So my idea is to focus on the code paths of the module/TOE.

So assuming,

1)      same underlying CPU type: x86, x86\_64, ARMv7, Aarch64, PPC, MIPS64,

2)      same operating system (and this matters at the “level” or “depth” at which the code is executing: so for native code, Windows vs. Linux.  At the java layer, the Java JVM, etc.

3)      that the code is “OpenSSL” like (in the sense that there’s no way to “call” a specific code path, and that once compiled, the executable code performs runtime checking of the CPU [CPUID] and then chooses its code path based upon instruction availability.

My approach would be for a vendor/lab to show that the CPUs used for CAVP testing is equivalent to others based upon the instruction sets offered.  And there might be different ways (with varying levels of effort) to show this:

a)      Assert that the code is ANSI C code, that compiles into the same machine instructions irrespective of the target CPU.  Given that NIAP/CC doesn’t look at source code any more, I suppose that a vendor would just assert this.

b)      CAVP test on a given CPU, then, show the CPUID for the CPU (e.g., “cat /proc/cpuinfo”), and then assert that any CPU with an identical set of instructions is equivalent.

c)       For a CPU that is different, a vendor could show (through source) that the instruction difference won’t matter to the given crypto algorithms (e.g., nx doesn’t change AES, but avx and aes instructions do).

d)      Finally, a vendor/lab could enumerate every different runtime code path, compile each of those code paths (so as to force use of it (either compiling different executables with a single code path each, or compiling in logic to force a given code path at runtime), and then test each different code path, thus asserting that the code will run correctly on any CPU (i.e., on any combination of instructions supported by the module/product).

A CAVP certificate would always need to list the explicit CPU model tested, so that anyone could check whether the code path executed during CAVP testing will be the same code path used on some other CPU.  However, the vendor could assert a list of equivalent CPUs through one of the above listed method (a-d).

”

I think it offers vendors/labs a menu of options, some easy equivalence, some more complex ones, and some hard ones that would appear to library maintainers where they are familiar with the code and with some thought, can benefit the community (RHEL OSSL example).

Ed