Primality Tests

Francesco Vesigna

August 20, 2024

Abstract

In this brief document we will explore some of the most popular primality tests with a certain regard for proofs and implementation.

1 Introduction

It is necessary in Cryptography to test whether a number is a prime or not. This is useful to construct cryptographic protocols such as RSA. A very simple example of a primality test is to check divisors using $\sqrt{n}$ as an upper bound. Unfortunately, this algorithm is exponential-time so it can't be used. In this document we'll provide more advanced primality tests (mainly probabilistic). In the end we'll cover the best known method known ad ECPP based on the Goldwasser-Kilian algorithm.

2 Miller Rabin Primality Test

Although the Miller Rabin test never tests the primality of a number n, it checks if a number is not prime. If we repeat the procedure k times we can have smaller and smaller probability about n not begin a prime. This test takes a number n to check and a number k of iterations for the procedure. To prove the procedure later on we'll suppose that the input number n is odd and not a perfect power.

- We choose a random $c \in (\mathbb{Z}/n\mathbb{Z})^*$
- We check a property of c, if this property is not verfied n is not prime else n could be a prime.

If the second case is verified c is a witness that n isn't a prime. If we repeat the procedure k times it becomes very likely to find at least one witness c if n is a composite number.

2.1 Property

We can write n as $n = a2^b + 1$ where b is an odd number. If n is prime then at least one of this two sentences is verified:

- $c^a \equiv 1 \ (n)$
- $c^{a2^i} \equiv -1 \ (n)$ for some $i \in \{0, \cdots, b-1\}$

2.2 Proof

Since n is prime we can apply Fermat's little theorem so it holds $c^{a2^b} \equiv c^{n-1} \equiv 1 \ (n)$.If $c^a \equiv 1 \ (n)$ is verfied we are done otherwise $c^a \not\equiv 1 \ (n)$. Thus there is some $i \in \{0, \cdots, b-1\}$ with $c^{a2^i} \not\equiv 1 \ (n)$ and $c^{a2^{i+1}} \equiv 1 \ (n)$. If we say that $x = c^{a2^i}$ then n divides $x^2 - 1$. We can rewrite $x^2 - 1$ as $(x-1)(x+1)$. By Euclid's lemma n has to divide $(x-1)$ or $(x+1)$. We supposed that $c^{a2^i} \not\equiv 1 \ (n)$ so $n \mid (x+1)$, hence

$$x + 1 \equiv 0 \ (n)$$

$$c^{a2^i} \equiv -1 \ (n) \text{ for some } i \in \{0, \cdots, b-1\}$$

2.3 Examples

Let's suppose the input $n = 21$. Then $2 \in (\mathbb{Z}/21\mathbb{Z})^*$ since $gcd(2, 21) = 1$.

$$n = 5 \cdot 2^2 + 1$$

$$2^5 \equiv 11\ (21)$$

$$2^{10} \equiv 16\ (21)$$

The property is not verified so $c = 2$ is a witness that 21 isn't a prime.

Let's now suppose the input $n = 23$. Then $3 \in (\mathbb{Z}/23\mathbb{Z})^*$ since $gcd(3, 23) = 1$.

$$n = 11 \cdot 2 + 1$$

$$3^{11} \equiv 1\ (23)$$

The propery is verfied so 23 could be a prime. More iterations on other c will show the same result.

2.4 Lower-Bound on Probablity

Let's fix a composite number n (odd and not a perfect power), how many witnesses c of n begin composite are there?. We will prove that at least 50% are indeed witnesses. Actually, these witnesses are many more ($\geq 75\%$), the weaker statement will be easier to prove and will be enough for every case scenario though.

2.5 Proof

Let's consider the following set of equations

$$x^1 \equiv -1,\ \ x^2 \equiv -1,\ \ x^4 \equiv -1,\ \ x^8 \equiv -1,\ \ \cdots x^{2^{b-1}} \equiv -1$$

Let $x^{2^j} \equiv -1$ be the largest of these equations that has some solutions in $(\mathbb{Z}/n\mathbb{Z})^*$.
Let $G = \{c \in (\mathbb{Z}/n\mathbb{Z})^* \text{ s.t. } c^{a2^j} \equiv \pm 1\}$. We now need to show three facts.

