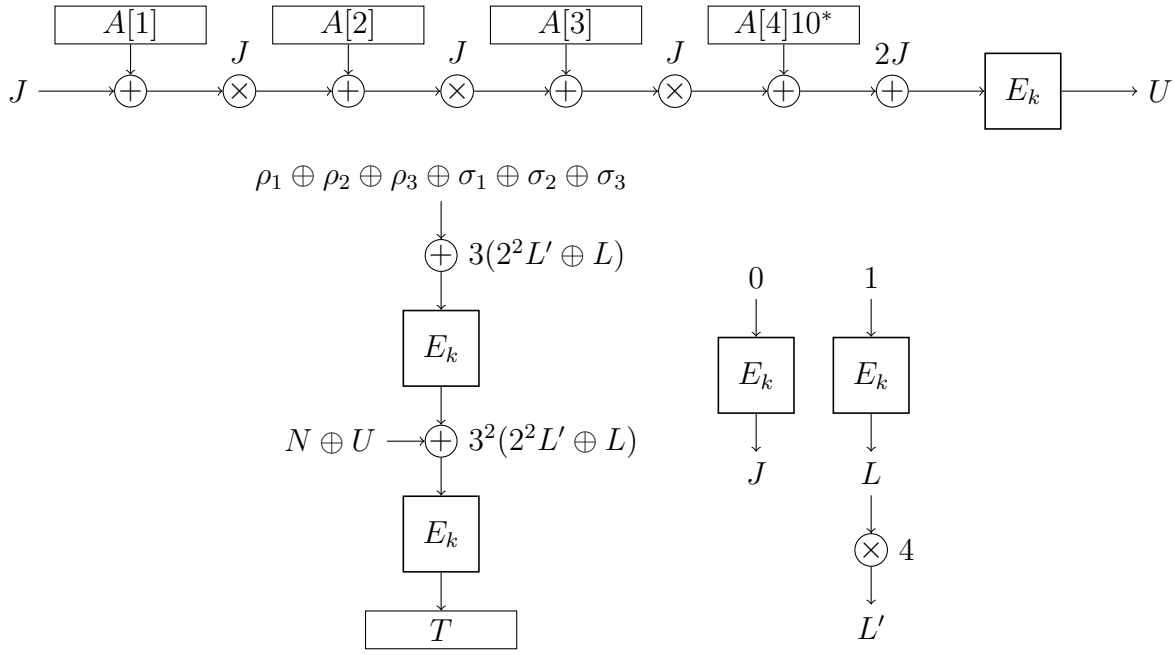

Figure 3: Processing associated data (top, for lengths up to $4n$ bits), computing the tag (bottom left), and the secret values (bottom right).

FRACTIONAL$(V, L, \eta, \Sigma, M[2\ell-1], M[2\ell] \| M^*)$:

$V \leftarrow V \oplus M[2\ell-1]$
$C[2\ell-1] \leftarrow V$
$V \leftarrow (V \otimes L) \oplus (M[2\ell] \| M^*)$
$C[2\ell] \leftarrow V$
$\rho \leftarrow E_k(7\eta \oplus C[2\ell])$
$\Sigma \leftarrow \Sigma \oplus \rho$
$C[2\ell-1] \leftarrow \rho \oplus C[2\ell-1]$
$\sigma \leftarrow E_k(7(\eta \oplus L) \oplus C[2\ell-1])$
$\Sigma \leftarrow \Sigma \oplus \sigma$
$C[2\ell] \leftarrow \sigma \oplus C[2\ell]$
return $(C[2\ell-1], C[2\ell], \eta, \Sigma, V)$

