Primality Certification with Elliptic Curves

Ben Black

December 11, 2019

1 Primality Testing and Certification

In modern day cryptography, the Miller-Rabin probabilistic primality test is universally used for checking primality of general integers. It is much faster both practically and algorithmically than any deterministic primality test, for any reasonable bound on certainty. In fact, if we set our reasonable bound of certainty to that of a hardware failure giving us an incorrect answer, then it is still much, much faster. Because the chance of hardware failure increases with increased computation time and memory usage, it can be argued that Miller-Rabin can actually give a more certain result than some deterministic tests.

So how can general purpose deterministic primality tests prove their worth in this imperfect world? The way I will discuss in this paper is the ability to generate short, easily checkable proofs of primality, which can be verified much faster using much simpler code than the generator of the proof. Even if this process of finding the proof is fairly slow, then being able to check them may still have potential uses.

- If you are using an internet service which generates primes, and you don't like the certainty guarantees of checksums to validate that there were no errors in the transfer, then you could send both the prime, and the proof of the prime, and verify the proof on on site.
- If you wanted to store large databases of provable primes and didn't trust that the storage hardware would keep them correctly, you could also store the proof of the prime, and just check it whenever you need to use the prime.

The best algorithm for finding such a proof of primality is ECPP, the elliptic curve primality test discovered by Atkin and Morain. Although far slower than Miller-Rabin, it is fast enough to operate on cryptographically important primes in a few seconds. Unfortunately, ECPP is quite complicated and well beyond the scope of this paper. However, the condition one needs to check the primality certificate it generates is quite simple, just based on clever analysis of orders in elliptic curves. I will introduce the theory of an elementary version of the certificate the Pocklington criterion, then the elliptic curve primality criterion,

show a simple algorithm to generate these checks for arbitrary primes, and then argue that these may be useful in real world applications.

2 Pocklington primality test

The elliptic curve primality condition is modeled after the Pocklington primality test. I will explain it here in order to give some extra intuition.

2.1 Inspiration

Fermat's little theorem states that for p prime, and a coprime to p,

$$a^{(p-1)} = 1 \pmod{p}$$

We wish to find a converse to this statement that gives some set of conditions that force primality. To start, we know that $(p-1)$ is the smallest power of a that equals 1. Also, for m composite, we know that $a^l = 1 \pmod{p}$ for some $l < p-1$. Specifically, we can use the fact that $(\mathbb{Z}/p\mathbb{Z})^*$ is cyclic to give the following statement.

Theorem 1 *If there exists an a coprime to n such that $a^{n-1} = 1 \ (mod\ n)$ but $a^{(n-1)/q} \neq 1 \ (mod\ n)$ for every prime $q|n-1$, then n is prime.*

We now have a condition for primality. However, there is the problem that we need to factor $n-1$, which is not generally computationally tractable. However, if we already knew factors, then we could easily and quickly verify the primality of n by the above condition. So the certificate of primality of n would simply be the factorization of $n-1$.

2.2 Pocklington Criterion

Pocklington's insight was that if there was a large prime factor q of $n-1$, then there are some conditions that allow us to show primality only based on the primality of q, and even if we don't know the other factors of $n-1$. Formally,

Theorem 2 *Let $n > 1$ be an integer, q a prime divisor of $n-1$, and a coprime to n. If*

$$q > \sqrt{n} - 1$$

and

$$\gcd(a^{(n-1)/q} - 1, n) = 1$$

Then n is prime

2.3 Proof of Pocklington Criterion

Suppose n is composite. Then there must be a $p \leq \sqrt{n}$ that divides n.

We are going to look at the order of $\mathbb{F}_p$ which is, $p-1$, to reach a contradiction.

We have

$$a^{n-1} = 1 \pmod{p}$$

Since q is prime, and $q > \sqrt{n} - 1 \geq p - 1$, q is coprime to $p-1$. Therefore, q is multiplicatively invertible in $(\mathbb{Z}/(p-1)\mathbb{Z})^*$, and so we can just multiply the exponent of a by the inverse of q

$$(a^{(n-1)})^{1/q} = a^{(n-1)/q} = 1 \pmod{p}$$

Which contradicts the condition $\gcd(a^{(n-1)/q} - 1, n) = 1$. So n is prime.

2.4 Proof Certificate using Pocklington's Criterion

Note that factoring $n-1$ is trivial if it factors into $2p$, where p is prime. Also note that we get the size condition of Pockington's criterion, $p > \sqrt{n} - 1$, for free because is p such a large factor of $n-1$.