(i) The first is that if c is not a witness for compositeness of n, then $c \in G($ so $G \neq \emptyset$ or we are done). If c is not a witness, then $c^a \equiv 1\ (n)$ or $c^{a2^i} \equiv -1\ (n)$ for some $i \in \{0, \cdots, b-1\}$. If $c^a \equiv 1\ (n)$ then $(c^a)^{2^j} = 1$ in $(\mathbb{Z}/n\mathbb{Z})^*$. If $c^{a2^i} \equiv -1\ (n)$ then $(c^a)2^i \equiv -1\ (n)$, but $i \leq j$ since we defined j in such way. We can now rewrite the equation in this way $(c^{a2^i})^{2^{j-i}} \equiv \pm 1\ (n)$.

(ii) The second is that G must be a subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$. Let's take two elements $g, h \in G$ so we have that $g^{a2^j} \equiv \pm 1$ and $h^{a2^j} \equiv \pm 1$. We note that $(gh)^{a2^j} \equiv g^{a2^j} h^{a2^j} \equiv \pm 1$ so $gh \in G$. Moreover $(c^{-1})^{a2^j} \equiv \pm 1^{-1} \equiv \pm 1$. So $G \leq (\mathbb{Z}/n\mathbb{Z})^*$.

(iii) $G \subsetneq (\mathbb{Z}/n\mathbb{Z})^*$ We must show that there is some element $w \in (\mathbb{Z}/n\mathbb{Z})^*$ that isn't in G. We choose some prime p dividing n. Since n is not a perfect power we have to rewrite it as $n = gp^k$ with p not dividing g. Remember that $x^{2^j} \equiv -1$ has at least one solution in $(\mathbb{Z}/n\mathbb{Z})^*$ by definition, let's call it x_0. By the chinese remainder theorem there exists some $w \in (\mathbb{Z}/n\mathbb{Z})$ such that

$$w \equiv x_0\ (p^k) \quad \wedge \quad w \equiv 1\ (g)$$

Since x_0 by definition is in $(\mathbb{Z}/n\mathbb{Z})^*$ we have $gcd(x_0, n) = 1 \implies gcd(x_0, p^k) = 1 \wedge gcd(x_0, g) = 1 \implies gcd(w, n) = 1 \implies w \in (\mathbb{Z}/n\mathbb{Z})^*$. Also $w^{a2^j} \equiv x_0^{a2^j}\ (p^k)$, but since $x_0^{a2^j} \equiv (x_0^{2^j})^a \equiv -1^a \equiv -1\ (n)$ we have $x_0^{a2^j} \equiv w^{a2^j} \equiv -1\ (p^k)$.
Furthermore $w^{a2^j} \equiv 1\ (g)$. However it follows that $w^{a2^j} \equiv -1\ (p^k) \implies w^{a2^j} \not\equiv 1\ (n)$ and $w^{a2^j} \equiv 1\ (g) \implies w^{a2^j} \not\equiv -1\ (n)$. Therefore $w^{a2^j} \not\equiv \pm 1\ (n)$ and so $w \notin G$ as desired.

By Lagrange's theorem the order of G has to divide the order of $(\mathbb{Z}/n\mathbb{Z})^*$. The ratio has to be an integer and in the worst case (G begin as big as possible) is 2.

2.6 Implemantation in C++

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;
void mcd_euclid_ext(int a, int b, int* res)
{
    int swap[3];/* swap aux vector*/
    int mat[2][2]={{1,0},{0,1}};/*aux matrix*/
    while(b){
        /* update matrix */
        swap[0]=mat[0][0]; swap[1]=mat[0][1];
        mat[0][0]=mat[1][0]+(a/b)*mat[0][0];
        mat[0][1]=mat[1][1]+(a/b)*mat[0][1];
        mat[1][0]=swap[0]; mat[1][1]=swap[1];
        /* finish update*/
        /* classic ecl algo*/
        swap[2] = b;
        b = a % b;
        a = swap[2];
        /* classic ecl algo*/
    }
    bool sign=(mat[1][1]*mat[0][0]-mat[1][0]*mat[0][1]< 0);
    /* chossing c given by second row solutions*/
    res[0]=a;
    res[1]=(sign)?-mat[1][1]:mat[1][1];
    res[2]=(sign)?mat[1][0]:-mat[1][0];
    /* returining vars*/
}
```

Iterative implementation of the classic extended euclidean algorithm using a two by two matrix. We use a vector to store the results. Given the following bezout identity $au + bv = d$ the results are given in the following way $v[0] = d, v[1] = u, v[2] = v$.

