```
    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
```

NOTE - Conceptually, the procedure is a left shift of each byte by one bit with carry propagating from one byte to the next. Also, if the 15th (last) byte shift results in a carry, a special value (hexadecimal 0x87) is xor'ed into the first byte. This value is derived from the modulus of the Galois Field (polynomial $x^{128}+x^7+x^2+x+1$).

5.2.2 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 procedure is parsed as follows:

— The key is partitioned into three fields, Key = KEY_{AD} | KEY_{ECB} | KEY_{AES} with KEY_{AD} (the associated data key) consisting of the first 16 bytes, KEY_{ECB} (the ECB pass key) consisting of the following 16 bytes and KEY_{AES} (the AES encrrytion/decryption key) consisting of the remaining 16 or 32 bytes.

— If not empty, the associated data T 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 2 and Table 2 or equivalent. An illustration of these steps (for plaintext of 130 full blocks and one partial block) is provided in Figure 2.

Brian Gladman 11/9/08 7:53 PM
Deleted: Key

Brian Gladman 11/9/08 8:49 AM
Deleted: 1

Brian Gladman 11/9/08 7:54 PM
Deleted: Key_2

Brian Gladman 11/9/08 7:55 PM
Deleted: Key

Brian Gladman 11/9/08 8:49 AM
Deleted: 3

Brian Gladman 11/9/08 7:53 PM
Deleted: Key_3

Brian Gladman 11/9/08 8:49 AM
Deleted: last

Brian Gladman 11/9/08 7:54 PM
Deleted: Key_2

Brian Gladman 11/9/08 8:50 AM
Deleted: before them

Brian Gladman 11/9/08 7:55 PM
Deleted: Key_1

Brian Gladman 11/9/08 8:52 AM
Deleted: first

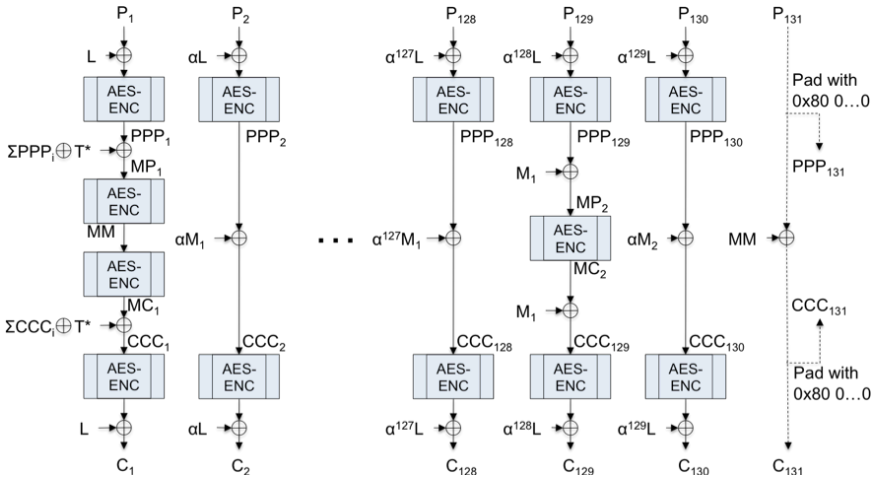

Figure 2, An illustration of EME2-AES: T* is computed from the associated data using the function H from Table 2, and the M_i's are computed as $M_i = MP_i$ xor MC_i

Table 1, The function H for processing the associated data T

```
// Function H( KEY_AES, KEY_AD, T = (T_1 ... T_r-1 T_r) ) :
1.  if len(T) == 0 then
2.    T_star = AES-Enc(KEY_AES, KEY_AD)
3.  else
4.    KEY_AD = Mult-by-alpha(KEY_AD)
5.    for j = 1 to r - 1
6.      TT_j = AES-Enc(KEY_AES, KEY_AD ⊕ T_j) ⊕ KEY_AD
7.      KEY_AD = Mult-by-alpha(KEY_AD)
8.    if len(T_r) < 16 then
9.      T_r = T_r | 0x80 0x00...      // pad T_r to 16 bytes, 0x80 followed by 0's
10.     KEY_AD = Mult-by-alpha(KEY_AD)
11.   TT_r = AES-Enc(KEY_AES, KEY_AD ⊕ T_r) ⊕ KEY_AD
12.   T_star = TT_1 ⊕ TT_2 ⊕ … ⊕ TT_r

13. return T_star                    // return the 16 byte value T_star
```