Figure 4: Case 1: $\ell > 1$, $|M[2\ell-1]| = n$, and $0 < |M[2\ell]| < n$

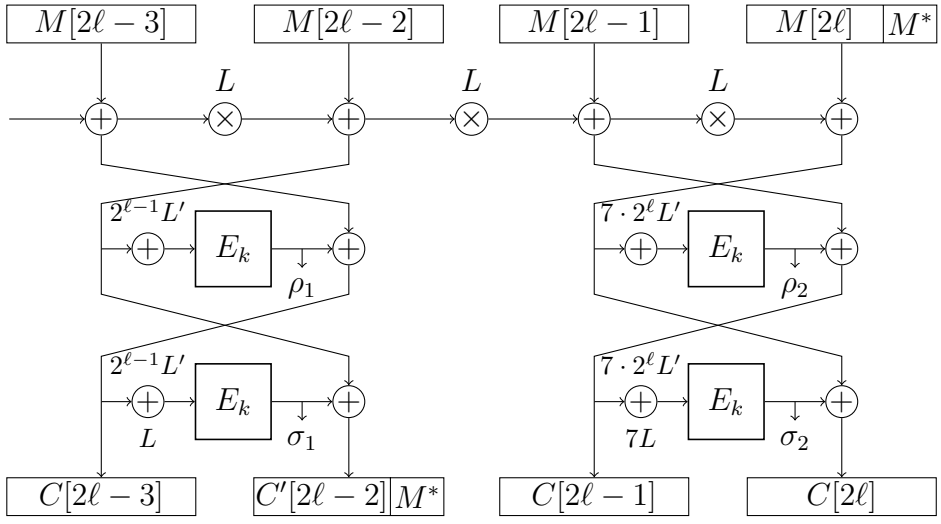

Figure 5: Messages where the last block is not of full length, i.e. $0 < |M[2\ell]| < n$. Here M^* is "stolen" from ciphertext block $C[2\ell-2]$ and used in the input to the final fragment.

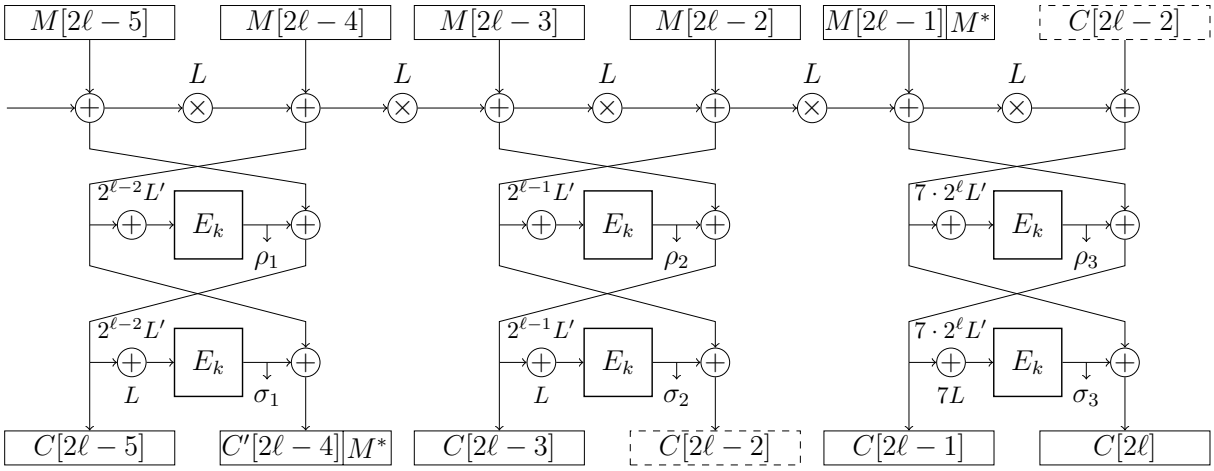

Figure 6: Messages where the last fragment is of length less than or equal to n, i.e. $0 < |M[2\ell-1]| \leq n$. Here M^* is stolen from ciphertext block $C[2\ell-4]$ and used in the input to the final fragment together with ciphertext fragment $C[2\ell-2]$.

1.4.2 Case 2: $\ell > 2$ and $0 < |M[2\ell-1]| \leq n$

When there is no last block $M[2\ell]$, we replace it with the preceding ciphertext block, $C[2\ell-2]$. Then we steal ciphertext M^* of length $n - |M[2\ell-1]|$ from the ciphertext block $C[2\ell-4]$ such that $C[2\ell-4] = C'[2\ell-4]M^*$. The rest of the computation is similar to the previous case (Sect. 1.4.1) and is depicted in Fig. 6.

1.4.3 Case 3: $|M| \leq 3n$

Messages of length $0 \leq |M| < 2n$ are padded using 10^* and then encrypted as usual, but using different tweaks: $7 \cdot L'$ and $7 \cdot (L' \oplus L)$ instead of L' and $L' \oplus L$.

Messages of length $2n$ are encrypted as usual. Messages of length $2n < |M| \leq 3n$ are encrypted as in Sect. 1.4.2, except M^* is replaced with 10^* padding, and the tweaks for the last two block cipher calls are $7^2 \cdot 2L'$ and $7^2 \cdot (2L' \oplus L)$ instead of $7 \cdot 2L'$ and $7 \cdot (2L' \oplus L)$.

