Fractional Data for Nonce-Misuse Resistant Mode for Kiasu, Joltik and Deoxys

Jérémy Jean, Ivica Nikolić, Thomas Peyrin

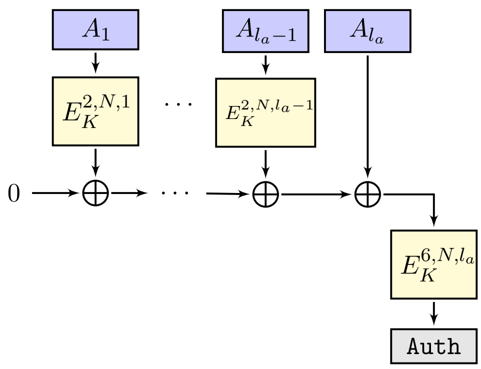
Nanyang Technological University, Singapore.
{JJean,INikolic,Thomas.Peyrin}@ntu.edu.sg

http://www1.spms.ntu.edu.sg/~syllab/CAESAR

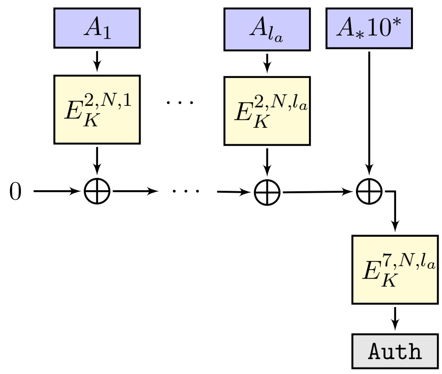
April 2, 2014

As mentioned in the original submission documents, KIASU, Joltik and Deoxys support fractional messages, which have not necessarily a length multiple of the block size n. As in the COPA [1] article, they make use of two different techniques: first, tag splitting [2] in the case where the size $|M|$ of the message M is strictly smaller than n, and second, the XLS technique [3] in the case where $|M|$ is strictly greater than n, while not being a multiple of n. This was not described in details in the original submission documents and we give in this add-on a full specification of the COPA mode for KIASU, Joltik and Deoxys. We emphasize that empty messages should be treated as partial block, and therefore need 10^* padding.

Notations. In the sequel, we denote $\lceil X \rceil_n$ the value X truncated to its first n bits, and $\lfloor X \rfloor_n$ the value X truncated to its last n bits. Moreover, $X \lll a$ will denote the word X rotated by a positions to the left. We recall that $E_K(T, M)$ refers to the encryption of message block M using tweak T and key K, while $D_K(T, M)$ denotes the decryption operation on the same inputs.

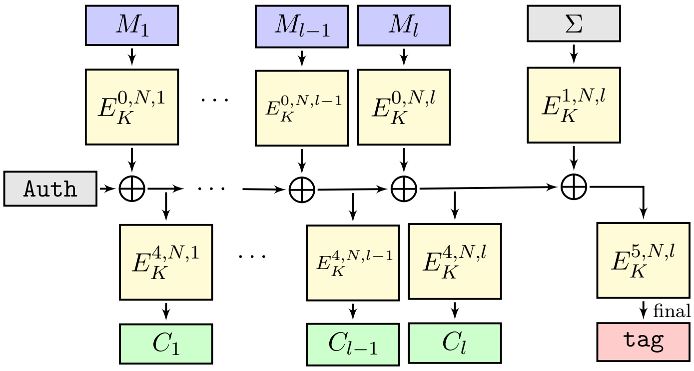


(a) Case where $|A|$ is a multiple of n.

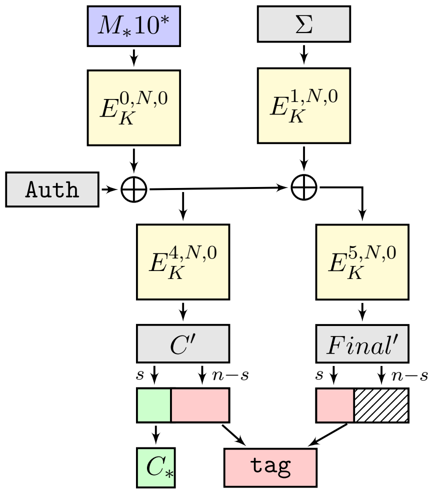


(b) Case where $|A|$ is **not** a multiple of n.

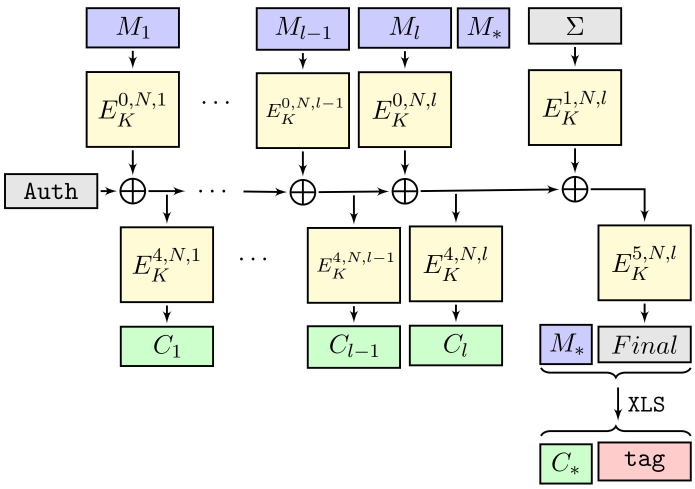
Figure 1: Handling the associated data A of length $|A|$. We distinguish two cases, whether $|A|$ is a multiple of the block size n or not.



