# 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 (i.e., big-endian) and
are prepended by a signed 2-octet length. Note that strings are *not* null terminated.

The values of the integral types in Java 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 (i.e., big endian).

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

| Order | Size (in octets) | Field | Detailed Description |
|---|---|---|---|
| 1 | 4 | KDF PRF & version # | Key Derivation Function (KDF) Pseudo Random Function (PRF) & serialization version #; represented as int as YYYYMMDD. See 'KDF PRF&/ vers' worksheet. |
| 2 | 8 | timestamp | |
| 3 | 2 | xformLen | strlen of cipherXform; always > 0. |
| 4 | xformLen octets | cipherXform | cipher transformation string, in form of cipherAlgorithm/cipherMode/paddingScheme; e.g., "AES/CBC/PKCS5Padding". |
| 5 | 2 | keysize | key size of cipher, in bits. |
| 6 | 2 | blocksize | cipher block size, given in octets. |
| 7 | 2 | ivLen | IV length, in octets; 0 if no IV present. |
| 8 | ivLen octets | *IV* | Initialization vector, if ivLen > 0; otherwise omitted. |
| 9 | 4 | ciphertextLen | length of raw cipher text, in octets. |
| 10 | ciphertextLen octets | rawCiphertext | raw cipher text (for ciphertextLen octets). |
| 11 | 2 | macLen | length of MAC, in octets; set to 0 if no MAC used. |
| 12 | macLen octets | *MAC* | Message Authentication Code (MAC) value if macLen > 0 (i.e., if MAC present); otherwise omitted. |

## Calculation of MAC:

MAC is calculated by computing a derived key using the Key Derivation Function (KDF) Pseudo Random Function (PRF), via
JavaEncryptor's private method, computeDerivedKey(), which in turn calls KeyDerivationFunction.computeDerivedKey()
with "authenticity" as the 'purpose' parameter.
This authKey is then passed to CipherText.computeAndStoreMAC() for the CipherText object.

The MAC (implemented in KeyDerivationFunction.computeDerivedKey()) is calculated based on NIST SP 800-108, section 5.1:

MAC = PRF( authKey, IV || rawCipherText );

Where '||' denotes concatenation, and PRF is a suitable pseudo random function.
The following PRFs are currently supported: HmacSHA1, HmacSHA256, HmacSHA384, & HmacSHA512.

Memory layout for use the first 4-byte 'int' (shown in network byte order [aka, big-endian]).

```
============================== Big-Endian Bit Ordering ====================================
|<------ Byte[0] ------>|<------ Byte[1] ------>|<------ Byte[2] ------>|<------ Byte[3] ------>|
------------------------------------------------------------------------------------------
|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31|32|
------------------------------------------------------------------------------------------
|                                                                       |Un|    MAC    |
|<---------------- Version Date Indicator in form of YYYYMMDD ---------------->|us| Algorithm |
|                                                                       |ed| Indicator |
------------------------------------------------------------------------------------------
```

Version is KDF version and MAC algorithm indicator specifies which PRF to use.

```
// Proposal for bits 29-32. Default set via Encryptor.KDF.PRF in ESAPI.properties file.
//        Built-in default uses HmacSHA256 if that property is not set.
//
// Allowed MAC algorithms and there respective key sizes.
//
// Value    MAC Alg name         hash size (bits)        Notes /Comments
// =====================================================================
    00       HmacSHA1                 160
    01       HmacSHA256               256                Default
    02       HmacSHA384               384
    03       HmacSHA512               512
    04       Reserved for SHA-3 winner  224              SHA-3 must provide
    05       Reserved for SHA-3 winner  256              msg digests of 224,
    06       Reserved for SHA-3 winner  384              256, 384, & 512 bits.
    07       Reserved for SHA-3 winner  512              Names for SHA-3 TBD.
    08       OtherReservedFuture01    ???                Thus leaving us room
    09       OtherReservedFuture02    ???                for 8 other MACs in
    ...      ...                      ...                the future.
    15       OtherReservedFuture08    8192         Uncle Albert's MACjik Elixir
```