

Re: JSH: Nearly done

Source: <http://sci.tech-archive.net/Archive/sci.math/2005-01/7905.html>

From: Rick Decker (rdecker_at_hamilton.edu)

Date: 01/27/05

Date: Wed, 26 Jan 2005 20:08:51 -0500

jstevh@msn.com wrote:

> *Mark Nudelman wrote:*

>

>> *jstevh@msn.com wrote:*

>>

>>> *If any of you have an ounce of mathematical ability then meet me*

>

> *with*

>

>>> *mathematics.*

>>>

>>> *I dare you.*

>>

>> *I wish you _would_ talk about the mathematics more, rather than all*

>

> *the long*

>

>> *postings about how people are mistreating you.*

>>

>> *Specifically I would like you to answer two questions:*

>>

>> *1. Why is it easier to factor T rather than M? In another post I*

>

> *think you*

>

>> *said it's _not_ because it may have small factors, as Nora proposed.*

>

>

> *No.*

>

> $T = M^2 - j^2 = (M-j)(M+j)$

>

> *and j can be chosen such that M+j has any given prime you wish as a*

> *factor.*

>

Obviously.

> *So you can guarantee that the numbers you need to factor are smaller than M.*

>

Sure (look at $M - g$), but not necessarily that much smaller. At any rate, I don't see that a *full* factorization of T would be any easier than factoring M. Fortunately, you might not need to factor T fully.

> *My prototype program already does that to a certain extent, so you could put in some huge number into it, but it'd probably take it a few days to process through and more than likely, it wouldn't factor.*

>

> *But it would process through in a few days, even with an RSA challenge number.*

>

> *That's how fast it is.*

>

You're generalizing from a rather small data set. If, say, your algorithm had running time $O(M)$, then adding five extra digits to M might take 100000 times as long as it did before. (See my response to your "polynomial time" below.)

>

>>2. *What's the basis for your statement that the whole process will*

>

> *execute*

>

>>*in polynomial time?*

>>

>

>

> *I'm focusing on building a full method that relies only on my work, so it has to call itself recursively to factor, and such a method can potentially chew through even an RSA challenge type number in minutes.*

>

> *Then it's just a matter of iterating through the various combinations, which I've figured out are at about 150 million for the number generated by the first 1000 primes.*

>

> *Potentially this method can factor an RSA challenge sized number in seconds.*

>

> *I call that polynomial time.*

You might call it that, but that certainly isn't the standard usage. I know it's unlikely that you'll research what "polynomial time" algorithms really are, so I'll tell you: For a factoring algorithm to be said to run in polynomial time its running time has to be less than some polynomial in *the number of digits* of the input.

What we're *not* looking for is some factoring algorithm that runs, say, in $2M^2$ steps when given input M. Instead, what we

want is an algorithm that has running time $O((\log M)^k)$ for some fixed constant k . Testing trial divisors, for example, is polynomial in M , but we're looking for something much faster than that, namely polynomial in $\log M$.

>

> *Now, when I talk about development, I'm talking about using the method fully, so that it acts recursively.*

>

As a rule of thumb, recursive algorithms often take longer than the equivalent non-recursive versions. You should know this. That said, it doesn't hurt to use recursion in a "proof of concept" program.

> *NOW you can use some other method to factor T , like elliptic curve, and then use that factorization, as if you can get rational x 's, then you will have about a 50% probability per.*

>

> *I call that an inelegant solution and rely on my own algorithm for my research.*

>

> *However, if rational x 's can be consistently found, then the mathematics says that the method will more than likely factor a number without regard to size as long as you have T factored, as only a few rational x 's are needed with the 50% success rate per.*

>

You keep trumpeting this 50% success rate. Do you have either empirical or theoretical results to back this claim up? Not that I think it's wrong—I just haven't seen any evidence pro or con.

> *Now, all the math at the lower level has been worked out, while I haven't yet proven conclusively that you can get rational x 's at will, and even for very large numbers.*

You can **always** find rational solutions for x . That's not a deep result.

>

> *Maybe for really big numbers rational x 's become difficult to find.*

>

> *But at this point, I don't see a mathematical reason why they should.*

>

> *I think it would be quite useful if one of you could settle the question.*

>

Done. I'll give a proof if anyone's interested.

Regards,

Rick