

Re: Surrogate factoring works very well

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2007-03/msg00627.html>

- *From:* "Joseph Ashwood" <ashwood@xxxxxxx>
 - *Date:* Wed, 21 Mar 2007 04:29:37 GMT
-

<jstevh@xxxxxxxxxx> wrote in message
news:1174437391.159293.307780@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

After running some simple tests, it appears to fail on a wide range of numbers. For my testing purposes I made a few inconsequential changes. In Factor.java I changed the return type of main to int, and the println "Couldn't find factors" to return 0, before the catch I put a return 1, and after the catch return 0, this doesn't affect the factoring at all. I then created a wrapper class:

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.security.SecureRandom;

public class BigMain {

    /**
     * @param args
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        //build composite
        BigInteger prime1 = null;
        BigInteger prime2 = null;
        BigInteger composite = null;
        SecureRandom rand = new SecureRandom();
        int realBits = 0;
        FileOutputStream fout = new
        FileOutputStream("output"+System.currentTimeMillis()+".csv");
        for(int bits = 6; bits < 256; bits++)
        {
            prime1 = BigInteger.probablePrime(bits, rand);
            prime2 = BigInteger.probablePrime(bits, rand);
            composite = prime1.multiply(prime2);
            realBits = composite.bitCount();
            String [] whatever = new String[1];
```

Re: Surrogate factoring works very well

```
whatever[0] = composite.toString();
int ret = 0;
long time1 = System.currentTimeMillis();
ret = Factor.main(whatever);
long time2 = System.currentTimeMillis();

String bigout = "";

if(ret == 1)
bigout =
realBits+","+composite+","+prime1+","+prime2+","+time2-time1+",TRUE\n";
else
bigout =
realBits+","+composite+","+prime1+","+prime2+","+time2-time1+",FALSE\n";

fout.write(bigout.getBytes());

}

}

}
```

It's small enough to just put the entire text here. It of course generates a CSV file that can be easily viewed in most spreadsheets. Starting at 10 bits of actual input length it generated the first failure for 53762053 which should factor to 6599 and 8147. After that it solved one at 14 bits, and one at 15 bits, after that I stopped it at bits=60 (from the loop in BigMain) where it had a 67 bit long attempt that it failed. So it appears that the code does not work properly for large values.

Joe

.