

Re: Cohen's paper on byte order

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2003-04/1374.html>

From: Douglas A. Gwyn (DAGwyn@null.net)

Date: 04/21/03

Date: Mon, 21 Apr 2003 01:14:37 -0400
From: "Douglas A. Gwyn" <DAGwyn@null.net>

Mok-Kong Shen wrote:

- > *'Where' in the document it is said that these are*
- > *'internal' values?*

What else could they be? FIPS-197 very carefully defines the 2-nibble byte notation specifically in the context of AES internal bytes (only), and also very carefully limits the external data model to a bit stream, with no reference to any other possible organization of the data externally.

- > *... So if 'any'*
- > *document provides test vectors, you would claim*
- > *that these are 'internal' entities, ...*

No, I'm talking specifically about FIPS-197. Many (most?) comparable standards do specify external data format within each octet. FIPS-197 very intentionally does not.

Note that FIPS-197 gives similar test vectors for the key schedule, which very clearly pertains only to internal values.

- >>*No, according to the FIPS the user has specified*
- >>*bit sequences (indexed 0,1,2,3,...), not byte*
- >>*values. There are no external bytes insofar as*
- >>*the FIPS is concerned. That is the whole problem!*
- > *Read Sec. 3.2 which says (brackets mine):*
- > *The basic unit for processing in the AES algorithm*
- > *is a byte, a sequence of eight bits treated as a*
- > *single entity. The input, output and Cipher Key bit*
- > *sequences described in Sec. 3.1 are processed as*
- > *arrays of bytes that are formed by dividing these*
- > *sequences into groups of eight contiguous bits to*
- > *form arrays of bytes (see Sec. 3.3).*

It also defines "byte" specifically as an internal

notion, which is appropriate for describing the computational algorithm. The I/O/key processing does involve groups of 8 bits, collected into an internal AES "byte", which the FIPS further describes. The FIPS-197 I/O process very specifically maps into/from internal byte values according to a scheme (Figure 2) that comes into contact with the external data representation *only* in the form of indexed bit sequence, and which therefore does not provide guidance about any multibit external object layout. This has been advertised previously as deliberate. My complaint is that it was inappropriate, since what we need in nearly all applications (excepting only inherently bit-stream systems) is to encrypt/decrypt arrays of octets with associated numeric values (for example, standard character codes).

> *As I pointed out in the previous post, the user provides a sequence of bytes (in notation of Oxpq*

How does he get them there? That notation is for use in the standard to describe internal operations, not an I/O spec. That is very clear from the wording in the actual standard.

You keep trying to "finesse" the whole issue by ignoring it. That doesn't solve anything.

> *... How he does that (with whatever magic) I don't care, for that's his job.*

Yes, that's my job. The way it is done affects whether my implementation will correctly communicate with other implementations. Apparently you agree that this aspect has not been specified. I want it specified.

> *... (The difference is only that the programmer on a big-endian machine has an easier job than his colleague on a little-endian machine who has to figure out how to achieve the same thing.)*

Wrong about that for something like the fifth time.

> *The test vectors are expressed as bytes as a user will provide and write down on a piece of paper.*

No they aren't; they're already published in the standard, and their primary purpose is so that implementors can verify that they're performing the computation correctly. This has nothing to do with the I/O interface.

- > *I simply don't care how the programmer is going*
- > *to do with the sequence, whether swaping something*
- > *or doing other weird stuff or not.*

Then you simply don't care about the whole issue being debated, and ought to shut up.

There has never been any argument about the operations being sufficiently well defined once one is "inside" the I/O layer.

- > *If, as analogy, in a numerical processing I give a*
- > *number 123 to be input to an integer variable j of*
- > *the program, do I have to tell the programmer how*
- > *to turn that 123 (as given on paper as characters)*
- > *into the binary in the computer??*

The characters 1,2,3 would only be on paper if it were a scanner/character recognition application. You should say that the characters are stored in some sequence in a file, and the analogous issue would be whether they should be stored in order 3,2,1 or 1,2,3 when the file is read in sequence using the usual byte-sequential access mechanisms for that environment. In this particular lousy analogy, one could probably get away without specifying that, since there is a cultural convention for an appropriate order (1,2,3) that "goes without saying". But for bit order in a byte there is certainly not a cultural consensus, and a comparable lack of specification cannot be compensated for by the programmer's cultural bias. Or rather, it can, but different programmers will make operationally different choices, which is undesirable when unanimity is a goal. (As it is for FIPS concerning data that is likely to be exchanged among disparate systems.)

- >>(The key schedule values don't normally appear in
- >>external data.)
- > *But the key 'exists' first on external media.*

The values of the key *schedule* given in the appendix most certainly do not normally appear on external media.

- > *Give solid counter-arguments to what I wrote above*
- > *and then we could better discuss this point.*

No, since you had stated clearly that you "simply don't care" about the only thing I was trying to

sci.crypt: Re: Cohen's paper on byte order

discuss, you have nothing useful to contribute to that discussion. And I'm not particularly interested in straightening out your problems in an area that should become clear to you if you were to draw a few appropriate diagrams and think about them.