5. The EME2-AES Transform

5.1 The Mult-by-alpha Operation

The encryption and decryption procedures described in the following sections use a function Mult-by-alpha(X), that multiplies a 16-byte value X by a primitive element α in the field GF(2^{128}). The input value is first converted into a byte array X[i], i = 0,1,...,15, where X[0] is the first byte of the byte array.

The multiplication by alpha is defined by the following, or a mathematically equivalent, procedure:

```
Mult-by-alpha(X)
   Input: byte array X[i], i = 0,1,...,15

   Output: byte array Y[i], i = 0,1,...,15

   for i=0 to 15 do
      Y[i] = 2*X[i] mod 256
      if (i>0 and X[i-1]>127) then Y[i]=Y[i]+1
   end-for
   if (X[15] > 127) then Y[0] = Y[0] xor 0x87
```

5.1 EME2-AES Encryption

The EME2-AES encryption procedure can be described by the formula:

C = EME2-AES-Enc(*Key*, *T*, *P*),

where

Key is the 48 or 64 byte EME2-AES key
T is the value of the associated data, of arbitrary byte length (zero or more bytes)
P is the plaintext, of length 16 bytes or more
C is the ciphertext resulting from the operation, of the same byte-length as *P*

The input to the EME2-AES encryption routine is parsed as follows:

- The key is partitioned into three fields, Key = Key_1 | Key_2 | Key_3, with Key_3 consisting of the last 16 bytes, Key_2 consisting of the 16 bytes before them, and Key_1 consisting of the remaining first 16 or 32 bytes.

- If not empty, the associated data is partitioned into a sequence of blocks T = T_1 | T_2 | … | T_r, where each of the blocks T_1, T_2, …, T_{r-1} is of length exactly 16 bytes, and T_r is of length between 1 and 16 bytes.

- The plaintext P is partitioned into a sequence of blocks P = P_1 | P_2 | … | P_m, where each of the blocks P_1, P_2, …, P_{m-1} is of length exactly 16 bytes, and P_m is of length between 1 and 16 bytes.

The ciphertext shall then be computed by the sequence of steps in Table 1 or equivalent. An illustration of these steps (for plaintext of 130 full blocks and one partial block) is provided in Figure 1.

Figure 1. An illustration of EME2-AES

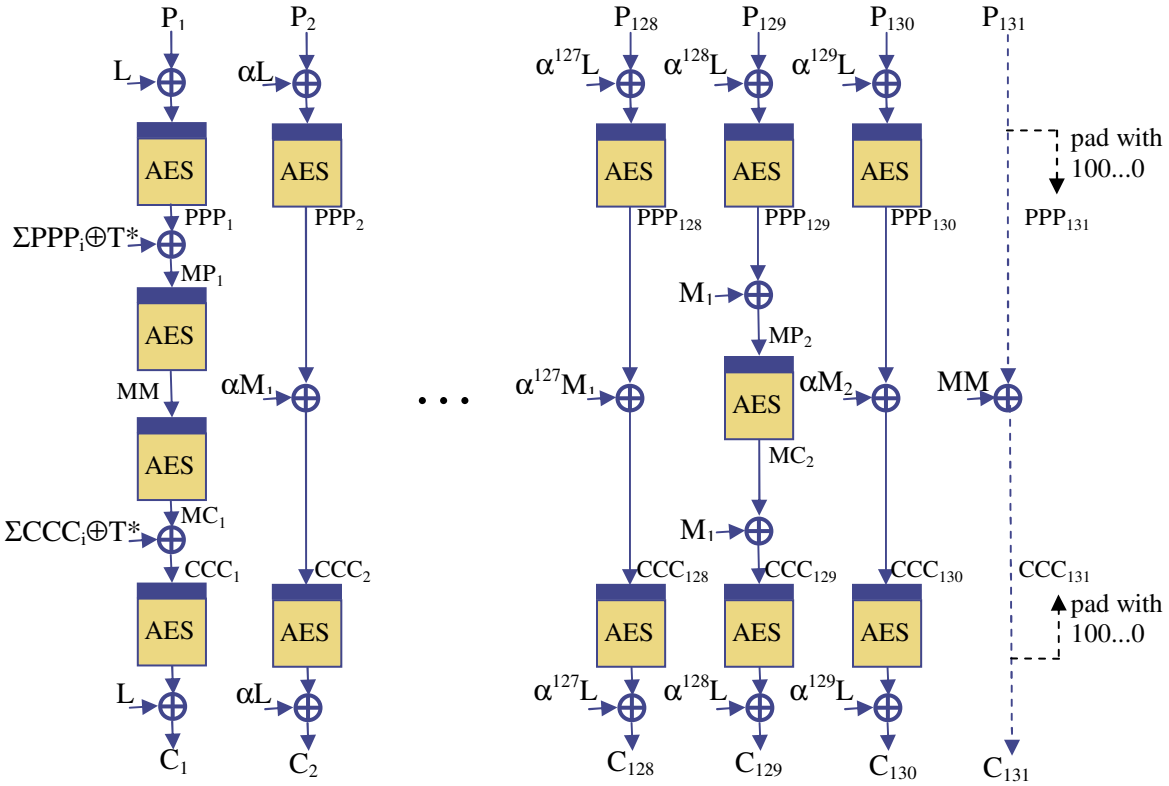
P1
P2
L
αL
AES
AES
PPP1
PPP2
ΣPPPi⊕T*
MP1
AES
MM
αM1
AES
MC1
ΣCCCi⊕T*
CCC1
CCC2
AES
AES
L
αL
C1
C2
P128
P129
P130
P131
α127L
α128L
α129L
AES
AES
AES
PPP128
PPP129
PPP130
pad with 100...0
PPP131
M1
MP2
α127M1
AES
αM2
MM
MC2
M1
CCC128
CCC129
CCC130
CCC131
pad with 100...0
AES
AES
AES
α127L
α128L
α129L
C128
C129
C130
C131

