ECPP in PARI/GP

Jared Asuncion

29 January 2018

Problem

Given an integer N, prove that it is prime.

Proposition

Let $N > 6$ be an integer. If there exists:

- *an integer m*
- *a prime q*
- *an elliptic curve E over $\mathbb{Z}/N\mathbb{Z}$*
- *and a point P on E*

such that

- $m = qs$ *for some* $s \in \mathbb{Z}$
- $q > \left(N^{1/4} + 1\right)^2$
- $mP = \infty$
- $sP \neq \infty$

then N is prime.

Manual verification

?

Manual verification

```
? N = 97;
?
```

Manual verification

```
? N = 97;
? m = 93;
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
? s = m/q;
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
? s = m/q;
? (N^(1/4) + 1)^2
% = 17.125435787226096882657755669491649229
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
? s = m/q;
? (N^(1/4) + 1)^2
% = 17.125435787226096882657755669491649229
? ellmul(E,P,m)
% = [0]
?
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
? s = m/q;
? (N^(1/4) + 1)^2
% = 17.125435787226096882657755669491649229
? ellmul(E,P,m)
% = [0]
? ellmul(E,P,s)
% = [Mod(23, 97), Mod(46, 97)]
```

Manual verification

```
? N = 97;
? m = 93;
? q = 31;
? E = ellinit( [Mod(69, N), Mod(2, N)] );
? P = [12, 91];
? s = m/q;
? (N^(1/4) + 1)^2
% = 17.125435787226096882657755669491649229
? ellmul(E,P,m)
% = [0]
? ellmul(E,P,s)
% = [Mod(23, 97), Mod(46, 97)]
```

If $q = 31$ is prime, then $N = 97$ is prime.

First examples of ecpp

?

First examples of ecpp

```
? N = 97;
?
```

First examples of ecpp

```
? N = 97;
? ecpp(N)
% = 97
?
```

First examples of ecpp

```
? N = 97;
? ecpp(N)
% = 97
? N = 51;
?
```

First examples of ecpp

```
? N = 97;
? ecpp(N)
% = 97
? N = 51;
? ecpp(N)
% = 0
?
```

First examples of ecpp

```
? N = 97;
? ecpp(N)
% = 97
? N = 51;
? ecpp(N)
% = 0
? N = ecpp(2^100000);
?
```

First examples of ecpp

```
? N = 97;
? ecpp(N)
% = 97
? N = 51;
? ecpp(N)
% = 0
? N = ecpp(2^100000);
? ecpp(N)
% = 0
```

Bigger example for ecpp

?

Bigger example for ecpp

```
? N = randomprime(10^50)
% = 62180214830762693481322910416470452917087712765l7
?
```

Bigger example for ecpp

```
? N = randomprime(10^50)
% = 62180214830762693481322910416470452917087712765l7
? cert = ecpp(N)
```

Bigger example for ecpp

```
? N = randomprime(10^50)
% = 62180214830762693481322910416470452917087712765l7
? cert = ecpp(N)
% = [[62180214830762693481322910416470452917087712765l7, -11
6885889616512420038448l, 104197826361, 0, [2722194275352954 6
79804716336431441756799877037507, 175958981765049204609477l3
5878551002500244807467O]], [596751554253496444309794963565O1
173759, -57415119056906282?2, 112816, 596751554253496444309?
949635644950588 4, [767459446148065486489158698799211076 8, 25
3673327046939063084752484331809276 6]], [528960036035222348 2
1941044055977 7, -3655711335140995 1, 2416487659, 0, [5013395l
3959449988194168803017069, 104622117438367884727793381682328
]], [21889622902279526384973l, 836507980571, 27916833, 0, [4
272841864056506938396 6, 72784625979380726131541]]]
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? ??ecpp
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
? N = cert[1][1]
% = 62180214830762693481322910416470452917087712765l7
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
? N = cert[1][1]
% = 62180214830762693481322910416470452917087712765 17
? t = cert[1][2]
% = -116885889616512420038448 1
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
? N = cert[1][1]
% = 62180214830762693481322910416470452917087712765 17
? t = cert[1][2]
% = -116885889616512420038448 1
? s = cert[1][3]
% = 104197826361
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
? N = cert[1][1]
% = 62180214830762693481322910416470452917087712765l7
? t = cert[1][2]
% = -116885889616512420038448l
? s = cert[1][3]
% = 104197826361
? a4 = cert[1][4]
% = 0
?
```

