MA 341: Advanced Problems for Fun: # 4

Crypto!

1. Now let's try a different factoring technique. Let $p = 13$ and $q = 31$. Let $B = 4$ be a so-called "smoothness" bound. Compute

$$M = \prod_{\ell \text{ prime } \leq B} \ell^{\lfloor \log_\ell(B) \rfloor}.$$

2. Continuing from the last step, we have $n = 13 \cdot 31 = 403$, but we'll pretend we don't know that factorization. Compute $d = \gcd(2^M - 1, n)$. (Note: you can compute $2^M - 1 \bmod n$ before computing the gcd – this may be easier than computing that larger number in practice!) How does this[1] help us factor n?

3. What happens if we tweak B a little? For example, if we set $B = 5$ instead, or $B = 3$? Why does this method work sometimes? As a hint, recall that $a^{p-1} \equiv 1 \bmod p$ when p is prime and $\gcd(a, p) = 1$.

4. Is it easier to factor n when $p - 1$ has lots of small prime factors, or just a few large prime factors using this $p - 1$ method?

The second technique illustrates just one pitfall to watch out for when implementing cryptosystems - not all primes are just as good for creating difficult numbers n to factor. In the crypto world, one may call Pollard's $p-1$ method an "attack" against RSA, whose security relies on the difficulty of factoring $n = pq$ when the numbers are large. In practice, the numbers used are a bit too large for this attack to succeed, assuming $p-1$ and $q-1$ don't have that many small factors, so designers of actual cryptosystems need to keep these sort of technical points in mind.

[1] This is called Pollard's $p - 1$ method.