(a) Case $|M| \geq n$ is a multiple of n.



(b) Case $|M| = s < n$. We use the tag splitting technique.



(c) Case $|M| > n$ and is **not** a multiple of n. We use `XLS`.

Figure 2: Handling the message M of length $|M|$. We distinguish three cases depending on the value of $|M|$ in comparison to the block size n.

Algorithm 1: The encryption algorithm $\mathcal{E}_K^{\equiv}(N, A, M)$. The value N is encoded on $\log_2(max_m)$ bits, while the integer values i, l and l_a are encoded on $\log_2(max_l)$ bits.

/ Associated data */*
$A_1||\ldots||A_{l_a}||A_* \leftarrow A$ where each $|A_i| = n$ and $|A_*| < n$
$\text{Auth} \leftarrow 0^n$
for $i = 1$ **to** $l_a - 1$ **do**
    $\text{Auth} \leftarrow \text{Auth} \oplus E_K(0010||N||i, A_i)$
end
if $A_* \neq \epsilon$ **then**
    $\text{Auth} \leftarrow \text{Auth} \oplus E_K(0010||N||l_a, A_{l_a})$
    $\text{Auth} \leftarrow \text{Auth} \oplus pad10^*(A_*)$
    $\text{Auth} \leftarrow E_K(0111||N||l_a, \text{Auth})$
else
    $\text{Auth} \leftarrow \text{Auth} \oplus A_{l_a}$
    $\text{Auth} \leftarrow E_K(0110||N||l_a, \text{Auth})$
end

/ Message */*
if $|M| < n$ **then**
    $M_* \leftarrow pad10^*(M)$
    $\text{Auth} \leftarrow \text{Auth} \oplus E_K(0000||N||0, M_*)$
    $C' \leftarrow E_K(0100||N||0, \text{Auth})$
    $\text{Auth} \leftarrow \text{Auth} \oplus E_K(0001||N||0, M_*)$
    $\text{Final}' \leftarrow E_K(0101||N||0, \text{Auth})$
    $C \leftarrow \lceil C' \rceil_{|M|}$
    $\texttt{tag} \leftarrow \lfloor C' \rfloor_{n-|M|} \;||\; \lceil \text{Final}' \rceil_{|M|}$
    return $(C, \texttt{tag})$
end

$M_1||\ldots||M_l||M_* \leftarrow M$ where each $|M_i| = n$ and $|M_*| < n$
$\text{Checksum} \leftarrow 0^n$
for $i = 1$ **to** l **do**
    $\text{Checksum} \leftarrow \text{Checksum} \oplus M_i$
    $\text{Auth} \leftarrow \text{Auth} \oplus E_K(0000||N||i, M_i)$
    $C_i \leftarrow E_K(0100||N||i, \text{Auth})$
end
$C_* \leftarrow \epsilon$
$\text{Auth} \leftarrow \text{Auth} \oplus E_K(0001||N||l, \text{Checksum})$
$\text{Final} \leftarrow E_K(0101||N||l, \text{Auth})$
if $M_* \neq \epsilon$ **then**
    $C_* \;||\; \text{Final} \leftarrow \texttt{XLS}(M_* \;||\; \text{Final}, l)$, with $|C_*| = |M_*|$
end
$\texttt{tag} \leftarrow \text{Final}$
return $(C_1||\ldots||C_l||C_*, \texttt{tag})$

Algorithm 2: The verification/decryption algorithm $\mathcal{D}_K^{\equiv}(N, A, C, \texttt{tag})$. The value N is encoded on $\log_2(max_m)$ bits, while the integer values i, l and l_a are encoded on $\log_2(max_l)$ bits.

```
/* Associated data */
A_1||...||A_{l_a}||A_* ← A where each |A_i| = n and |A_*| < n
Auth ← 0^n
for i = 1 to l_a − 1 do
  | Auth ← Auth ⊕ E_K(0010||N||i, A_i)
end
if A_* ≠ ϵ then
  | Auth ← Auth ⊕ E_K(0010||N||l_a, A_{l_a})
  | Auth ← Auth ⊕ pad10*(A_*)
  | Auth ← E_K(0111||N||l_a, Auth)
else
  | Auth ← Auth ⊕ A_{l_a}
  | Auth ← E_K(0110||N||l_a, Auth)
end

/* Ciphertext */
if |C| < n then
  | C' ← C_* || ⌈tag⌉_{n−s}
  | X ← D_K(0100||N||0, C')
  | M' ← D_K(0000||N||0, Auth ⊕ X)
  | M_* ← unpad01*(M')
  | Checksum ← Checksum ⊕ M'
  | Auth ← X ⊕ E_K(0001||N||0, Checksum)
  | Final' ← E_K(0101||N||0, Auth)
  | if |M_*| = |C_*| and ⌈Final'⌉_s = ⌊tag⌋_s then return M_*
  | else return ⊥
end

C_1||...||C_l||C_* ← C where each |C_i| = n and |C_*| < n
Checksum ← 0^n
for i = 1 to l do
  | X_i ← D_K(0100||N||i, C_i)
  | M_i ← D_K(0100||N||i, X_i ⊕ Auth)
  | Checksum ← Checksum ⊕ M_i
  | Auth ← X_i
end
M_* ← ϵ
Auth ← Auth ⊕ E_K(0001||N||l, Checksum)
Final ← E_K(0101||N||l, Auth)
if C_* ≠ ϵ then
  | M_* || Final' ← XLS^{−1}(C_* || tag), with |M_*| = |C_*|
  | if Final ≠ Final' then return ⊥
else
  | if Final ≠ tag then return ⊥
end
return M_1||...||M_l||M_*
```

