1. The XCB-AES Transform

1.1 Data Units and Associated Data

This standard applies to encryption of a data stream divided into consecutive data units, where the data stream refers to the information that is to be encrypted and stored on the storage device. Information that is not to be encrypted is considered to be outside of the data stream. The data units in a data stream typically have the same size, but the XCB-AES transform defined in this standard does not require this uniformity of size.

Each data unit's size shall be at least 256 bits and at most 2^{39} bits. Each data unit is assigned an associated data value which is a non-negative integer. The associated data values are assigned consecutively, starting from an arbitrary non-negative integer.

The mapping between the data unit and the transfer, placement and composition of data on the storage device is beyond the scope of this standard. Devices compliant with this standard should include documentation describing this mapping. In particular, a single data unit does not necessarily correspond to a single logical block on the storage device. For example, several logical blocks might correspond to a single data unit. Data stream, as used in this standard, does not necessarily refer to all of the bits sent to be stored in the storage device. For example, if only part of a logical block is encrypted, only the encrypted bytes are viewed as the data stream, i.e. input to the encryption algorithm in this standard.

1.2 Notation

The two main functions used in XCB are AES block cipher encryption and multiplication over the field GF (2^{128}). The AES block cipher encryption of the value X with the key K is denoted as AES-enc(K, X), and the AES block cipher decryption is denoted as AES-dec(K, X). The multiplication of two elements X, Y $\in$ GF (2^{128}) is denoted as X $\cdot$ Y , and the addition of X and Y is denoted as X $\oplus$ Y. Addition in this field is equivalent to the bitwise exclusive-or operation, and the multiplication operation is defined in Section ???. We denote the number of bits in a bit string X as #X.

The function len(S) returns a 64-bit string containing the nonnegative integer describing the number of bits in its argument S , with the least significant bit on the right. The expression 0^n denotes a string of *n* zero bits, and A||B denotes the concatenation of two bit strings A and B. The function msb_t(S) returns the initial t bits of the string S . We consider bit strings to be indexed starting on the left, so that bit zero of S is the leftmost bit. When S is a bit string and 0 a <= b < #S , we denote as S [a; b] the length b - a substring of S consisting of bits a through b of S . The symbol {} denotes the bit string with zero length.

1.3 Definition

The XCB-AES encryption and decryption operations use the AES block cipher encryption functions **AES-enc** and **AES-dec**, as well as the hash function **h** and the pseudorandom function **c**. The variables H, K_e , K_d and K_c are derived from K, essentially by running the AES-enc encryption function in counter mode.

Optionally, these values can be stored between evaluations of these algorithms, in order to trade off some storage for a decreased computational load.

Let k be the size of the key fed to the AES function (either 128 or 256 bits). The function $\mathbf{c} : \{0, 1\}^k \times \{0, 1\}^{128} \rightarrow \{0, 1\}^l$, where the output length l is bounded by $0 <= 1 <= 2^{39}$, generates an arbitrary-length output by running the AES-enc function in counter mode, using its 128-bit input as the initial counter value. Its definition is

$$\mathbf{c}(K, W, l) = \text{AES-enc}(K, W) \| \text{AES-enc}(K, \text{incr}(W)) \| \ldots \mathbf{msb}_t (\text{AES-enc}(K, \text{incr}^{n-1}(W)),$$

where we make the output length l an explicit parameter for clarity; $n = \lceil l/128 \rceil$ is the number of 128-bit blocks in the output and $t = l \bmod 128$ is number of bits in the trailing block. Here the function $\text{incr} : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$ is the increment operation that is used to generate successive counter values. This function treats the rightmost 32 bits of its argument as a nonnegative integer with the least significant bit on the right, increments this value modulo 2^{32}. More formally,

$$\text{incr}(X) = X[0; 95] \| (X[96; 127] + 1 \bmod 2^{32}),$$

where we rely on the implicit conversion of bit strings to integers.

