

Re: Release 1.1 (beta) of my AES implementation

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2003-06/1279.html>

From: Tom St Denis (*tomstdenis_at_iahu.ca*)

Date: 06/25/03

Date: Wed, 25 Jun 2003 16:14:55 GMT

Mok-Kong Shen wrote:

>
> *Tom St Denis wrote:*
>
>
>> *So what did you have in mind?*
>>
>> *#include "aes.c"*
>> *#include "sha1.c"*
>> *#include "...."*
>>
>> ???
>
>
> *Exactly. (Unfortunately.)*
>
>
>> *You're not going to get any users [re: bug finders] that way.*
>
>
> *I am not sure of that. In my past experience lots of*
> *people program that way.*

Lots of script kiddies maybe. From my experience the software developers at companies like Intel, Microsoft, HP, Sony and many universities seem to like it the way I do it.

Your approach lacks serious finesse and is a pain in larger scale projects. Heck even LTM [which is fairly small] takes about a minute to build on my 2Ghz processor. If I had to rebuild it each time I wanted to check a simple idea out I wouldn't get anything done.

Now imagine using your style of coding in a project 10 times bigger than LTM....

> *Or am I missing something?*

Yeah, post secondary education in computer science....

Re: Release 1.1 (beta) of my AES implementation

That being said... imagine this case

- one application
- splits into two threads both use AES
- both threads share memory space

This is common, specially for your beloved win32 platform [as a note of interest threads are not common in unix and are emulated in Linux]. Applications which may need to handle more than one connection [e.g. sockets] or handle bi-directional pipes, etc...

Sure, if you intend to write single thread applications only then your approach works fine.

My point though, is why limit yourself out of the gate? You should always try to write the best code you can [e.g. as portable as can be, as thread safe as can be, as flexible as can be, etc...].

If you cripple your code out of the gate then it can only really go downhill from there. And to be honest "good" coding style doesn't involve a lot of work when you are in the habit of doing it. Makefiles are simple to write, proper code factoring [e.g. no code in headers] is simple to do, writing thread safe routines is fairly trivial, etc...

You asked for comments on your code. I'm giving you the dime tour of what a high school grad knows in CS. Imagine if a real pro actually replied... whoa...

Tom