COLM v1

Preliminary document

Designers/Submitters:
Elena Andreeva — KU Leuven and iMinds, Belgium,
Andrey Bogdanov — DTU Compute, Denmark,
Nilanjan Datta — Indian Statistical Institute, India,
Atul Luykx — KU Leuven and iMinds, Belgium,
Bart Mennink — KU Leuven and iMinds, Belgium,
Mridul Nandi — Indian Statistical Institute, India,
Elmar Tischhauser — DTU Compute, Denmark,
Kan Yasuda — NTT Secure Platform Laboratories, Japan

colm@colm.ae

May 11, 2016

1 Specification

At a high level, COLM can be seen as a block cipher based Encrypt-Linear mix-Encrypt mode, designed with the goal to achieve online misuse resistance, to be fully parallelizable, and to be secure against blockwise adaptive adversaries. In this section we provide a definition of the COLM family of authenticated ciphers, which includes the parameter space defining the family and a list of recommended parameter sets.

1.1 Notation

By $\{0,1\}^*$ we denote the set of all strings, and by $\{0,1\}^n$ the set of strings of length n. 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. By $\lfloor B \rfloor_s$ we denote the s most significant bits of X.

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 $C = E(k, M) = E_k(M)$. Throughout the document E denotes the block cipher AES-128, with $\kappa = n = 128$.

A block is a string of length at most 128 bits. It is complete if it has length 128 bits, and incomplete otherwise. The set of complete blocks is denoted $\mathbb{B} = \{0,1\}^{128}$. Any string $B \in \{0,1\}^r$, can be represented as a sequence of blocks $(B[1], B[2], \ldots, B[\ell-1], B^*[\ell])$, where $\ell = \lceil \frac{r}{128} \rceil$. For $i < \ell$, $B[i]$ is the ith complete block of B, and $B^*[\ell]$ is the final block of B which may be incomplete. For $0 \le a \le b < \ell$ we denote $B[a, \ldots, b] = (B[a], B[a+1], \ldots, B[b])$, and $B[, \ldots, b] = B[1, \ldots, b]$.

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 $\mathrm{GF}(2^n)$.

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 $\mathrm{GF}(2^n)$.

1.2 Parameters

COLM uses AES-128 with a key and state of size 128 bits. It has two parameters:

- τ: the number of blocks after which intermediate tags are generated, where $\tau \in \{0, \ldots, 127\}$ and $\tau = 0$ by definition means that no intermediate tags are generated;
- l_τ: the length of the intermediate tags, where $l_\tau \in \{64, \ldots, 128\}$.

The first recommended parameters set is $(\tau, l_\tau) = (0, 128)$, the second recommended parameter set is $(\tau, l_\tau) = (127, 128)$. For low end devices, when intermediate tags are beneficial, we suggest $\tau = 127$. As we keep $l_\tau = 128$, we leave it implicit throughout. We simply denote COLM_τ to refer to COLM with the particular choice of τ.

1.3 Linear Mixing Function ρ

COLM internally uses a linear mixing function ρ, that takes two inputs $x \in \mathbb{B}$ and $st \in \mathbb{B}$ and gives $y \in \mathbb{B}$ and $st' \in \mathbb{B}$ in the following way:

$$\begin{aligned} y &= x \oplus 3 \cdot st\,, \\ st' &= x \oplus 2 \cdot st\,. \end{aligned}$$

Now, as y and st' are linear functions of x and st, we can represent x and st' as a linear combination of y and st: $x = y \oplus 3 \cdot st$ and $st = (y \oplus 3 \cdot st) \oplus 2 \cdot st = y \oplus st$. We call this linear function ρ^{-1}. So, ρ^{-1} is a linear function that takes two inputs $y, st \in \mathbb{B}$ and gives $x \in \mathbb{B}$ and $st' \in \mathbb{B}$ in the following way:

$$\begin{aligned} x &= y \oplus 3 \cdot st \\ st' &= y \oplus st \end{aligned}$$

1.4 COLM Encryption

COLM encryption takes the following inputs:

- Encryption key $K \in \{0,1\}^{128}$.
- Public message number, or nonce, npub $\in \{0,1\}^{64}$.
- Parameter set param $\in \{0,1\}^{64}$. The most significant 16 bits represent the intermediate tag interval τ, the next 8 bits denote the intermediate tag length l_τ. The remaining 40 bits are kept as optional for future use and currently assigned to the fixed value 0^{40}.
- Associated data $A \in \{0,1\}^*$, with the restriction that $0 \leq |A| \leq 2^{64}$.
- Message (or plaintext) $M \in \{0,1\}^*$, with the restriction that $1 \leq |M| \leq 2^{64}$.