Understanding that not all n are of this sort, we can still use Pockington's criterion to say that if $\gcd(a^{(n-1)/q} - 1, n) = \gcd(a^2 - 1, n) = 1$, and that p is prime then n is prime. Of course you need to check if p is prime, but hopefully you can use the same method to check that. And so on until you create a decreasing chain of primes

$$n > p_0 > p_1 > ... > p_{small}$$

that demonstrate the previous number's primality.

Example 1 *Creating a full proof of primality for* $n = 47$*,*

n	$q = (n-1)/2$	$\gcd(a^2 - 1, n)$
47	23	1
23	11	1
11	5	1
5	2	1

2 is clearly prime, so each n is in turn prime, and so 47 is prime.

Even though this method does not work for general primes, we will make a similar method work using elliptic curves. The key will be that although there is only one $(\mathbb{Z}/n\mathbb{Z})^*$, there are many elliptic curves over $\mathbb{Z}/n\mathbb{Z}$, and some of them will have order $2p$, where p is prime.

3 Elliptic Curves Overview

3.1 Introductory Sources

There are many fabulous introductions to the basics of elliptic curves, including

- The fabulous explanation by Andrea Corbellini. It includes interactive components and a thorough overview that does not require much background in algebra. Posted here:

  `http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduc`
  `#elliptic-curves`

- Joseph H. Silverman John T. Tate Rational Points on Elliptic Curves

As these are better written than I could ever manage, I not attempt to introduce elliptic curves in a comprehensive manner, especially since most of the theory is unnecessary for our purpose. Instead, I will just cover the important topics such as order of elliptic curves.

3.2 Refresher

Recall that elliptic curves are defined in a plane of $\mathbb{F}^2$, where $\mathbb{F}$ is some field, as formulas

$$y^2 = x^3 + ax + b \text{ where } a, b \in \mathbb{F}$$

There is a group operation $+$, with identity $\mathbb{O}$ which is an additional point on the elliptic curve that is not on the plane. It is often referred to as the point at infinity. You can add points (x_1, y_1) and (x_2, y_2) on elliptic curves by the formulas

$$m = \left\{ \begin{array}{ll} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } x_1 = x_2 \end{array} \right\}$$

$$x_r = (m^2 - x_1 - x_2)$$

$$y_r = -(y_1 + m(x_r - x_1))$$

And (x_r, y_r) is the result.

Finally, multiplication of a point by an integer m is repeated addition of the point by itself m times.

3.3 Calculating Elliptic Curve Arithmetic

Calculating these formulas is difficult and error prone to work out by hand, and is not pleasant even using an ordinary calculator. I wrote a python file to support this paper, and in it is an `add` function and a `mul` function to help calculate addition and multiplication of elliptic curves.

It can be found here: `https://github.com/weepingwillowben/num_proj/blob/master/prime_check.py`. It uses the gmpy2 math library for modular arithmetic, and so unfortunately, it is non–trivial to install. Installation instructions are at the top of the file. All elliptic curve examples shown were made using this code to calculate operations.

3.4 Note about $E(\mathbb{Z}/n\mathbb{Z})$

Now that we are getting into the main subject of the paper, I have to mention a problem with elliptic curves in these sorts of proofs. That is, elliptic curves are only well defined over fields, and $\mathbb{Z}/n\mathbb{Z}$ is not a field unless n is prime. However, it turns out to not be too much of a problem, as operations work as you might expect them to, if they are well defined.

In particular, let $L, M \in E(\mathbb{Z}/n\mathbb{Z})$. Define $L_p = (x_p, y_p) \in E(\mathbb{F}_p)$, and $\mathbb{O}_p = \mathbb{O} \in E(\mathbb{F}_p)$. Then if $L + M$ is defined, then $(L + M)_p = L_p + M_p$.

So how do we check if the group operation is well defined? Recall the formula for the slope of the line

$$m = \left\{ \begin{array}{ll} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } x_1 = x_2 \end{array} \right\}$$

This operation fails in $\mathbb{Z}/n\mathbb{Z}$ if the denominator of the respective formula is not invertible $\pmod{n}$. This occurs exactly when

$$\left\{ \begin{array}{ll} \gcd(x_2 - x_1, n), & \text{if } x_1 \neq x_2 \\ \gcd(2y_1, n), & \text{if } x_1 = x_2 \end{array} \right\} \neq 1$$

It turns out that $\mathbb{Z}/n\mathbb{Z}$ is near enough to a field that this is the only way in which the elliptic curve assumptions fail, so this is an if and only if.