```cpp
int power( int x, unsigned int y, int p){
    int res = 1;
    x = x % p;
    while (y > 0) {
        if (y & 1)
            res = (res * x) % p;
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}
```

Iterative implementation of the classic repeated squaring algorithm. We used some tricks using the binary expansion of the exponent y to speed up the algorithm. When there's a one on the binary expansion we multiply what we have to what was saved in the results.

$$x = 4, \quad y = 5 = 0b101 \quad p = 7$$

$$res \leftarrow 4, \quad y \leftarrow 0b010, \quad x \leftarrow 4^2 \equiv 2\ (7)$$

$$y \leftarrow 0b001, \quad x \leftarrow 2^2 \equiv 4\ (7)$$

$$res \leftarrow 4 * 2 \equiv 1\ (7), \quad y \leftarrow 0b000$$

```
bool millerTest(int d, int n) {
    /* find a \in U(n) */
    int check[3];
    int a;
    do{
        a = 2 + rand() % (n - 4);
        mcd_euclid_ext(a,n,check);
    }
    while(check[0] != 1);

    int x = power(a, d, n);

    /* 1st condition */
    if (x == 1)
        return true;

    /* 2nd condition */
    while (d != n - 1){
        x = (x * x) % n;
        d *= 2;
        if (x == n - 1) return true;
     }
    return false;
}

bool isPrime(int n, int k) {
    /* check neg */
    if (n <= 1 || n == 4) return false;
    /* check basic primes */
    if (n <= 3) return true;

    /* finding d */
    int d = n - 1;
    while (d % 2 == 0)
        d /= 2;

    for(int i = 0; i < k; i++){
        if (!millerTest(d, n))
            return false;
    }
    return true;
}
```

3 Soloway Strassen Primality Test

Let's make an observation. If a number n is prime, for all $a \in (\mathbb{Z}/n\mathbb{Z})^*$ it holds

$$\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \ (n)$$

If n is odd, it makes sense to ask yourself if the equation holds. Remember that in this case $\frac{a}{n}$ is the Jacobi symbol not the Legendre symbol. We wanna show, as we showed for the Miller Rabin test that the order of the group G (witnesses that n is not a composite number) is at most half of $(\mathbb{Z}/n\mathbb{Z})^*$.

3.1 Lower Bound on Probability Proof

$$G = \left\{ a \in (\mathbb{Z}/n\mathbb{Z})^* \text{ t.c. } \left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \ (n) \right\}$$

We must show that $G \leq (\mathbb{Z}/n\mathbb{Z})^*$ and $G \neq (\mathbb{Z}/n\mathbb{Z})^*$.

(i) In the first place we define the the map π

$$\begin{aligned} \pi : \ (\mathbb{Z}/n\mathbb{Z})^* &\rightarrow (\mathbb{Z}/n\mathbb{Z})^* \\ a &\mapsto \left(\frac{a}{n}\right) a^{-\frac{(n-1)}{2}} \end{aligned}$$

G is the kernel of this group homomorphism (since the Jacobi symbol and exponentiating something are both group homomorphisms) so it is automatically a subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$.

(ii) Let a be an element of G.

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \ (n) \implies a^{n-1} \equiv 1 \ (n)$$

Since $\frac{a}{n} \in \{\pm 1\}$ the Soloway-Strassen witnesses are contained in the set of the Fermat witnesses (which is a weaker test relying only on the Little Fermat test). We won't study this test because there are no advantages over the SS primality test.

(iii) Let's distinguish two cases. If n is not squarefree, we are done (the Fermat test has as witnesses Carmicheal numbers but if it's squarefree then it's not a Carmicheal number). If n is squarefree, we choose a prime factor p of n $(n = pm)$ and we fix $a \in (\mathbb{Z}/p\mathbb{Z})^*$ a non quadratic residue mod p. At this point we'll use che Chinese Remainder Theorem to construct a number $b \in \mathbb{Z}/n\mathbb{Z}$.

$$b \equiv a \ (p), \ b \equiv 1 \ (m)$$

Since $a \in U(p)$ and $1 \in U(m)$ we have that $b \in U(n) = (\mathbb{Z}/n\mathbb{Z})^*$. To justify this, it is important to remember that the CRT also states $U(m) \times U(p) \cong U(n)$ as a corollary of the famous $\mathbb{Z}_m \times \mathbb{Z}_p \cong \mathbb{Z}_n$. Using some nice properties of the Jacobi Symbol we get

$$\left(\frac{b}{n}\right) = \left(\frac{b}{p}\right)\left(\frac{b}{m}\right) = \left(\frac{a}{p}\right)\left(\frac{1}{m}\right) = -1$$

If this b was in G, it would hold that $b^{n-1/2} \equiv -1 \ (n)$, which implies $b^{n-1/2} \equiv -1 \ (m)$ and this is impossible $(b \equiv 1 \ (m))$.

The argument ends with the Lagrange Theorem just like the Miller-Rabin proof.

3.2 Accuracy

We now know that $\mathbb{P}(n \text{ passing } k \text{ times} \mid n \text{ composite }) \leq 2^{-k}$.

$$\mathbb{P}(n \text{ prime } \mid n \text{ passing the test } k \text{ times }) =$$

$$= \frac{\mathbb{P}(n \text{ prime} \wedge n \text{ passing the test } k \text{ times})}{\mathbb{P}(\ n \text{ prime} \wedge n \text{ passing the test } k \text{ times }) + \mathbb{P}(n \text{ composite} \wedge n \text{ passing the test } k \text{ times})}$$

$$= \frac{\mathbb{P}(n \text{ prime })}{\mathbb{P}(\ n \text{ prime }) + \mathbb{P}(n \text{ composite }) \cdot \mathbb{P}(n \text{ passing the test } k \text{ times} \mid n \text{ composite})}$$

$$= \frac{1}{1 + \frac{\mathbb{P}(n \text{ composite })}{\mathbb{P}(\ n \text{ prime })} \cdot \mathbb{P}(n \text{ passing the test } k \text{ times} \mid n \text{ composite})}$$

$$\mathbb{P}(\text{ SS success }) \approx 1 - \frac{1 - \frac{1}{\log n}}{\frac{1}{\log n}} \cdot 2^{-k}$$

3.3 C++ implementation