Upon these inputs, COLM_τ returns a *tagged ciphertext* $C \in \{0,1\}^{|M|+128}$, and in case of $\tau > 0$ a list of intermediate tags $T \in \{0,1\}^{128 \cdot h}$, where $h = \lfloor \frac{l-1}{127} \rfloor$ denotes the number of intermediate tags generated and $l = \lceil \frac{|M|}{128} \rceil$, denotes the number of blocks of M.

COLM_0 – AEAD without Intermediate Tags

COLM_0 is depicted in Figure 1.4.

Subkey Generation. As a first step, $L = E_K(0)$ is computed.

Initial Value Generation. The initial value is generated from the associated data. Split the associated data A into

$$A = (A[1], \ldots, A[a-1], A^*[a]),$$

where $|A^*[a]| \leq 128$ and for $i = 1, \ldots, a-1$, $|A[i]| = 128$. If the final associated data block $A^*[a]$ is incomplete, we make it complete by 10^* masking: $A[a] = A^*[a]10^*$. Otherwise we set $A[a] = A^*[a]$. The initial value IV is

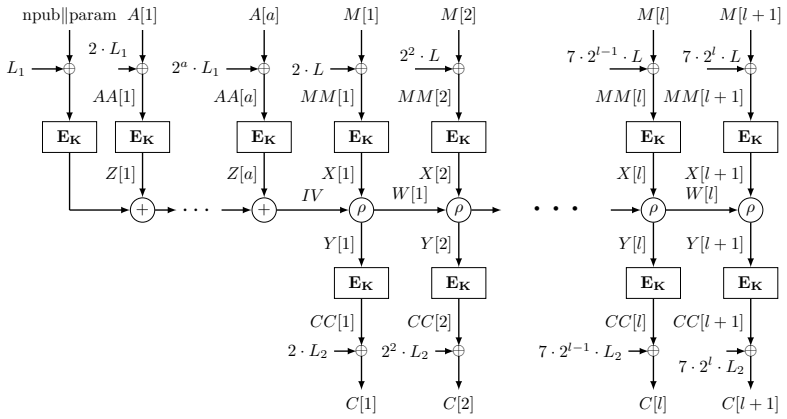

Figure 1.1: COLM authenticated encryption for complete final message block. Here we denote $L_1 = 3 \cdot L$ and $L_2 = 3^2 \cdot L$.

computed as follows:

$$\begin{aligned}
W'[0] &= E_K((\text{npub}\|\text{param}) \oplus \Delta_A[0])\,, \\
AA[i] &= A[i] \oplus \Delta_A[i] && \text{for } i = 1, \ldots, a\,, \\
Z[i] &= E_K(AA[i]) && \text{for } i = 1, \ldots, a\,, \\
W'[i] &= Z[i] \oplus W'[i-1] && \text{for } i = 1, \ldots, a\,, \\
IV &= W'[a]\,,
\end{aligned}$$

where the masking values are computed as follows:

$$\Delta_A[i] = \begin{cases} 3 \cdot 7 \cdot 2^{i-1} \cdot L & \text{if } i = a \text{ and } |A^*[a]| < 128 \\ 3 \cdot 2^i \cdot L & \text{otherwise.} \end{cases}$$

Tagged Ciphertext Generation. The tagged ciphertext is generated using the message M and the IV. Split the message M into

$$M = (M[1], \ldots, M[l-1], M^*[l])\,,$$

where $|M^*[l]| \leq 128$ and for $i = 1, \ldots, l-1$, $|M[i]| = 128$. We pad $M[l]$ and generate $M[l+1]$ as follows:

$$\begin{aligned} M[l] &= M[1] \oplus \cdots \oplus M[l-1] \oplus (M^*[l]10^*) \\ M[l+1] &= M[l]\,. \end{aligned}$$

Now, we generate the tagged ciphertext C from the padded message $(M[1], \ldots, M[l+1])$ as follows:

$$\begin{aligned} W[0] &= IV\,, & \\ MM[i] &= M[i] \oplus \Delta_M[i] & \text{for } i = 1, \ldots, l+1\,, \\ X[i] &= E_K(MM[i]) & \text{for } i = 1, \ldots, l+1\,, \\ (Y[i], W[i]) &= \rho(X[i], W[i-1]) & \text{for } i = 1, \ldots, l+1\,, \\ CC[i] &= E_K(Y[i]) & \text{for } i = 1, \ldots, l+1\,, \\ C[i] &= CC[i] \oplus \Delta_C[i] & \text{for } i = 1, \ldots, l+1\,, \end{aligned}$$

where the masking values are computed as follows:

$$\Delta_M[i] = \begin{cases} 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |M^*[l]| = 128\,, \\ 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |M^*[l]| < 128\,, \\ 2^i \cdot L & \text{otherwise,} \end{cases}$$

$$\Delta_C[i] = \begin{cases} 3^2 \cdot 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |M^*[l]| = 128\,, \\ 3^2 \cdot 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |M^*[l]| < 128\,, \\ 3^2 \cdot 2^i \cdot L & \text{otherwise.} \end{cases}$$