In the context of factoring and primality proving, the failure of addition reveals that n is not prime, and in fact, gives a factor of n, and so we don't have to continue to try to work in $\mathbb{Z}/n\mathbb{Z}$ anymore. So the failure condition for addition in $E(\mathbb{Z}/n\mathbb{Z})$ actually helps our work, instead of limiting us.

4 Elliptic Primality Testing

4.1 Approach

For the Pocklington criterion, we examined the order of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$ in order to prove primality. In $(\mathbb{Z}/n\mathbb{Z})^*$, the order is $(n-1)$. So we look at the order of elliptic curves in order to find a similar result.

4.2 Orders of Elliptic Curves

4.2.1 Theory

The order of a point on an elliptic curve, denoted $\#L$, is the number of times you can add a point $L \in E_{A,B}(\mathbb{F}_p)$ to itself before getting back the identity $\mathbb{O}$. Formally, $\#L$ is the smallest m such that $mL = \mathbb{O}$. Note that this implies that if you multiply L by $m+1$, you get back L, and $(m+2)L = 2L$, and so on, and so you can think of the multiplicand operating in $\mathbb{Z}/m\mathbb{Z}$. This is a special case of the fundamental theorem of finite abelian groups. Alternatively, $mL = \mathbb{O}$ can be though of as Fermat's little theorem for elliptic curves. Order analysis on points forms the basis for the primality test, mirroring Pocklington's criterion.

Order on the curve itself is defined very differently. For $E_{A,B}(\mathbb{F}_p)$, the order of an elliptic curve, denoted $\#E_{A,B}(\mathbb{F}_p)$ is the number of points in the elliptic curve. For any $L \neq \mathbb{O} \in E_{A,B}(\mathbb{F}_p)$, $\#L$ divides $\#E_{A,B}(\mathbb{F}_p)$.

4.2.2 Algorithm

It turns out that you can compute $\#E_{A,B}(\mathbb{F}_p)$ by counting the number of points on the elliptic curve in $\mathbb{F}_p^2$, plus the point at infinity. For a small p, one can count the points by just looking at the formula

$$y = \pm\sqrt{x^3 + ax + b}$$

And checking counting how many times y is an integer for all values $x \in \mathbb{F}_p$.

Example 2 *Counting points in $E_{1,1}(\mathbb{F}_5)$,*

We have $x = 0, 1, 2, 3, 4$, and so $x^3 + x + 1 = 1, 3, 1, 1, 4$, respectively.
The numbers which are quadratic residues, i.e. have a non-zero square roots in $\mathbb{F}_5$ are 1 and 4. So we have 4 values of $\pm y$, each of which non-zero, and so are distinct. So we have 8 points in total, 9 including $\mathbb{O}$.
So $\#E_{1,1}(\mathbb{F}_5) = 9$.

Lets check the above theorem that the order is exactly the number of points on the curve.

Example 3 *Checking that point order divides number of points in $E_{1,1}(\mathbb{F}_5)$,*

Take $(0, 1) \in E_{1,1}(\mathbb{F}_5)$. Then perhaps using the point addition calculator here: `https://cdn.rawgit.com/andreacorbellini/ecc/920b29a/interactive/modk-add.html`, we can compute that
$(0, 1) + (0, 1) = (4, 2)$
$(4, 2) + (0, 1) = (2, 1)$
$(2, 1) + (0, 1) = (3, 4)$
$(3, 4) + (0, 1) = (3, 1)$
$(3, 1) + (0, 1) = (2, 4)$
$(2, 4) + (0, 1) = (4, 3)$
$(4, 3) + (0, 1) = (0, 4)$
$(0, 4) + (0, 1) = \mathbb{O}$
Which is equal to $9 \times (0, 1)$, so we confirmed that $\#(0, 1) = 9 \mid 9$, as desired.

This counting algorithm is impractical if p is large. Luckily, we have polynomial time algorithms for this, Schoof's algorithm, and it's variants and improvements. So this is a tractable problem. However, Schoof's algorithm is extremely complicated, as well as rather slow, so I will not cover it here.

4.3 Bounds on Orders of Elliptic Curves

Although counting precisely how many points are on a particular curve is a challenge, Hasse discovered useful bounds for it.

In particular, for N the number of points on an elliptic curve over $\mathbb{F}_q$,

$$|N - (q+1)| \leq 2\sqrt{q}$$

We will use this in order to derive a contradiction that will prove the primality of N.

4.4 Elliptic Primality Criterion

We are going to use this order exactly like we used $n-1$ in the $(\mathbb{Z}/p\mathbb{Z})^*$ based primality proof, and establish a nearly identical criterion.

