Format of portable serialization of org.owasp.esapi.crypto.CipherText object

This spreadsheet describes the portable serialization for the ESAPI for Java 2.0's CipherText object.
This serialization is intended to be platform and operating system independent.
It is expected that other ESAPI language implementations will implement this portable serialization
to make it possible to exchange data with encrypted data with other ESAPI implementations.

NOTE: All data is serialized as “network byte order”, which is the same as big-endian byte order.

String type: All strings are written out as UTF-8 encoded byte arrays in network byte order and
are prepended by a signed 2-octet length. Note that strings are *not* null terminated.

The values of the integral types are integers in the following ranges:
For byte, 1 octet, range from -128 to 127, inclusive
For short, 2 octets, range from -32768 to 32767, inclusive
For int, 4 octets, range from -2147483648 to 2147483647, inclusive
For long, 8 octets, range from -9223372036854775808 to 9223372036854775807, inclusive

Memory Layout:

Ordered as a byte array. All fields are written in network byte order.

Notation: Fields shown in *italics* are considered optional. If they are omitted, the length preceding that field will be 0.

Order	Size	Field	Detailed Description
1	long	version #	serialization version #; represented as int as YYYYMMDD
2	long	timestamp	time encrypted, in milliseconds since midnight, Jan 1, 1970 UTC
3	short	xformLen	strlen
4	xformLen octets	cipherXform	cipher transformation string, in form of cipherAlgorithm/cipherMode/paddingScheme; e.g., “AES/CBC/PKCS5Padding”
5	short	keysize	key size, in bits
6	short	blocksize	cipher block size (octets)
7	short	ivLen	IV length, in octets
8	ivLen octets	*IV*	Initialization vector, if ivLen > 0; otherwise omitted
9	int	ciphertextLen	length of raw cipher text, in octets
10	ciphertextLen octets	rawCiphertext	raw cipher text (for ciphertextLen octets)
11	short	macLen	length of MAC, in octets
12	macLen octets	*MAC*	Message Authentication Code (MAC) if macLen > 0; otherwise omitted

Calculation of MAC:

MAC is calculated by computing a derived key, via
SecretKey authKey = CryptoHelper.computeDerivedKey(key, keySize, "authenticity");
This authKey is then passed to CipherText.computeAndStoreMAC() for the CipherText object.

The MAC is calculated based on

MAC = HmacSHA1(authKey, IV + rawCipherText);

Where '+' denotes concatenation.