The algorithm returns the tagged ciphertext

$$C = (C[1], \ldots, C[l], \lfloor C[l+1] \rfloor_{|M^*[l]|})\,.$$

$\mathbf{COLM}_{127}$ – AEAD with Intermediate Tags

COLM_{127} is similar to COLM_0 except for the generation of intermediate tags and some changes in the ciphertext masking, and we only discuss the changes. Recall that $h = \lfloor \frac{(l-1)}{127} \rfloor$ is the number of intermediate tags generated for M. Let $h_i = \lfloor \frac{(i-1)}{127} \rfloor$ denote the number of intermediate tags generated during the message processing up to ith block.

The ciphertext is now computed as

$$C[i] = CC[i] \oplus \Delta_C[i + h_i] \qquad \text{for } i = 1, \ldots, l + 1\,,$$

and the intermediate tags as

$$\begin{aligned} TT[j] &= E_K(W[127 \cdot j]) && \text{for } j = 1, \ldots, h\,, \\ T[j] &= TT[j] \oplus \Delta_C[128 \cdot j] && \text{for } j = 1, \ldots, h\,, \end{aligned}$$

where the masking values are now computed as follows:

$$\Delta_C[i] = \begin{cases} 3^2 \cdot 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l + h, l + h + 1\} \text{ and } |M^*[l]| = 128\,, \\ 3^2 \cdot 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l + h, l + h + 1\} \text{ and } |M^*[l]| < 128\,, \\ 3^2 \cdot 2^i \cdot L & \text{otherwise.} \end{cases}$$

1.5 COLM Decryption and Tag Verification

COLM decryption takes the following inputs:

- Encryption key $K \in \{0,1\}^{128}$.
- Public message number, or nonce, npub $\in \{0,1\}^{64}$.
- Parameter set param $\in \{0,1\}^{64}$, as defined for the encryption.
- Associated data $A \in \{0,1\}^*$.
- Tagged ciphertext $C \in \{0,1\}^+$, and for COLM_{127} additionally the list of intermediate tags $T \in \{0,1\}^{128 \cdot h}$.

Upon these inputs, it returns the corresponding plaintext M if verification succeeds, and $\perp$ otherwise.

The decryption and verification is a performed as a three step process: First, the subkey L and the initial value IV are generated using the same procedure as for the encryption. Then the decryption is performed, followed by the tag verification.

1.5.1 COLM$_0$ Decryption

For the case of COLM$_0$ (with no intermediate tags), the decryption procedure is as follows.

$$\begin{aligned}
W[0] &= IV\,, \\
CC[i] &= C[i] \oplus \Delta_C[i] && \text{for } i = 1, \ldots, l\,, \\
Y[i] &= E_K^{-1}(CC[i]) && \text{for } i = 1, \ldots, l\,, \\
(X[i], W[i]) &= \rho^{-1}(Y[i], W[i-1]) && \text{for } i = 1, \ldots, l\,, \\
MM[i] &= E_K^{-1}(X[i]) && \text{for } i = 1, \ldots, l\,, \\
M[i] &= MM[i] \oplus \Delta_M[i] && \text{for } i = 1, \ldots, l\,, \\
M^*[l] &= M[1] \oplus \cdots \oplus M[l]\,, \\
M[l+1] &= M[l]\,,
\end{aligned}$$

where the masking values are computed as follows (identical to the enryption):

$$\Delta_M[i] = \begin{cases} 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |C[l+1]| = 128\,, \\ 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |C[l+1]| < 128\,, \\ 2^i \cdot L & \text{otherwise,} \end{cases}$$

$$\Delta_C[i] = \begin{cases} 3^2 \cdot 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |C[l+1]| = 128\,, \\ 3^2 \cdot 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l, l+1\} \text{ and } |C[l+1]| < 128\,, \\ 3^2 \cdot 2^i \cdot L & \text{otherwise.} \end{cases}$$

1.5.2 COLM$_{127}$ Decryption

For the variant with intermediate tags every 127 blocks, the decryption is similar to COLM$_0$ except for the two following changes:

$$CC[i] = C[i] \oplus \Delta_C[i + h_i] \qquad \text{for } i = 1, \ldots, l+1\,,$$

and

$$\Delta_C[i] = \begin{cases} 3^2 \cdot 7 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l+h, l+h+1\} \text{ and } |C[l+1]| = 128\,, \\ 3^2 \cdot 7^2 \cdot 2^{i-1} \cdot L & \text{if } i \in \{l+h, l+h+1\} \text{ and } |C[l+1]| < 128\,, \\ 3^2 \cdot 2^i \cdot L & \text{otherwise;} \end{cases}$$

with h_i and h defined as in the encryption routine of COLM$_{127}$.