Table 2, The EME2-AES Encryption Procedure

```
// Function EME2-AES-Enc(KEY_AES, KEY_ECB, KEY_AD, T, P = (P_1 ... P_m-1 P_m) ):
1.  T_star = H(KEY_AES, KEY_AD, T)      // Process the associated data
2.  if len(P_m) = 16 then
3.    lastFull = m
4.  else
5.    lastFull = m - 1
6.    PPP_m = P_m | 0x80 0x00 ...      // Pad P_m to 16 bytes, 0x80 followed by 0's

//      First ECB pass
7.  L = KEY_ECB
8.  for j = 1 to lastFull
9.    PPP_j = AES-Enc(KEY_AES, L ⊕ P_j)
10.   L = Mult-by-alpha(L)

//      Intermediate mixing
11. MP = PPP_1 ⊕ PPP_2 ⊕ ... ⊕ PPP_m ⊕ T_star
12. if len(P_m) < 16 then
13.   MM = AES-Enc(KEY_AES, MP)
14.   MC = MC_1 = AES-Enc(KEY_AES, MM)
15. else
16.   MC = MC_1 = AES-Enc(KEY_AES, MP)
17. M = M_1 = MP ⊕ MC
18. for j = 2 to lastFull
19.   if (j - 1 mod 128 > 0) then      // use the current mask M
20.     M = Mult-by-alpha(M)
21.     CCC_j = PPP_j ⊕ M
22.   else                             // calculate a new mask M
23.     MP = PPP_j ⊕ M_1
24.     MC = AES-Enc(KEY_AES, MP)
25.     M = MP ⊕ MC
26.     CCC_j = MC ⊕ M_1
27. if lastFull < m then
28.   C_m = P_m ⊕ (MM truncated to len(P_m) bytes)
29.   CCC_m = C_m | 0x80 0x00 ...      // Pad C_m to 16 bytes, 0x80 followed by 0's
30. CCC_1 = MC_1 ⊕ CCC_2 ⊕ ... ⊕ CCC_m ⊕ T_star
```

Brian Gladman 11/9/08 7:55 PM
Deleted: **Key$_1$…Key$_3$** … [1]

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$…Key$_3$ … [2]

Brian Gladman 11/9/08 7:53 PM
Deleted: Key$_3$…Key$_3$ … [3]

Brian Gladman 11/9/08 8:07 PM
Deleted: i

Brian Gladman 11/9/08 8:07 PM
Deleted: $_i$…Key$_1$…Key$_3$…$_i$…Key$_3$ … [4]

Brian Gladman 11/9/08 7:53 PM
Deleted: Key$_3$…Key$_3$ … [5]

Brian Gladman 11/9/08 7:53 PM
Deleted: Key$_3$…Key$_3$ … [6]

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$…Key$_3$…Key$_3$ … [7]

Fabio Maino 11/7/08 10:25 PM
Formatted: Bullets and Numbering … [8]

Brian Gladman 11/9/08 8:12 PM
Deleted: ¶

Brian Gladman 11/9/08 7:55 PM
Deleted: **Key$_1$…Key$_2$…Key$_3$** … [9]

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$…Key$_3$… … [10]

Brian Gladman 11/9/08 7:54 PM
Deleted: Key$_2$

Brian Gladman 11/9/08 8:03 PM
Deleted: i

Brian Gladman 11/9/08 8:03 PM
Deleted: $_i$…Key$_1$…$_i$ … [11]

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$

Brian Gladman 11/9/08 8:04 PM
Deleted: i

Brian Gladman 11/9/08 8:04 PM
Deleted: i… ¶ … [12]

Brian Gladman 11/9/08 8:04 PM
Deleted: $_i$…$_i$ … [13]

Brian Gladman 11/9/08 8:05 PM
Deleted: $_i$

Brian Gladman 11/9/08 7:55 PM
Deleted: Key$_1$

Brian Gladman 11/9/08 8:05 PM
Deleted: $_i$

```
//      Second ECB Pass
31. L = KEY_ECB
32. for j = 1 to lastFull
33.   C_j = AES-Enc(KEY_AES, CCC_j) ⊕ L
34.   L = Mult-by-alpha(L)

35. return C = (C_1 ... C_m-1 C_m)
```

5.2.3 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 procedure is parsed as follows:

— The key is partitioned into three fields, Key = KEY_{AD} | KEY_{ECB} | KEY_{AES}, with KEY_{AD} (the associated data key) consisting of the first 16 bytes, KEY_{ECB} (the ECB pass key) consisting of the following 16 bytes, and KEY_{AES} (the AES encryption/decryption key) consisting of the remaining 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 plaintext shall then be computed by the sequence of steps in Table 3 or equivalent. (Note that the only difference between the encryption and decryption procedures is that all the AES-Enc calls in Table 3 are replaced in Table 4 by calls to AES-Dec. Also note that the function H is defined in Table 1).