Theorem 3 *Let n be an integer not divisible by 2 or 3. Let $A, B \in \mathbb{Z}/n\mathbb{Z}$ such that $\gcd(4A^3 + 27B^2, n) = 1$, and let $L \neq \mathbb{O} \in E_{A,B}(\mathbb{Z}/n\mathbb{Z})$. If there exists a prime $q > (\sqrt[4]{n} + 1)^2$ such that $qL = \mathbb{O}$, then n is prime.*

Example 4 *Checking primality of* 103

Lets take $L = (85, 81) \in E_{A=72,B=93}(\mathbb{Z}/103\mathbb{Z})$. Then we can calculate that $53 \times L = \mathbb{O}$, and so by the theorem, 103 is prime if 53 is prime.

4.5 Proof

Proof by contradiction, assume some prime $p|n$. Mirroring the Pocklington criterion proof, we are going to look at the order of $L_p \in E_{A,B}(\mathbb{F}_p)$, which is the point L defined over $\mathbb{F}_p$ instead of $\mathbb{Z}/n\mathbb{Z}$.

1. $\gcd(4A^3 + 27B^2, n) = 1$ ensures that the elliptic curve is well defined over $E_{A,B}(\mathbb{F}_p)$, even though we don't know p, because if $p|4A^3 + 27B^2$, then $p|\gcd(4A^3 + 27B^2, n)$.

2. The n does not divide 2 or 3 is going to be necessary for a later theorem that we need to show that we can actually find such an elliptic curve in polynomial time. But it isn't necessary for the criterion.

Now the meat of the proof, using the notation from the $E(\mathbb{Z}/n\mathbb{Z})$ section of the paper.
$qL_p = (qL)_p$, by the elliptic curve behavior over $\mathbb{Z}/n\mathbb{Z}$.
$qL = \mathbb{O}$ by the criterion, and so $qL_p = \mathbb{O}_p = \mathbb{O}$, again by behavior over $\mathbb{Z}/n\mathbb{Z}$.
But since $L_p \neq \mathbb{O}$ and q is prime, the order of L_p must be q. After all, if it were a number t smaller than q then we could talk about qL in terms of $q \pmod t$, due to the fundamental theorem of albelian groups, where $0 \pmod t$ maps to

$\mathbb{O}$. But q is coprime to t, so q couldn't be 0, and so qL couldn't be $\mathbb{O}$ unless $t = 1 \implies L_p = \mathbb{O}$. But that is not true, by assumption. So indeed, the order of L_p must be q.
But by Hasse's bound on the order of elliptic curves,

$$\begin{aligned} \#L_p &\leq \#E_{A,B}(\mathbb{F}_p) && \text{Definition of order on } L_p \\ &\leq (\sqrt{p}+1)^2 && \text{Hasse's bound} \\ &\leq (\sqrt[4]{n}+1)^2 && \text{Condition on } p \\ &< q && \text{Condition on } q \end{aligned}$$

Which is a contradiction, so n must be prime.

4.6 Chains of Conditions

Like I showed earlier with the Pocklington Criterion, these conditional proof of primality based on a smaller number's primality can be chained together to create a complete certificate of primality.

Example 5 *Checking that point counting and order are equivalent in* $E_{1,1}(\mathbb{F}_5)$,

We can extend the example from before to also include the conditions that show that 53 is prime.

n	L	A	B	q	qL
103	(85,81)	72	93	53	$\mathbb{O}$
53	(4,30)	37	52	29	$\mathbb{O}$

Now, we can easily check that 103 is prime if 29 is prime using the criterion.

4.7 Implementation

Included in the my python code is a `check_prime` function that takes in a curve, point, assumed prime, and number to check and outputs whether it is prime using the method described above. If you are good with code, you should be able to play around with it and see that it works exactly as described above. There is also a `check_certificate` that checks a list of such information.

5 Generating the Primality Proof

The ability to check this condition would not be useful in a practical setting unless we had the ability to generate the proof conditions for general numbers in polynomial time. There are two main ways of finding the curves and points which meet the criterion. The first one, the Goldwasser–Kilian algorithm, is much simpler, and so that is what I will describe in depth. The second one is much faster and is what is used in practice, but also much more complicated.

5.1 Goldwasser-–Kilian Algorithm Overview

This algorithm is based on guess and check. Naively, We want to just guess a curve, a point L, and large prime q such that $qL = \mathbb{O}$. In this algorithm, we will still end up guessing the curve and the point, but instead of finding the prime by guessing, we can use the relationship between point counting and order and do fast point counting using Schoof's algorithm in order to find our candidate q.