```cpp
// To calculate Jacobian symbol of a given number
int calculateJacobian(int a, int n)
{
    if (!a)
        return 0;// (0/n) = 0

    int ans = 1;
    if (a < 0)
    {
        a = -a; // (a/n) = (-a/n)*(-1/n)
        if (n % 4 == 3)
            ans = -ans; // (-1/n) = -1 if n = 3 (mod 4)
    }

    if (a == 1)
        return ans;// (1/n) = 1

    while (a)
    {
        if (a < 0)
        {
            a = -a;// (a/n) = (-a/n)*(-1/n)
            if (n % 4 == 3)
                ans = -ans;// (-1/n) = -1 if n = 3 (mod 4)
        }
        while (a % 2 == 0)
        {
            a = a / 2;
            if (n % 8 == 3 || n % 8 == 5)
                ans = -ans;
        }

        swap(a, n);

        if (a % 4 == 3 && n % 4 == 3)
            ans = -ans;
        a = a % n;

        if (a > n / 2)
            a = a - n;
    }

    if (n == 1)
        return ans;

    return 0;
}
```

```c
bool solowayStrassen(int p, int k)
{
    /* p odd number */
    if (p < 2)
        return false;
    if (p != 2 && p % 2 == 0)
        return false;

    for (int i = 0; i < k; i++)
    {
        // Generate a random number a
        int ans[3];
        int a;
        do{
            a = rand() % (p - 1) + 1;
            mcd_euclid_ext(a,p,ans);

        }
        while(ans[0] != 1);

        int jacobian = (p + calculateJacobian(a, p)) % p;
        int mod = power(a, (p - 1) / 2, p);

