Errata of ORANGE

Designers/Submitters:
Bishwajit Chakraborty - Indian Statistical Institute,Kolkata
Mridul Nandi - Indian Statistical Institute, Kolkata, India

bishu.math.ynwa@gmail.com
mridul.nandi@gmail.com

May 1, 2019

In spec_orange.pdf: We make the following corrections on the specification document. No corresponding change is required in the reference implementations.

1. ORANGE-Zest$_{[P]}$.enc **line 9**: return value $(C, \mathsf{proc\_tg}(U))$ and not $\mathsf{proc\_tg}(U)$.

2. function ORANGISH **line 21**: There is no α multiplication in Hash. So $Z \leftarrow \mathsf{proc\_hash}(X, (A_{d-1}\| \ldots \|, A_0), 0, 0)$ and not $Z \leftarrow \mathsf{proc\_hash}(X, (A_{d-1}\| \ldots \|, A_0), 1, 1)$.

3. function proc_txt **line 37**: return value is (D', U_d) and not (D', U_a).

4. function ORANGE-Zest$_{[P]}$.dec :
   (a) **line 5** : $a = 0$ is changed to $a = 0, m \neq 0$.
   (b) **line 9** : $m \neq 0$ is changed to $a \neq 0, m \neq 0$.

5. function mult **line 32** : return value is $\alpha^c \cdot V^b \parallel V^t$ and not $V^t \parallel \alpha^c \cdot V^b$.

In crypto_aead/orangezestv1/ref/orangemodule.h: The primitive polynomial $alpha\_128$ was getting reset to x^{128} instead of $x^{128} + x^7 + x^2 + x + 1$. (in **lines 84-90** of orangemodule.h). This is corrected by using an **else** argument in **line 88** of orangemodule.h.

The revised **Test vectors for ORANGE-Zest** in **Appendix B** can be found bellow :

Test vectors for ORANGE-Zest

Test vector 1:
$Key = 000102030405060708090A0B0C0D0E0F$
$Nonce = 000102030405060708090A0B0C0D0E0F$
$PT =$
$AD = 00010203$
$CT = 84A4C553119EA342C50CCCCE43782567$

Test vector 2:
$Key = 000102030405060708090A0B0C0D0E0F$
$Nonce = 000102030405060708090A0B0C0D0E0F$
$PT =$
$AD =$
$CT = 5A65624E01D1349D2211EFBD52217976$

Test vector 3:
$Key = 000102030405060708090A0B0C0D0E0F$

$Nonce = 000102030405060708090A0B0C0D0E0F$
$PT = 000102030405060708090A0B0C0D0E0F101112131415161718191A$
$AD = 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D$
$CT = 06C8617CFB5C8CACA64F1F2B9460EADE7776AB0F814F4CFB0E561C621AB9EB080D6CE$
$0D200E80EE74E8C00$

Acknowledgment: We would like to thank Miguel Montes for pointing out the mismatch between reference implementation and pseudocode specification.