The functions h_1 and h_2 are defined in terms of the underlying hash function h as

$$h_1 (H, Z, B) = h(H, 0^{128}\|Z, B\|0^{(\#B \bmod 128) + 128})$$
$$h_2 (H, Z, B) = h(H, Z^{128}\|0^{128}, E\|0^{\#B \bmod 128}\|\text{len}(Z\|L)\|\text{len}(B))$$

The function $h : \{0, 1\}^{128} \times \{0, 1\}^a \times \{0, 1\}^c \rightarrow \{0, 1\}^{128}$, a $c \in [128, 2^{39}]$ is defined by $h(H, A, C) = X_{m+n+1}$, where the variables $X_i \in \{0, 1\}^{128}$ for $i = 0, \ldots, m+n+1$ are defined as

$$
\begin{aligned}
X_i &= 0 & &\text{for } i = 0 \\
X_i &= (X_{i-1} \oplus A_i) \cdot H & &\text{for } i = 1, \ldots, m-1 \\
X_i &= (X_{m-1} \ (A^*_m \| 0^{w-v})) \cdot H & &\text{for } i = m \\
X_i &= (X_{i-1} \ C_{i-m}) \cdot H & &\text{for } i = m+1, \ldots, m+n-1 \\
X_i &= (X_{m+n-1} \ (C^*_n \| 0^{w-u})) \cdot H & &\text{for } i = m+n \\
X_i &= (X_{m+n} \ (\text{len}(A) \| \text{len}(C))) \cdot H & &\text{for } i = m+n+1.
\end{aligned}
$$

Here we let A_i denote the 128-bit substring $A[128*(i-1); 128*i -1]$, and let C_i denote $C[128*(i-1)w; 128*i-i]$. In other words, A_i and C_i are the ith blocks of A and C, respectively, if those bit strings are decomposed into 128-bit blocks. This function is identical to the universal hash function that is used as a component of the Galois/Counter Mode (GCM) of Operation [SP800-38D]. (It is equivalent to the function used in Step 5 of Algorithm 4 of that specification, but please note that it is different than GHASH as defined in that document.)

1.4 Multiplication in GF(2^{128})

???Add here section for GF mult from .0 doc (Serge)???

1.5 The XCB-AES Encryption Operation

The XCB-AES encryption operation for an m-bit block P is modeled with this equation:

$$CT \leftarrow \text{XCB-AES-enc}(K,P,Z)$$

where:

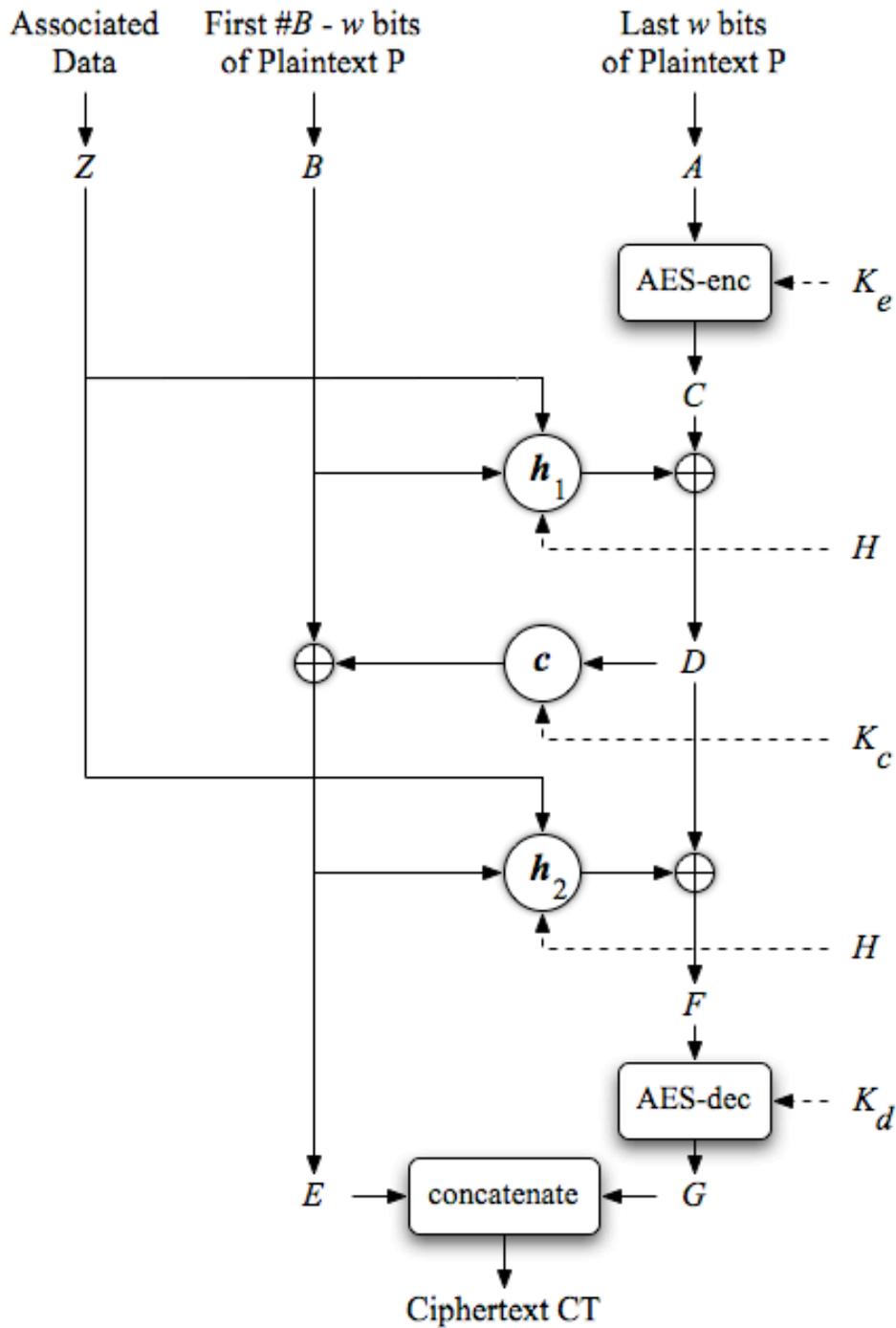K is either the 128 or 256 bit XCB-AES key