        if (!jacobian || mod != jacobian)
            return false;
    }
    return true;
}
```

4 Conclusions

The chances we got to be right is incredibly high for both tests. The MR test though is more likely to be right even though we didn't prove this for simplicity's sake.

$$\mathbb{P}(\text{ MR success }) \approx 1 - \frac{1 - \frac{1}{\log n}}{\frac{1}{\log n}} \cdot 4^{-k}$$

4.1 Proof of True Lower Bound for MR Test

Given N an odd composite integer. The probability that a random integer $a \in [1, N-1]$ is a witness for N is at least $\frac{3}{4}$

Let N be an odd composite number of the form $N = 2^s t + 1$ with t odd, and let $N = q_1 \cdots q_r$ be the factorization of N into powers of distinct primes; note that t must be coprime to all of the q_j. Let $a \in [1, N-1]$ be chosen at random, and put $b = a^t$. If a is not a witness then either $b \equiv 1$ (N) for some in which case $b \equiv 1$ (q_j) for all the q_j, or $b^{2^i} \equiv -1$ (N) for some $0 \le i < s$, in which case $b^{2^i} \equiv -1$ (q_j) for all the q_j. If we put $i = -1$ in the first case, then b mod q_j is an element of order 2^{i+1} in the $2 - \mathit{Sylow}$ subgroup S_j of $(\mathbb{Z}/q_j\mathbb{Z})^*$ for $1 \le j < r$. We will show that the probability that every $b_j \equiv b$ (q_j) lies in S_j and has order 2^i is at most $1/4$.

(i) N is not squarefree. Then some $q_j = p^k$ with $k > 1$. Since p is odd, the group $(\mathbb{Z}/p^k)^*$ is cyclic of order $\phi(p^k) = p^{k-1}(p-1)$, and t is coprime to p, so the probability that b_j lies in S_j at most $1/p^{k-1}$, which is less than $1/4$ if $p^k > 0$. If $q_j = p^k = 9$ then $t \equiv \pm 1$ (6) and only 2 of the 8 nonzero values of a leads to $b_j \in S_j$.

(ii) N is a product of $r > 3$ distinct primes q_j. Each $2 - \mathit{Sylow}$ subgroup S_j is a cyclic of order 2^{k_j} for some $k_j > 1$, and at most half the elements in S_j can have any particular order. If we assume each b_j actually lies in G_j then they are uniformly distributed (since t is odd), and the probability they all have the same order is at most $1/4$.

(iii) N is a product of 2 distinct primes. Write $q_1 = 2^{s_1} t_1 + 1$, and $q_2 = 2^{s_2} t_2 + 1$, with t_1, t_2 odd. Define the random variable X_j to be -1 if b_j does not lie in S_i and otherwise let $X_p = i$, where b_j has order 2^i in S_j. We wish to show $\mathbb{P}(X_1 = X_2 \ge 0) \le 1/4$. Suppose $s_1 > s_2$. Half the elements in S_1 have order $2^{s_1} > 2^{s_2}$, so $\mathbb{P}(0 < X_1 < s_2) \le 1/2$, and $\mathbb{P}(X_2 = X_1 \mid 0 \le X_1 \le s_2) \le 1/2$; therefore $\mathbb{P}(X_1 = X_2 \le 0) \le 1/4$. Now suppose that $s_1 = s_2$. We have

$$2^s t = N - 1 = q_1 q_2 - 1 = (q_1 - 1)(q_2 - 1) + (q_1 - 1) + (q_2 - 1) = 2^s t_1 t_2 + 2^{s_1} t_1 + 2^{s_2} t_2$$

thus if t_1 divides t then it also divides t_2, and conversely.If t_1 and t_2 both divide t, then $t_1 = t_2$ and $q_1 = q_2$, a contradiction. So assume that t_1 doesn't divide t. Then $t_1 \neq 1$ must be divisible by an odd prime $l \ge 3$ that does not divide t. It follows that $\mathbb{P}(X_1 \ge 0) \le 1/3$, and also we have that $\mathbb{P}(X_1 = X_2 \mid X_1 \ge 0) \le 1/2$, therefore $\mathbb{P}(X_1 = X_2 \ge 0) \le 1/6 < 1/4$.

5 Implementation in C++ for RSA protocol

```
int main(){
begin:
    int precise;
    cout << "insert k to have 1/2^k probability to be wrong"<< endl;
ch_k:
    cin >> precise;
    if( precise < 1){
        goto ch_k
        cout << "insert a high positive integer please" << endl;
    }

    srand(time(nullptr));/* change the seed */

    int p,q;
    do{
        p = rand();
        q = rand();
    }
    while(!isPrime(p,precise) || !isPrime(q,precise));