1.5.3 COLM$_0$ Verification

For the case of COLM$_0$ (without intermediate tags), the verification is performed as follows.

$$\begin{aligned} MM[l+1] &= M[l+1] \oplus \Delta_M[l+1]\,, \\ X[l+1] &= E_K(MM[l+1])\,, \\ (Y[l+1], W[l+1]) &= \rho(X[l+1], W[l])\,, \\ CC[l+1] &= E_K(Y[l+1])\,, \\ C'[l+1] &= CC[l+1] \oplus \Delta_C[l+1]. \end{aligned}$$

The verification now succeeds if:

- for $|C[l+1]| = 128$, we have $C[l+1] = C'[l+1]$;
- for $|C[l+1]| < 128$, we have $C[l+1] = \lfloor C'[l+1] \rfloor_{|C[l+1]|}$ and the last $128 - |C[l+1]|$ bits of $M^*[l]$ are 10^*.

Upon successful verification, the plaintext $M[1], \ldots, \lfloor M^*[l] \rfloor_{|C[l+1]|})$ is returned. Otherwise, the output is $\perp$.

1.5.4 COLM$_{127}$ Verification

For the variant with intermediate tags every 127 blocks, the verification is similar to COLM$_0$ with the *additional* verification of all intermediate tags. For this, one computes

$$\begin{aligned} TT[j] &= T[j] \oplus \Delta_C[128 \cdot j] && \text{for } j = 1, \ldots, h\,, \\ W'[j] &= E_K^{-1}(TT[j]) && \text{for } j = 1, \ldots, h\,; \end{aligned}$$

and verifies that the following condition is met:

$$W[127 \cdot j] = W'[j] \qquad \text{for } j = 1, \ldots, h.$$

Note that the masks Δ_M and Δ_C for COLM$_{127}$ differ from those of COLM$_0$.

2 Comparison with COPA and ELmD

In a nutshell, COLM resembles the best of both COPA [2] and ELmD [3]:

- The parallelizable character of both, with one difference: COPA uses PMAC for authentication, while ELmD uses PHASH. PHASH is, unlike PMAC, fully parallelizable. For COLM, we have opted to follow the PHASH approach.
- COPA uses encryption in both layers, while ELmD uses encryption in the top layer and decryption in the bottom. COLM uses encryption in both layers, following COLM.
- The simple linear mixing in-between the two encryption layers. We have opted to use simple XOR for the authentication part, and the function ρ for encryption.
- The masking of the blockciphers is simplified in comparison with both COPA and ELmD.
- Intermediate tags are supported, following ELmD.

The major differences between COPA/ELmD on the one hand and COLM on the other hand are summarized as follows:

	COPA	ELmD	COLM
Simplified masking	✗	✗	✓
Fully parallelizable authentication	✗	✓	✓
XOR mixing for authentication	✓	✗	✓
ρ mixing for encryption	✗	✓	✓
Bottom layer encryption	✓	✗	✓
Intermediate tags	✗	✓	✓

3 Security Goals

COLM achieves the following security levels:

	COLM_0	COLM_{127}
confidentiality for the plaintext	64	64
integrity for the plaintext	64	64
integrity for the associated data	64	64
integrity for the public message number	64	64

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). The security levels correspond to the birthday bound security of the block size of AES, and are supported by security proofs on COPA [1, 2] and ELmD [4, 3]. A dedicated security analysis for COLM will follow soon.

4 Intellectual Property

COLM uses a PMAC-like construction for achieving integrity. According to its designers, one or more patents related to PMAC are pending. Depending on the breadth of the claims in these patents, this part of COLM could be covered by them.

The submitters however explicitly state that COLM itself will not be patented.

5 Consent

The submitter/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 submitter/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 submitter/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 submitter/submitters acknowledge that the committee decisions reflect the

collective expert judgments of the committee members and are not subject to appeal. The submitter/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 submitter/submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

References

[1] 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)

[2] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v.1 (2014), submission to CAESAR competition

[3] Datta, N., Nandi, M.: ELmD v1.0 (2014), submission to CAESAR competition

[4] Datta, N., Nandi, M.: Elme: A misuse resistant parallel authenticated encryption. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. Lecture Notes in Computer Science, vol. 8544, pp. 306–321. Springer (2014)