Choose L, a random point on $\mathbb{Z}/n\mathbb{Z}$, and choose a random elliptic curve E over $\mathbb{Z}/n\mathbb{Z}$, such that L is in E. Count the number of points m. Check if m satisfies $2q$, with q a probable prime (checking with Miller-Rabin). Also check that $qL = \mathbb{O}$. If both hold, then you have found the point, curve, and prime number that satisfy the condition.

5.2 Implementation

To make this more concrete, I made a really inefficient version of this algorithm based on the description in An Overview of Elliptic Curve Primality Proving by Frank Li [1].

The `gen_and_check_certificate` function prints out the certificate and then checks it. It uses an exponential time point counting function instead of Schoof's algorithm, and an exponential time modular square root algorithm, so it unfortunately doesn't work on large primes, but hopefully it can be of use and maybe some fun.

5.3 Distribution of Orders of Random Elliptic Curves

At this point you might wonder how we know that we can find elliptic curves of the desired form. Perhaps we have the same problem as we had with Pockington's criterion, and we just described a special purpose primality proving algorithm. Or perhaps it takes too long to randomly find them.

It turns out that this really does work, though. The proof that there will always be an elliptic curve that we can find in expected polynomially many guesses is the following theorem due to Lenstra:

Theorem 4 *Let $p \neq 2, 3$ prime, and let $S \subseteq \{p+1-\lfloor\sqrt{p}\rfloor, .., p+1+\lfloor\sqrt{p}\rfloor\}$. Let A,B be chosen from a uniform random distribution over $\mathbb{F}_p$. Then there exists fixed constants k such that*

$$Prob(\#E_{A,B}(\mathbb{F}_p) \in S) > \frac{k}{log(p)} \frac{|S| - 2}{2\lfloor\sqrt{p}\rfloor + 1}$$

6 Real World Considerations

For all practical purposes, the Primo software by Marcel Martin is the best available elliptic curve primality prover implementation for general numbers.

Although the algorithm it uses to generate primality certificates is quite complicated, the certificate itself, and the method of checking it are both quite simple. It only requires a little more mathematical theory than in this paper, and only a couple hundred lines of code using a good math library like gmpy2. In fact, several people have actually created independent checks for Primo's certification files, although I have yet to find any that actually work on the most recent version of Primo. By pairing this independent checking with Primo's builtin verification, one should be able to rule out implementation errors or hardware failures with a very high confidence level, easily on par with that of a thorough Miller–Rabin test.

These checks should also be extremely fast. Assuming that the Primo checker is not taking incorrect shortcuts when verifying the certificate, we should be able to evaluate it's performance checking the certificates as the actual speed of checking primality. On my machine, using all cores, generating the proof of primality for the prime $(10^{1999} + 7321)$ took 678s. But checking the certificate only took 0.13 seconds. For comparison, gmpy2's `is_prime` function, which uses Miller–Rabin, took 0.62 seconds for 6 runs of Miller–Rabin (the lowest end of reasonable confidence) and 1.69 seconds for 25 runs (the default). As gmp is well regarded as a high quality, fast library, and gmp's `mpz_probab_prime_p` (which gmpy2's `is_prime` uses) is actually used in some real world cryptographic applications, I think it fair to say that checking these elliptic curve primality certificates can be at least as fast, if not faster than a decent Miller-Rabin test.

So hopefully I have shown how deterministic primality testing may have some use in practical applications as well as explaining the basic theory enough to explore the topic with some more comfort.

References

[1] Li, Frank. "Elliptic Curves for Primality Proving." SpringerReference (n.d.): n. pag. Stanford University, 15 Dec. 2011. Web. `http://theory.stanford.edu/~dfreeman/cs259c-f11/finalpapers/primalityproving.pdf`.

[2] Goldwasser, Shafi, and Joe Kilian. "Primality Testing Using Elliptic Curves." *Journal of the ACM* 46.4 (1999): 450-72. Web. `https://pdfs.semanticscholar.org/997d/f4e2a661aed97b2ad782531aa2ce122cab4d.pdf`.

[3] Atkin, A., and F. Morain. "ELLIPTIC CURVES AND PRIMALITY PROVING." *Mathematics of Computation* (1993): 29-68. Web. `http://www.ams.org/journals/mcom/1993-61-203/S0025-5718-1993-1199989-X/S0025-5718-1993-1199989-X.pdf`.