    /* RSA algorithm */

    int phi_n = (p-1)*(q-1);
    int aux[3];
regenerate:
    int e = rand() % phi_n;
    if( e < 0){
        e += phi_n;
    }
    mcd_euclid_ext(e,phi_n,aux);

    if(aux[0]!= 1 ){
        goto regenerate;
    }

    int d = aux[1] % phi_n;
    if( d < 0){
        d+=phi_n;
    }


    cout << "your n is "<< p*q<< endl;
    cout << "your e is " << e << endl;
    cout << "your d is " << d << endl;
    cout << "want to keep this config (Y/N)?"<< endl;
    char yn;
choice:
    cin>> yn;
    if(yn == 'n' || yn == 'N'){
        goto begin;
    }
    if(yn != 'y' && yn !='Y'){
        cout << "c'mon be serious" << endl;
        goto choice;
    }
    return 0;
}
```

6 Curiosities

Actually, there are some deterministic polynomial-time primality tests like the AKS primality test. Unfortunately, these are outperformed by the probabilistic tests we showed, on top of that they are quite complicated. A good example of a deterministic test would be the Miller test, which is a deterministic version of the Miller Rabin test, based on the unproved extended Riemann hypothesis. In the implentation on the algorithms we followed what the theory suggested but if you choose an a which is not coprime to n you already found yourself a witness. Now let's see a fun primality test that is completely useless in general but could be incredibly fast in some cases.

6.1 Wilson Primality Test

Wilson Primality test is based on the Wilson Theorem. A natural number $n > 1$ is a prime number if and only if the product of all positive integers less than n is congruent to -1.

$$(n-1)! \equiv -1 \ (n)$$

This test has no use if we want to operate like MR or SS but can be used in the other direction, to determine the primality of the successors of large factorials, it is indeed a very fast and effective method. This is of limited utility, however.

6.2 Proof

The proof has two sections. The first we'll dive into will be the contrapositive (if n is composite the equation doesn't hold). The second will be the direct one.

(i) Suppose than n is composite. Therefore is divisible by some prime number q where $2 \leq q < n$. Let's suppose for the sake of contradiction that $(n-1)!$ were congruent to -1 mod n. We see that if that holds $(n-1)! \equiv -1 \ (q)$. Since q has to be less than n, it is also in the factorial expansion. This would mean that $(n-1)! \equiv (n-1) \cdots q \cdots 1 \equiv 0 \ (q)$ which is a contradiction.

(ii) It is obvious for $p = 2$, so suppose p is an odd prime, $p \geq 3$. Now let's consider the cyclic multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$. Let the generator a be given.

$$(p-1)! = a^0 a^1 \cdots a^{p-2} = a^{\frac{(p-2)(p-1)}{2}}$$

Now we notice that $a^{\frac{p-1}{2}}$ has order 2 by Fermat's little theorem and since $p-2$ is odd.

$$(a^{\frac{p-1}{2}})^{p-2} = a^{\frac{p-1}{2}}$$

At this point $[-1]$ is the only element of order 2 in $(\mathbb{Z}/p\mathbb{Z})^*$, so we are done.

6.3 Implementation in C++

```cpp
#include <iostream>
#include <ctime>
using namespace std;

int fact(int p){
    if(p <= 1 )
        return 1;
    return p*fact(p-1);
}
bool WilsPrime(int k){
    return (fact(k-1) % k== -1 || fact(k-1) % k == k-1 );
}
int main(){
    srand(time(nullptr));
    int r;
    do{
        r = rand();

    }
    while( WilsPrime(r));
    cout<< "ur randommized prime is "<< r <<endl;
    cout<< "never use this algo again" << endl;