Algorithm 3: `XLS` algorithm: extending an n-bit cipher to an $(n+s)$-bit cipher ($s < n$).

Input: An $(n+s)$-bit value M, a counter l
Output: An $(n+s)$-bit value C

$(M_1, M_2) \leftarrow (\lceil M \rceil_n, \lfloor M \rfloor_s)$

$X_1 \leftarrow E_K(1000||N||l, M_1)$
$(X_{1,n-s}, X_{1,s}) \leftarrow (\lceil X_1 \rceil_{n-s}, \lfloor X_1 \rfloor_s)$
$X'_{1,n-s} \leftarrow X_{1,n-s} \oplus 1$
$(X'_{1,s}, X_2) \leftarrow \mathbf{mix}(X_{1,s}, M_2)$
$X'_1 \leftarrow X'_{1,n-s} \,||\, X'_{1,s}$

$Y_1 \leftarrow E_K(1001||N||l, X'_1)$
$(Y_{1,n-s}, Y_{1,s}) \leftarrow (\lceil Y_1 \rceil_{n-s}, \lfloor Y_1 \rfloor_s)$
$Y'_{1,n-s} \leftarrow Y_{1,n-s} \oplus 1$
$(Y'_{1,s}, C_2) \leftarrow \mathbf{mix}(Y_{1,s}, X_2)$
$Y'_1 \leftarrow Y'_{1,n-s} \,||\, Y'_{1,s}$

$C_1 \leftarrow E_K(1000||N||l, Y'_1)$
$C \leftarrow C_1 \,||\, C_2$
return C

Algorithm 4: `XLS` $^{-1}$ algorithm: inverting the `XLS` algorithm 3.

Input: An $(n+s)$-bit value C, a counter l
Output: An $(n+s)$-bit value M

$(C_1, C_2) \leftarrow (\lceil C \rceil_n, \lfloor C \rfloor_s)$

$Y'_1 \leftarrow E_K^{-1}(1000||N||l, C_1)$
$(Y'_{1,n-s}, Y'_{1,s}) \leftarrow (\lceil Y'_1 \rceil_{n-s}, \lfloor Y'_1 \rfloor_s)$
$Y_{1,n-s} \leftarrow Y'_{1,n-s} \oplus 1$
$(Y_{1,s}, X_2) \leftarrow \mathbf{mix}(Y'_{1,s}, C_2)$
$Y_1 \leftarrow Y_{1,n-s} \,||\, Y_{1,s}$

$X'_1 \leftarrow E_K^{-1}(1001||N||l, Y_1)$
$(X'_{1,n-s}, X'_{1,s}) \leftarrow (\lceil X'_1 \rceil_{n-s}, \lfloor X'_1 \rfloor_s)$
$X_{1,n-s} \leftarrow X'_{1,n-s} \oplus 1$
$(X_{1,s}, M_2) \leftarrow \mathbf{mix}(X'_{1,s}, X_2)$
$X_1 \leftarrow X_{1,n-s} \,||\, X_{1,s}$

$M_1 \leftarrow E_K^{-1}(1000||N||l, X_1)$
$M \leftarrow M_1 \,||\, M_2$
return M

Algorithm 5: The **mix** function used in `XLS`. Note that $\mathbf{mix}^{-1} = \mathbf{mix}$.

Input: A $2s$-bit value X
Output: A $2s$-bit value Y
$(A, B) \leftarrow (\lceil X \rceil_s, \lfloor X \rfloor_s)$
$S \leftarrow (A \oplus B) \lll 1$
$Y \leftarrow (A \oplus S) \,||\, (B \oplus S)$
return Y

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). Volume 8269 of Lecture Notes in Computer Science., Springer (2013) 424–443

[2] Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In Canteaut, A., ed.: FSE 2012. Volume 7549 of LNCS., Springer (March 2012) 196–215

[3] Ristenpart, T., Rogaway, P.: How to Enrich the Message Space of a Cipher. In Biryukov, A., ed.: FSE 2007. Volume 4593 of LNCS., Springer (March 2007) 101–118