Table 1. The EME2-AES Encryption Procedure

```
// Process the associated data T to get the 16-byte block T_star
1.  if len(T)==0 then T_star = AES-Enc(Key1, Key3)
2.  else
3.      Key3 = Mult-by-alpha(Key3)
4.      for i=1 to r-1
5.          TTi = AES-Enc(Key1, Key3⊕Ti) ⊕ Key3
6.          Key3 = Mult-by-alpha(Key3)
7.      If len(Tr)<16 then
8.          Tr = Tr | 10…0    // pad Tr to 16 bytes, a 1 byte followed by 0's
9.          Key3 = Mult-by-alpha(Key3)
10.     TTr = AES-Enc(Key1, Key3⊕Tr) ⊕ Key3
11.     T_star = TT1 ⊕ TT2 ⊕ … ⊕ TTr

// First ECB pass
12. L = Key2
13. for i=1 to m-1
14.     PPPi = AES-Enc(Key1, L⊕Pi)
15.     L = Mult-by-alpha(L)
16. if len(Pm)<16 then  PPPm = Pm | 10…0 // pad Pm to 16 bytes
17. else                PPPm = AES-Enc(Key1, L⊕Pm)

// Intermeidate mixing
18. MP = PPP1 ⊕ PPP2 ⊕ … ⊕ PPPm ⊕ T_star
19. if len(Pm)<16 then
20.     MM = AES-Enc(Key1, MP)
21.     MC = MC1 = AES-Enc(Key1, MM)
22. else MC = MC1 = AES-Enc(Key1, MP)
23. M = M1 = MP ⊕ MC
24. for i=2 to m-1
25.     if (i-1 mod 128 > 0) then
26.         M = Mult-by-alpha(M)
27.         CCCi = PPPi ⊕ M
28.     else
29.         MP = PPPi ⊕ M1
30.         MC = AES-Enc(Key1, MP)
31.         M = MP ⊕ MC
32.         CCCi = MC ⊕ M1
33. if len(Pm)<16 then
34.     Cm = Pm ⊕ (MM truncated to len(Pm) bytes)
35.     CCCm = Cm | 10…0 // pad Cm to 16 bytes
36. else if (m-1 mod 128 > 0) then
37.     M = Mult-by-alpha(M)
38.     CCCm = PPPm ⊕ M
39. else CCCm = AES-Enc(Key1, M1⊕PPPm) ⊕ M1
40. CCC1 = MC1 ⊕ CCC2 ⊕ … ⊕ CCCm ⊕ T_star

// Second ECB Pass
41. L = Key2
42. for i=1 to m-1
43.     Ci = AES-Enc(Key1,CCCi) ⊕ L
44.     L = Mult-by-alpha(L)
45. if len(Pm)==16 then Cm = AES-Enc(Key1,CCCm) ⊕ L
```