    return 0;
}
```

7 ECPP

We now consider a method that unequivocally prove that a given integer N is prime or composite using elliptic curves. Elliptic curve primality proving (ECCP) was introduced by Goldwasser and Kilian in 1986. It takes advantage of the fact that elliptic curves provide a way to generate abelian groups of varying orders over a fixed finite field if the Hilbert-Nullstellensatz is true . Let's make some definitions first.

Definition. Let $P = (P_x, P_y, P_z)$ be a projective point on the elliptic curve $E/\mathbb{Q}$, with $P_x, P_y, P_z \in \mathbb{Z}$, and let N be a nonzero integer. If $P_z \equiv 0\ (n)$ then P is zero *mod* N; otherwise, P is nonzero mod N. If $gcd(P_z, N) = 1$ then P is strongly nonzero mod N.

Note that if P is strongly nonzero mod N, then P is nonzero mod q for every prime $p \mid N$. When N is prime, the notions of nonzero and strongly nonzero coincide. We now state the theorem, using $\Delta(E) := -16(4A^3 + 27B^2)$ to denote the discriminant of an elliptic curve $E : y^2 = x^3 + Ax + B$ in short Weierstrass form.

Theorem (Goldwasser-Kilian). Let $E/\mathbb{Q}$ be an elliptic curve, and let $M, N > 1$ be integers with $M > (N^{1/4} + 1)^2$ and $N \perp \Delta(E)$, and let $P \in E(\mathbb{Q})$. If MP is zero mod N and $(M/l)P$ is strongly nonzero mod N for every prime l/M then N is prime.

Proof. Suppose for the sake of contradiction that the hypothesis holds and N is composite. Then N has a prime divisor $p \leq \sqrt{N}$, and E has good reduction at p since $N \perp \Delta(E)$. Let M_p be the order of the reduction of P on E modulo p. The point MP is zero mod N and therefore zero mod p, so $M_p \mid M$; and we must have $M_p = M$ since $(M/l)P$ is strongly nonzero mod N and therefore nonzero mod p, for every prime $l \mid M$. Thus P has order M on the reduction of E modulo p, and by the Hasse Theorem, $M \leq (\sqrt{p} + 1)^2$. But we also have $M > (N^{1/4} + 1)^2 \leq (p^{1/2} + 1)^2$, which is our desired contradiction.

In order to apply the theorem, we need to know the prime factors q of M. In particular, we need to be sure that these q are actually prime! Ti simplify matters, we restricts ourselves to the case that $M = q$ is prime, and introduce the notion of primality certificate.

Definition. A primality certificate for p is a tuple of integers (p, A, B, x_1, y_1, q) where $P = (x_1, y_1, 1)$ is a point on the elliptic curve $E : y^2 = x^3 + Ax + B$ over $\mathbb{Q}$, the integer $p > 1$ is prime to $\Delta(E)$, and qP is zero mod p with $q > (p^{1/4} + 1)^2$.

Note that $P = (x_1 : y_1 : 1)$ is strongly, nonzero mod p, since its z- coordinate is 1. Theorem implies that if there exist a primality certificate $(p, ..., q)$ for $N = p$ in which $M = q$ is prime, then q is prime. Thus a primality certificate $(p, ..., q)$ reduces the question q's primality. Using a chain of such certificates, we can reduce to a case in which q is so small that we are happy to test its primality via trial division. This leads to the following.

7.1 Golwasser-Kilian Algorithm

Given an odd integer p a candidate prime and a bound b, with $p > b > 5$, either construct a primality certificate (p, A, B, x_1, y_1, q) with $q \leq (\sqrt{p}+1)^2/2$ or prove that p is composite.

I Pick random integers, $A, x_0, y_0 \in [0, p-1]$, and set $B = y_0^2 - x0^3 - Ax_0$. Repeat until $gcd(4A^3 + 27B^2, p) = 1$, then define $E : y^2 = x^3 + Ax + B$.

II Use Schoof's algorithm to compute the number of points m on the reduction of E modulo p, assuming that p is prime. If anything goes wrong (which it might if p is actually composite), or if $m \notin \mathcal{H}(p)$ then return composite.

III Write $m = cq$, where c is b-smooth and q is b-coarse (all prime factors greater than b). If $c = 1$ or $q \leq (p^{1/4}+1)^2$, then go to step 1

IV Perform a Miller Rabin test on q. If it returns false then go to step 1

V Compute $P = (P_x, P_y, P_z) = c \cdot (x_0 : y_0 : 1)$ on $E/\mathbb{F}_p$. If $gcd(P_z, P) \neq 1$, go to step 1, else let $x_1 \equiv P_x/P_z \ (p)$ and $y_1 \equiv Py/P_z \ (p)$

VI Compute $Q = (Q_x, Q_y, Q_z) = q * (x_1 : y_1 : 1)$ on $E/\mathbb{F}_p$. If $Q_z \not\equiv 0 \ (p)$ then return composite

VII If $q > b$ then recursively verify that q is prime using inputs q and b; otherwise, verify that q is prime by trial division. Id q is found to be composite, go to step 1.

VIII Output the certificate $(p, A, \hat{B}, x_1, y_1, q)$, where $\hat{B} \equiv B \ (p)$ is chosen so that we have $y_1^2 = x_1^3 + Ax_1 + \hat{B}$ over $\mathbb{Q}$ not just $\mathbb{F}_p$