Main page
Recent changes
Random page
Help
Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

C 0000008 mmmv salting algorithm 03

C 1..100

There is an array, **ar_p**, with **P** elements, which are pairs of fixed length arrays of **array_type_1**. The arrays of array_type_1 are ordinary arrays that contain only positive whole numbers. In this context zero, 0, is considered to be a positive whole number. The length of an array of array_type_1 is **ar_len_1**, $1 \leq ar\_len\_1$. The arrays in each pair are indexed so that one of the arrays has the index 0 and another one has the index 1. The index of pair elements is designated as **ix_01**. One of the arrays in the pair consists of data and another one consists of random numbers.

There is an array of one digit base 4 positive whole numbers that has the length of P and that array is designated as **ar_b4**. The ar_b4 consists of random numbers.

As the lengths of the ar_p and the ar_b4 are both equal to P, there can be a bijection *(archival_copy)* between the elements of those 2 arrays. The bijection is defined so that for each ar_p element at index ix_p, $0 \leq ix\_p \leq (P-1)$, the corresponding element at ar_b4 resides at ar_b4 index ix_p. As the ar_b4 elements have exactly one digit, the ar_b4 elements, which are positive whole numbers, can have exactly 4 possible values: 0,1,2,3. The meaning of those values is as follows:

ar_b4 element value	format of the ar_p element, a pair of arrays of array_type_1, that corresponds to the ar_b4 element
0	Data is at pair element 0 and in Little-Endian format.
1	Data is at pair element 0 and in Big-Endian format.
2	Data is at pair element 1 and in Little-Endian format.
3	Data is at pair element 1 and in Big-Endian format.

The endianness in the above table determins, how the whole numbers at arrays array_type_1 are ordered in terms of array indices. Byte endianness and bit endiannes are NOT defined in this specification.

Purpose

The main purpose of this algorithm is to allow encryption of relatively huge files by keeping the encrypted size of the file percentage wise not much bigger than 200% of the original file size. The scheme would be that the data arrays at the ar_p pairs are salted by XOR'ing them with some relatively short bytestream, **ar_salt_01**, that has some random length other than n*ar_len_1 to make the salt "shift" among different ar_p pairs, the ar_salt_01 and the ar_b4 are encrypted with some strength-first-speed-second encryption algorithm that may produce a ciphertext that is literally 100 times the size of its cleartext, the ar_p is encrypted with XOR-ing. The bigger the difference between ar_b4 length (the P) and ar_len_1, the closer the resultant ciphertext is to the 200% of the cleartext file size.