P is a block of plaintext of m bits where $m \in [128, 2^{32}]$

Z is the value of the associated data

CT is the block of 128 bits of ciphertext resulting from the operation

The ciphertext shall then be computed by the following or an equivalent sequence of steps (see Figure 1):

$$H \leftarrow \text{AES-enc}(K,0^{128})$$
$$K_e \leftarrow \text{msb}_k(\text{AES-enc}(K,0^{125}\|001)\| \text{AES-enc}(K,0^{125}\|010))$$
$$K_d \leftarrow \text{msb}_k(\text{AES-enc}(K,0^{125}\|011)\| \text{AES-enc}(K,0^{125}\|100))$$
$$K_c \leftarrow \text{msb}_k(\text{AES-enc}(K,0^{125}\|101)\| \text{AES-enc}(K,0^{125}\|110))$$
$$A \leftarrow P[m–128; m\text{-}1]$$
$$B \leftarrow P[0; m\text{-}127]$$
$$C \leftarrow \text{AES-enc}(K_e,A)$$
$$D \leftarrow C \oplus h_1(H,Z,B)$$
$$E \leftarrow B \oplus c(K_c,D,\#B)$$
$$F \leftarrow D \oplus h_2(H,Z,E)$$
$$G \leftarrow \text{AES-dec}(K_d,F)$$
$$CT \leftarrow E\|G$$

1.6 The XCB-AES Decryption Operation

The XCB-AES decryption operation for an m-bit block P is modeled with this equation:

$$P \leftarrow \text{XCB-AES-dec}(K, CT, Z)$$

where:

K is either the 128 or 256 bit XCB-AES key

CT is a block of ciphertext of m bits where $m \in [128, 2^{32}]$

Z is the value of the associated data

P is the block of 128 bits of plaintext resulting from the operation

The plaintext shall then be computed by the following or an equivalent sequence of steps:

$$H \leftarrow \text{AES-enc}(K, 0^{128})$$
$$K_e \leftarrow \text{msb}_k(\text{AES-enc}(K, 0^{125}\|001)\| \text{AES-enc}(K, 0^{125}\|010))$$
$$K_d \leftarrow \text{msb}_k(\text{AES-enc}(K, 0^{125}\|011)\| \text{AES-enc}(K, 0^{125}\|100))$$
$$K_c \leftarrow \text{msb}_k(\text{AES-enc}(K, 0^{125}\|101)\| \text{AES-enc}(K, 0^{125}\|110))$$
$$G \leftarrow P[m-128; m-1]$$
$$E \leftarrow P[0; m-127]$$
$$F \leftarrow \text{AES-enc}(K_d, A)$$
$$D \leftarrow F \oplus h_2(H, Z, E)$$
$$B \leftarrow E \oplus c(K_c, D, \#B)$$
$$C \leftarrow D \oplus h_1(H, Z, B)$$
$$A \leftarrow \text{AES-dec}(K_e, F)$$
$$P \leftarrow B\|A$$