Table 3, The EME2-AES Decryption procedure

```
// Function EME2-AES-Dec(KEY_AES, KEY_ECB, KEY_AD, T, C = (C_1 ... C_m-1 C_m) ):
```

Brian Gladman 11/9/08 8:06 PM
Deleted: i

Brian Gladman 11/9/08 8:06 PM
Deleted: i

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 8:06 PM
Deleted: i

Fabio Maino 11/7/08 10:33 PM
Formatted: Indent: Left: 0.25", No bullets or numbering

Brian Gladman 11/9/08 7:53 PM
Deleted: Key

Brian Gladman 11/9/08 8:51 AM
Deleted: 1

Brian Gladman 11/9/08 7:54 PM
Deleted: Key2

Brian Gladman 11/9/08 7:55 PM
Deleted: Key

Brian Gladman 11/9/08 8:51 AM
Deleted: 3

Brian Gladman 11/9/08 7:53 PM
Deleted: Key3

Brian Gladman 11/9/08 8:51 AM
Deleted: last

Brian Gladman 11/9/08 7:54 PM
Deleted: Key2

Brian Gladman 11/9/08 8:51 AM
Deleted: before them

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 8:52 AM
Deleted: first

Brian Gladman 11/9/08 7:55 PM
Deleted: **Key1**

Brian Gladman 11/9/08 7:54 PM
Deleted: **Key2**

Brian Gladman 11/9/08 7:53 PM
Deleted: **Key3**

```
1.  T star = H(KEY_AES, KEY_AD, T)    // Process the associated data
2.  if len(C_m) = 16 then
3.     lastFull = m
4.  else
5.     lastFull = m - 1
6.     CCC_m = C_m | 0x80 0x00 ...   // Pad C_m to 16 bytes, 0x80 followed by 0's

//  First ECB pass
7.  L = KEY_ECB
8.  for j = 1 to lastFull
9.     CCC_j = AES-Enc(KEY_AES, L ⊕ C_j)
10.    L = Mult-by-alpha(L)

//  Intermeidate mixing
11. MC = CCC_1 ⊕ CCC_2 ⊕ ... ⊕ CCC_m ⊕ T star
12. if len(C_m) < 16 then
13.    MM = AES-Dec(KEY_AES, MC)
14.    MP = MP_1 = AES-Dec(KEY_AES, MM)
15. else
16.    MP = MP_1 = AES-Dec(KEY_AES, MC)
17. M = M_1 = MP ⊕ MC
18. for j = 2 to lastFull
19.    if (j - 1 mod 128 > 0) then    // use the current mask M
20.       M = Mult-by-alpha(M)
21.       PPP_j = CCC_j ⊕ M
22.    else                           // calculate a new mask M
23.       MC = CCC_j ⊕ M_i
24.       MP = AES-Dec(KEY_AES, MC)
25.       M = MP ⊕ MC
26.       PPP_j = MP ⊕ M_i
27. if lastFull < m then
28.    P_m = C_m ⊕ (MM truncated to len(C_m) bytes)
29.    PPP_m = P_m | 0x80 0x00 ...     // Pad P_m to 16 bytes, 0x80 followed by 0's
30. PPP_1 = MP_1 ⊕ PPP_2 ⊕ ... ⊕ PPP_m ⊕ T star

//  Second ECB Pass
31. L = KEY_ECB
32. for j = 1 to lastFull
33.    P_j = AES-Dec(KEY_AES, PPP_j) ⊕ L
34.    L = Mult-by-alpha(L)

35. return P = (P_1 ... P_m-1 P_m)
```

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1…Key3… ... [14]

Brian Gladman 11/9/08 7:54 PM
Deleted: Key2

Brian Gladman 11/9/08 8:08 PM
Deleted: i

Brian Gladman 11/9/08 8:08 PM
Deleted: i…Key1…i ... [15]

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 8:09 PM
Deleted: i

Brian Gladman 11/9/08 8:09 PM
Deleted: i… ... [16]

Brian Gladman 11/9/08 8:10 PM
Deleted: i…i ... [17]

Brian Gladman 11/9/08 8:10 PM
Deleted: i

Brian Gladman 11/9/08 7:55 PM
Deleted: Key1

Brian Gladman 11/9/08 8:10 PM
Deleted: i

Brian Gladman 11/9/08 7:54 PM
Deleted: Key2

Brian Gladman 11/9/08 8:11 PM
Deleted: i

Brian Gladman 11/9/08 8:11 PM
Deleted: i…Key1…i ... [18]