2 Security Claims

In this section we specify the security levels with respect to the recommended tag size of 16 bytes, nonce size of 16 bytes, and key size of 16 bytes. For these parameters AES-COBRA achieves the following security levels (in $\log_2$ of number of AES-128 calls):

	AES-COBRA
confidentiality of M	64
integrity of M	64
integrity of A	64
integrity of N	64
security against key recovery	128
security against tag guessing	128

The security levels of AES-COBRA correspond to the birthday bound security on the block size of AES[1] (see also Sect. 3). The security levels apply *both* in the cases when nonces are unique values (full security) and also when nonces are reused (full security up to common prefix, the maximum attainable for single pass schemes). We refer to [3] for the technicalities. Security against key recovery is 128 bits and security against tag guessing is 128 bits.

3 Security Analysis

AES is believed not to be distinguishable from a permutation drawn uniformly at random. AES-COBRA requires only the forward AES operation and no AES block cipher inverse and hence enjoys provable security under the pseudo-random permutation PRP assumption about the underlying block cipher. In [3] we show that under this assumption AES-COBRA cannot be distinguished from an ideal authenticated encryption scheme in up to about 2^{64} AES calls. That is, AES-COBRA is confidentiality secure (in the sense of indistinguishability from an online permutation) against chosen-plaintext (CPA) attacks and integrity secure against forgery up to approximately 2^{64} AES calls. Most importantly, and in contrast with the majority of existing authenticated encryption schemes, AES-COBRA security proofs hold for stronger nonce-repeating attackers and is hence secure against nonce misuse both with respect to confidentiality and integrity.

More precisely, in [3] we prove the nonce misuse security of AES-COBRA in the standard model under the PRP assumption on AES against an AE distinguisher up to the bound $\frac{22(\ell+1)^2q^2}{2^n} + \frac{(3q+1)q_f}{2^n}$, where q are the AES-COBRA queries each of at most 2ℓ blocks and q_f are the forgery attempts. These results mean that under the same key the amount of ciphertext should not exceed 2^{64} blocks. We note, that many existing AES-128 based authenticated encryption schemes with the same security levels conservatively limit the total amount of plaintext and associated data blocks under a fixed key to at most 2^{48} blocks (2^{52} bytes).

When a ciphertext decrypts to $\perp$ the implementor needs to ensure that no information beyond this fact is leaked to the adversary. An adversary can always produce a

[1] We clarify that our security results are of the same order as AES based authenticated encryption schemes, such as AES-GCM and AES-OCB

valid tag T with probability $2^{-|T|}$. It is recommended that applications take care of producing ciphertexts with a single tag size under the same key to avoid potential security degradation.

We clarify that our security model does not encompass timing and power consumption attacks.

4 Features of AES-COBRA

Online. AES-COBRA is designed to allow for online processing for both encryption and decryption, processing data on-the-fly as it arrives.

Nonce misuse resistance. AES-COBRA is designed to maintain security when the nonce is reused. More specifically, it achieves the maximally possible security against nonce reuse for a single-pass authenticated encryption scheme [5], meaning that when two plaintexts are encrypted using nonce N, the adversary can only determine the length of the common plaintext prefix of the two messages, since these will have the same corresponding ciphertext blocks.

Efficiency. AES-COBRA is designed to allow high-performance implementations in both software and hardware.

- Parallelizability: In AES-COBRA, the execution of the AES calls inside the two Feistel functions can be fully parallelized. Furthermore, AES calls of subsequent Feistel blocks can be executed in parallel once the corresponding multiplication chain has been computed. Since the multiplications can be computed independently from the AES calls, they can be executed in parallel to them as well. This results in significant increases in performance for messages longer than two blocks. Likewise, for shorter messages, the available parallelism in AES-COBRA can be exploited by processing multiple of them simultaneously. Both approaches naturally extend to the case where multiple cores are available.

- AES-COBRA is designed for efficiency for both short and long messages. Besides the AES key schedule, the overhead for short messages basically amounts to two AES calls for the tag generation and one AES call for computing the secret value L. On Intel's recent Haswell microarchitecture, AES-COBRA achieves a performance of up to 1.55 cycles/byte (cpb) for longer messages (2048 bytes) and 1.81 cpb for shorter messages (128 bytes).

- Key agility (computational cost under distinct keys): AES-COBRA requires two extra AES call every time a new key is used.

- Nonce agility (computational cost under distinct nonces): Changing the nonce while keeping the key constant requires no additional operations.