1.7 Using XCB-AES-128 and XCB-AES-256 for Encryption of Storage

The encryption and decryption procedures described in 1.5 and 1.6 use AES as the basic building block with a key of either 128 or 256 bits. For completeness, the first mode shall be referred to as XCB-AES-128 and the second as XCB-AES-256. To be compliant with the standard, the implementation shall support at least one of the above modes.

Exporting and archiving XCB-AES keys can be done using the Key backup Structure defined in clause 7 of IEEE 1619. Key scope defines the range of data encrypted with a single XCB-AES key. As defined in clause 7.1.4 of IEEE 1619, the Key Scope is represented by three integers: the value of the particular associated data corresponding to the data unit in the sequence of data units encrypted by this key, the size in bits of each data unit, and the number of units to be encrypted/decrypted under the control of this key. An implementation compliant with this standard may or may not support multiple data unit sizes.

In an application of this standard to sector-level encryption of a disk, the data unit typically corresponds to a logical block, the key scope typically includes a range of consecutive logical blocks on the disk, and the associated data value corresponding to the first data unit in the scope typically corresponds to the Logical Block Address (LBA) associated with the logical block in the range.

An XCB-AES key shall not be associated with more than one key scope.

NOTE - The reason of the above restriction is that encrypting more than one block with the same key and the same index introduces security vulnerabilities that might potentially be used in an attack on the system. In particular, key reuse enables trivial cut-and-paste attacks.

Test Cases

XCB-AES-128 Test Case 1

K	000102030405060708090a0b0c0d0e0f
Generating H	
ECB plaintext	00000000000000000000000000000000
H	c6a13b37878f5b826f4f8162a1c8d879
Generating K_e	
ECB plaintext	00000000000000000000000000000001
K_e	7346139595c0b41e497bbde365f42d0a
Generating K_d	
ECB plaintext	00000000000000000000000000000003
K_d	b9ad2b2e346ac238505d365e9cb7fc56
Generating K_c	
ECB plaintext	00000000000000000000000000000005
K_c	9b82998964728141405e23dd9f1dd01b
Z	80000000000000000000000000000000
Plaintext	0000000000000000000000000000000000000000000000000000000000000000
A	00000000000000000000000000000000
B	00000000000000000000000000000000
C	cdbd38925be0ebd4eddb4aeabcd4ef6a
h1	f6896dfb0465b8a2c294137196bf3054
D	3b3455695f8553762f4f599b2a6bdf3e
ctr(D)	f727d748b86e3b362f20810eedbe378a
E	f727d748b86e3b362f20810eedbe378a
h2	c3f4009a02b283665c8695b634d5d4cd
F	f8c055f35d37d01073c9cc2d1ebe0bf3
G	2a25b360ea69e0c8a677b1890c09e33f
Ciphertext	f727d748b86e3b362f20810eedbe378a2a25b360ea69e0c8a677b1890c09e33f
h1 invocation	
first input	00000000000000000000000000000000800000000000000000000000000000000000000000000000
second input	00000000000000000000000000000000000000000000000000000000000000000000000000000000
X_1	00000000000000000000000000000000
X_2	c6a13b37878f5b826f4f8162a1c8d879
X_3	70a464963f89b8747f39dd0bf8afb53e
X_4	34fd7521b8d29c3b10e45dc5c0d4630d
X_5	f6896dfb0465b8a2c294137196bf3054
h2 invocation	
first input	80000000000000000000000000000000000000000000000000000000000000000000000000000000
second input	f727d748b86e3b362f20810eedbe378a00000000000001000000000000000100
X_1	c6a13b37878f5b826f4f8162a1c8d879
X_2	70a464963f89b8747f39dd0bf8afb53e
X_3	ef90605248c08274c4301690247347b1
X_4	748d68a77a9a906459d0d8ecb76cd98a
X_5	c3f4009a02b283665c8695b634d5d4cd