5.2 EME2-AES Decryption

The EME2-AES decryption procedure can be described by the formula:

C = EME2-AES-Dec(*Key*, T, C),

where

Key is the 48 or 64 byte EME2-AES key
T is the value of the associated data, of arbitrary byte length (zero or more bytes)
C is the ciphertext, of length 16 bytes or more
P is the plaintext resulting from the operation, of the same byte-length as C

The input to the EME2-AES decryption routine is parsed as follows:

- The key is partitioned into three fields, Key = Key_1 | Key_2 | Key_3, with Key_3 consisting of the last 16 bytes, Key_2 consisting of the 16 bytes before them, and Key_1 consisting of the remaining first 16 or 32 bytes.
- If not empty, the associated data is partitioned into a sequence of blocks T = T_1 | T_2 | … | T_r, where each of the blocks T_1, T_2, …, T_{r-1} is of length exactly 16 bytes, and T_r is of length between 1 and 16 bytes.
- The ciphertext P is partitioned into a sequence of blocks C = C_1 | C_2 | … | C_m, where each of the blocks C_1, C_2, …, C_{m-1} is of length exactly 16 bytes, and C_m is of length between 1 and 16 bytes.

The ciphertext shall then be computed by the sequence of steps in Table 2 or equivalent. (Note that the only difference between the encryption and decryption routines is that all the AES-Enc operations within lines 12-45 are replaced by AES-Dec operations.

Table 2. The EME2-AES Decryption routine

```
// Process the associated data T to get the 16-byte block T_star
1.  if len(T)==0 then T_star = AES-Enc(Key1, Key3)
2.  else
3.      Key3 = Mult-by-alpha(Key3)
4.      for i=1 to r-1
5.          TTi = AES-Enc(Key1, Key3⊕Ti) ⊕ Key3
6.          Key3 = Mult-by-alpha(Key3)
7.      If len(Tr)<16 then
8.          Tr = Tr | 10…0    // pad Tr to 16 bytes, a 1 byte followed by 0's
9.          Key3 = Mult-by-alpha(Key3)
10.     TTr = AES-Enc(Key1, Key3⊕Tr) ⊕ Key3
11.     T_star = TT1 ⊕ TT2 ⊕ … ⊕ TTr

// First ECB pass
12. L = Key2
13. for i=1 to m-1
14.     CCCi = AES-Dec(Key1, L⊕Ci)
15.     L = Mult-by-alpha(L)
16. if len(Cm)<16 then  CCCm = Cm | 10…0 // pad Cm to 16 bytes
17. else                CCCm = AES-Dec(Key1, L⊕Cm)

// Intermeidate mixing
18. MC = CCC1 ⊕ CCC2 ⊕ … ⊕ CCCm ⊕ T_star
19. if len(Cm)<16 then
20.     MM = AES-Dec(Key1, MC)
21.     MP = MP1 = AES-Dec(Key1, MM)
```

```
else MP = MP1 = AES-Dec(Key1, MC)
M = M1 = MP ⊕ MC
for i=2 to m-1
    if (i-1 mod 128 > 0) then
        M = Mult-by-alpha(M)
        PPPi = CCCi ⊕ M
    else
        MC = CCCi ⊕ M1
        MP = AES-Dec(Key1, MC)
        M = MP ⊕ MC
        PPPi = MP ⊕ M1
if len(Cm)<16 then
    Pm = Cm ⊕ (MM truncated to len(Cm) bytes)
    PPPm = Pm | 10…0 // pad Pm to 16 bytes
else if (m-1 mod 128 > 0) then
    M = Mult-by-alpha(M)
    PPPm = CCCm ⊕ M
else PPPm = AES-Dec(Key1, M1⊕CCCm) ⊕ M1
PPP1 = MP1 ⊕ PPP2 ⊕ … ⊕ PPPm ⊕ T_star

// Second ECB Pass
L = Key2
for i=1 to m-1
    Pi = AES-Dec(Key1,PPPi) ⊕ L
    L = Mult-by-alpha(L)
if len(Cm)==16 then Pm = AES-Dec(Key1, PPPm) ⊕ L
```