- Ability to efficiently preprocess A: Associated data can be preprocessed independently of the message or the value of the nonce.
- Ability to efficiently preprocess plaintext: Similarly, the message (or parts thereof) can be processed in full without seeing A. This requires seeing the nonce, but already produces the full ciphertexts. Only the final tag generation requires A to be processed.

Inverse-free decryption and tag verification. AES-COBRA encryption and decryption/verification require only forward AES calls. This means there is no need to implement the inverse of the AES to implement the full AES-COBRA.

Security proof based on weaker assumptions. The inverse-free property of AES-COBRA also means that it enjoys provable security under the pseudo-random permutation (PRP) assumption about the underlying block cipher, the AES. Modes of operation requiring the computation of its inverse have to rely on a stronger assumption about the block cipher.

Combination of well-known techniques. AES-COBRA relies on design principles of well-known Feistel ciphers in combination with polynomial hashing to achieve integrity of both the associated data and the plaintext. For the confidentiality we use masking techniques which instantiate an XE [7] tweakable block cipher on top of the basic underlying and well-understood AES block cipher. For the integrity AES-COBRA further applies a checksum of intermediate state values of the Feistel structure, similar to ManTiCore [1]. The parallelizability feature is inspired by the OTR [6] design and our main nonce misuse resistance feature by the [2] authenticated encryption design.

4.1 Comparison to AES-GCM

Security against stronger adversaries. Compared to AES-GCM, AES-COBRA guarantees security against a stronger adversary in the nonce misuse setting. In the nonce misuse setting the security of AES-GCM fails completely. Furthermore, AES-COBRA's stronger security guarantee does not require a stronger assumption on the AES than the security proof of AES-GCM (which is in the nonce-respecting setting).

Performance. According to the performance study of [4] on Intel's Haswell platform, AES-COBRA provides a performance which is roughly equal to AES-GCM on shorter messages. For longer messages, AES-GCM is about 33% faster, however providing no guarantees when nonces are being repeated.

5 Design Rationale

AES-COBRA has been designed to allow for high performance in parallel environments and to maintain security even if nonce is reused. Among other platforms, AES-COBRA is well-suited for both parallel software and high-performance hardware, being based on AES and multiplications in the finite field. In software, AES-COBRA is efficient on recent Intel microarchitectures featuring hardware support for AES and finite field multiplications, especially on the latest Intel microarchitecture Haswell with improved finite field multiplication capabilities. AES is well-studied and wide-spread (including implementations and countermeasures against side-channel analysis), being, thus, the natural choice for the underlying block cipher.

The employment of the Feistel network seems necessary for efficiently authenticating and encrypting at the same time using polynomial hashing. It additionally allows for a scheme that does not need the inverse of the block cipher upon decryption. To achieve confidentiality we use masking techniques which instantiate an XE [7] tweakable block cipher on top of the AES block cipher. In order to achieve integrity, AES-COBRA utilizes the checksum of intermediate state values of the Feistel structure, similarly to ManTiCore [1]. The parallelizability in AES-COBRA takes ideas from the OTR [6] design and the nonce misuse resistance security by the [2] authenticated encryption design.

The designers have not hidden any weaknesses in AES-COBRA.

6 Intellectual Property

The submitters are not aware of any patent involved in AES-COBRA. Furthermore, AES-COBRA will not be patented. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

7 Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitters understand that if they disagree with published analyses then they are expected to

promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

References

[1] Anderson, E., Beaver, C.L., Draelos, T., Schroeppel, R., Torgerson, M.: ManTiCore: Encryption with Joint Cipher-State Authentication. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP. Lecture Notes in Computer Science, vol. 3108, pp. 440–453. Springer (2004)

[2] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT (1). Lecture Notes in Computer Science, vol. 8269, pp. 424–443. Springer (2013)

[3] Andreeva, E., Luykx, A., Mennink, B., Yasuda, K.: COBRA: A Parallelizable Authenticated Online Cipher Without Block Cipher Inverse. In: FSE 2014. Lecture Notes in Computer Science, Springer (2014), to appear

[4] Bogdanov, A., Lauridsen, M.M., Tischhauser, E.: AES-Based Authenticated Encryption Modes in Parallel High-Performance Software. Cryptology ePrint Archive, Report 2014/186 (2014), `http://eprint.iacr.org/`

[5] 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)

[6] Minematsu, K.: Parallelizable Authenticated Encryption from Function. Cryptology ePrint Archive, Report 2013/628 (2013)

[7] 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)

[8] Rogaway, P., Wooding, M., Zhang, H.: The Security of Ciphertext Stealing. In: Canteaut, A. (ed.) FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 180–195. Springer (2012)