

# Re: JSH: Contradictory behavior, issue of math fraud

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2007-09/msg00527.html>

---

- *From:* David Bernier <[david250@xxxxxxxxxxxxx](mailto:david250@xxxxxxxxxxxxx)>
  - *Date:* Mon, 03 Sep 2007 23:14:36 -0400
- 

Tim Peters wrote:

[JSH]

But if the  
idea turns  
out to be a  
brilliant one  
which  
means  
factoring is  
not a hard  
problem  
after all,  
then how  
can  
mathematicians  
who not  
only  
couldn't  
figure it out,  
but who  
ignored it  
when  
presented  
with it be  
considered  
to be true  
experts  
in the field?

[rossum]

In its current version  
surrogate factoring is too  
slow to be  
considered "brilliant". Only

Re: JSH: Contradictory behavior, issue of math fraud

when you have speeded it  
up  
sufficiently

[JSH]

I asked, what if?

[Mas Plak]

Stick to reality, James, I know it can be hard for you, very hard,  
but accept the FACT that surrogate factoring is twice as slow as  
random guessing.

TWICE AS SLOW AS RANDOM GUESSING.

[rossum]

Not always.

Be a little careful here, lest Jimmy leap to (yet another :-() wrong conclusion. Because his "methods" always have strange convolutions, "a probe" in a JSH method is typically a lot slower than "a probe" in the random-gcd algorithm. For example, random-gcd kicks out candidates just as fast as random (or pseudo-random) numbers can be generated, while James usually has you /in addition/ putzing around with systematically generating all two-integer factorizations of some other number too.

Some of the iterations of James' method are better than random. For example, with  $k = 30$ ,  $n = 7, 8, 9 \dots$  and a suggestion by Tim Peters of using  $S = 27000 * (2*k^2 + n*T)$  then the results are:

Fermat average = 8.35 probes.  
JSH average = 608.86 probes.  
Probe ratio = 1 : 72.883  
Trial average = 118.63 probes.  
Reverse average = 12.70 probes.  
Random average = 745.43 probes.  
500 trials, 0 misfactors found.

Average n's tried per factorisation: 2.430  
Average k's tried per n: 1.000

This version is better than random. Tim's reasoning for his suggestion was that by ensuring a lot of small factors in S,  $27000 =$

Re: JSH: Contradictory behavior, issue of math fraud

$(2 * 3 * 5)^3$ , each S will generate a lot more factor pairs and so is more likely to hit a factor of T. The results show that with Tim's suggestion fewer values of n have to be tried before hitting a factor.

James will never understand this, but "Mas Plak" may: James guesses at method after method that /essentially/ feed pseudo-random integers into a gcd calculation. But viewed as pseudo-random generators, they're of poor quality, so an effective way to improve their "performance" has always been for me to treat them /as/ nothing more than PRNGs, and make simple changes that improve the coverage of the pseudo-random candidates generated. Since the number of integer factors of an integer (the surrogate S) depends on the /powers/ of the primes in S's prime factorization, artificially multiplying S by  $2^3 * 3^3 * 5^3$  first can greatly boost the number of candidate pairs generated, and without making S itself any harder to factor.

Taking this further, I tried  $k = 30$  with  $n = 12, 18, 24, 30, \dots$  and I got James' method down to about 580 probes.

Nevertheless, I bet it was still much slower (by clock time) than plain old random-gcd.

I downloaded the root certificates of Verisign from:

< <https://www.verisign.com/support/roots.html> >

It comes as no surprise to me that the signature parameters are SHA1 hash with a 2048-bit RSA key.

I also watched their "Life of a Threat" video, from

<

[http://www.verisign.com/Resources/Managed\\_Security\\_Services\\_Tours\\_&\\_Demos/security-threat-video.html](http://www.verisign.com/Resources/Managed_Security_Services_Tours_&_Demos/security-threat-video.html)

>

which is pretty good.

David Bernier

.