$8 \cdot 13^{4 \cdot 8005} + 183$ is a probable prime

Curtis Bright

April 2, 2012

Abstract

This note (most likely) resolves an apparent open problem concerning if there is a prime of the form $8 \cdot 13^{4i} + 183$ for $i \geq 1$.

1 Overview

The 35670-digit number

$$p \coloneqq 8 \cdot 13^{4 \cdot 8005} + 183$$

is very likely prime, as determined by the Miller–Rabin primality test [6, 8].

To begin with, note that $p - 1$ has the partial factorization

$$p - 1 = 2 \cdot 5 \cdot 13 \cdot 509 \cdot C_{35665},$$

where C_{35665} is a 35665-digit composite number with no prime factors below 10^{12}. Since $p - 1$ contains a single factor of 2, the Miller–Rabin criterion to base a says that

$$a^{(p-1)/2} \equiv \pm 1 \pmod{p}$$

holds if p is prime. Furthermore, if p is not prime then at least 3/4 of the bases $a \in (\mathbb{Z}/p\mathbb{Z})^*$ do not satisfy this identity [7, 8].

The following congruences show that p satisfies the Miller–Rabin criterion for all prime bases under 40:

$$2^{(p-1)/2} \equiv 1 \pmod{p} \qquad 17^{(p-1)/2} \equiv 1 \pmod{p}$$
$$3^{(p-1)/2} \equiv 1 \pmod{p} \qquad 19^{(p-1)/2} \equiv 1 \pmod{p}$$
$$5^{(p-1)/2} \equiv 1 \pmod{p} \qquad 23^{(p-1)/2} \equiv -1 \pmod{p}$$
$$7^{(p-1)/2} \equiv -1 \pmod{p} \qquad 29^{(p-1)/2} \equiv -1 \pmod{p}$$
$$11^{(p-1)/2} \equiv -1 \pmod{p} \qquad 31^{(p-1)/2} \equiv 1 \pmod{p}$$
$$13^{(p-1)/2} \equiv 1 \pmod{p} \qquad 37^{(p-1)/2} \equiv -1 \pmod{p}$$

Already, this seems to strongly suggest that p is in fact prime. In practice the 3/4 bound is highly pessimistic and even a single base $a \neq \pm 1$ for which a number passes is good evidence of primality.

In fact, the probability that a randomly chosen k-bit integer passes for a randomly chosen base is less than $16k^2/4^{\sqrt{k}}$ [2]. For $k = \lceil \log_2 p \rceil = 118492$, this gives a miniscule probability of failure of about 10^{-196}. Of course, this theorem does not actually help us in this context since p was certainly not chosen randomly.

However, in addition p passed over 1000 tests to pseudo-randomly chosen bases. If p were not prime then we would expect the chance of this happening to be less than $(1/4)^{1000} \approx 10^{-604}$. The computations were performed using the GMP library [3], which uses the Mersenne twister [5] pseudo-random number generator.

2 Sample code

Figure 1 contains C code which employs the GMP library to compute $a^{(p-1)/2} \bmod p$. The base a is given as a single command-line argument, and the result is printed on the standard output stream. A single computation takes about 6 minutes to run on an Intel Core i3-540 processor.

3 Future work

Although p is almost certainly prime, the natural next question to ask is if we can formally prove that it is prime. Although much larger primes are known, they are all of a special form amenable to primality testing, and the form of p does not seem to be especially helpful in this regard.

The most promising method for proving its primality seems to be by utilizing elliptic curve primality proving [4]. This method generally works well in practice, although it has not been proven to run in polynomial time for all inputs. Currently the largest number proved prime using ECPP contains 26642 digits [1], so it is likely a formal proof of primality for p is still some years away.

References

[1] Chris Caldwell. The prime pages. `http://primes.utm.edu/top20/page.php?id=27`.

[2] Ivan Damgård, Peter Landrock, and Carl Pomerance. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, 61(203):pp. 177–194, 1993.

[3] Torbjörn Granlund et al. The GNU multiple precision arithmetic library 5.0.4, February 2012. `http://gmplib.org/`.

[4] J. Franke, T. Kleinjung, F. Morain, and T. Wirth. Proving the primality of very large numbers with fastECPP. In Duncan Buell, editor, *Algorithmic*

```
#include <gmp.h>

int main(int argc, char** argv)
{    mpz_t p, d, a;

     // compute d = (p-1)/2
     mpz_init(d);
     mpz_ui_pow_ui(d, 13, 4*8005);     // d = 13^(4*8005)
     mpz_mul_ui(d, d, 4);              // d = 4*d
     mpz_add_ui(d, d, 91);             // d = d + 91

     // compute p
     mpz_init_set(p, d);               // p = d
     mpz_mul_ui(p, p, 2);              // p = 2*p
     mpz_add_ui(p, p, 1);              // p = p + 1

     // compute a^((p-1)/2) mod p
     mpz_init_set_str(a, argv[1], 10); // a = command-line input
     gmp_printf("Computing %Zd^((p-1)/2) mod p...\n", a);
     mpz_powm(a, a, d, p);             // a = a^d mod p

     // nicely format -1 mod p
     mpz_add_ui(a, a, 1);              // a = a + 1
     mpz_mod(a, a, p);                 // a = a mod p
     mpz_sub_ui(a, a, 1);              // a = a - 1
     gmp_printf("Result: %Zd\n", a);

     // clean-up
     mpz_clear(a);
     mpz_clear(d);
     mpz_clear(p);

     return 0;
}
```

Figure 1: Code the for computation of $a^{(p-1)/2} \bmod p$.

Number Theory, volume 3076 of *Lecture Notes in Computer Science*, pages 194–207. Springer Berlin / Heidelberg, 2004.

[5] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[6] Gary L. Miller. Riemann's hypothesis and tests for primality. In *Proceedings of seventh annual ACM symposium on Theory of computing*, STOC '75, pages 234–239, New York, NY, USA, 1975. ACM.

[7] Louis Monier. Evaluation and comparison of two efficient probabilistic primality testing algorithms. *Theoretical Computer Science*, 12(1):97 – 108, 1980.

[8] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.