Interpreting the certificate

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ??ecpp
? #cert
% = 4
? N = cert[1][1]
% = 62180214830762693481322910416470452917087712765 17
? t = cert[1][2]
% = -116885889616512420038448 1
? s = cert[1][3]
% = 104197826361
? a4 = cert[1][4]
% = 0
? P = cert[1][5]
% = [272219427535295467980471633643144175679987703750 7, 1759
589817650492046094771358785510025002448074670]
```

Using ecppexport

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
?
```

Using ecppexport

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? expo = ecppexport(cert)
% = "\n[1]\n N = 6218021483076269348132291041647045291 7[+++]
?
```

Using ecppexport

```
? N = 621802148307626934813229104164704529170877127651 7;
? cert = ecpp(N);
? expo = ecppexport(cert)
% = "\n[1]\n N = 62180214830762693481322910416470452917[+++]
? print(expo)
[1]
 N = 621802148307626934813229104164704529170877127651 7
 t = -116885889616512420038448 1
 s = 104197826361
a4 = 0
 D = -3
 m = 621802148307626934813229221050594145683297166099 9
 q = 596751554253496444309794963565011737 59
 E = [0, 472676336470774367187664723891618695222023647759 5]
 P = [29363273453865917258980875171711126981091389024 00[+++]
```

Exporting to MAGMA/PRIMO

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
?
```

Exporting to MAGMA/PRIMO

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? magma = ecppexport(cert,1)
% = "[* [* 6218021483076269348132291041647045291708771276517[+++]
?
```

Exporting to MAGMA/PRIMO

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? magma = ecppexport(cert,1)
% = "[* [* 62180214830762693481322910416470452917087712[+++]
? primo = ecppexport(cert,2)
% = "[PRIMO - Primality Certificate]\nFormat=4\nTestCou[+++]
?
```

Exporting to MAGMA/PRIMO

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
? magma = ecppexport(cert,1)
% = "[* [* 62180214830762693481322910416470452917087712[+++]
? primo = ecppexport(cert,2)
% = "[PRIMO - Primality Certificate]\nFormat=4\nTestCou[+++]
? write("primo.out", primo);
```

Is the certificate valid?

```
? N = 62180214830762693481322910416470452917087712765l7;
? cert = ecpp(N);
?
```

Is the certificate valid?

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ecppisvalid(cert)
% = 1
?
```

Is the certificate valid?

```
? N = 621802148307626934813229104164704529170877127651 7;
? cert = ecpp(N);
? ecppisvalid(cert)
% = 1
? cert[4][5] = [11891533876854949407747 6, 2163004367571848091
16130];
?
```

Is the certificate valid?

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ecppisvalid(cert)
% = 1
? cert[4][5] = [11891533876854949407747 6, 2163004367571848091
16130];
? ecppisvalid(cert)
% = 0
?
```

Is the certificate valid?

```
? N = 62180214830762693481322910416470452917087712765 17;
? cert = ecpp(N);
? ecppisvalid(cert)
% = 1
? cert[4][5] = [11891533876854949407747 6, 21630043675718480916130];
? ecppisvalid(cert)
% = 0
? ecppisvalid(vector(100, i, i))
% = 0
```

Assume that we have a list $\mathcal{L}$ of discriminants we want to try.

PHASE I: Find m and q.

For each discriminant D under consideration:

- **Write $\mathbf{m = N + 1 - U}$**
  where (U, V) is an integer solution to $U^2 + |D|V^2 = 4N$.[a]
  - involves taking a square root of D modulo N
  - involves doing a half-gcd
- **Try to write $\mathbf{m}$ as $\mathbf{m = qs}$**
  where q and s are as in the theorem.
  - involves finding *small* factors of m
  - good idea to check pseudoprimality of q

[a] m can take at most six other forms as well

Assume that we have the corresponding (D, m, q) from Phase 1.

PHASE II: Find E and P.

- Find a root j modulo N of the Hilbert class polynomial of $\mathbb{Q}(\sqrt{D})$.
  - involves computing a class polynomial (of degree h)
  - involves finding a root of this polynomial modulo N
- Find an elliptic curve E modulo N with j-invariant j and a point P satisfying the conditions in the theorem.
  - finding the twist involves finding a quadratic non-residue

Proposition

A fundamental discriminant D is of the form:

$$D = \prod_{i=0}^{t} p_i^*$$

where

- $p_0^* \in \{-8, -4, 1, 8\}$
- $p_i^* \equiv 1 \,(mod\ 4)$ *and* $|p_i^*|$ *is prime for* $i > 0$.

Proposition

If $U^2 + |D|V^2 = 4N$ *has integer solutions* (U, V), *then*

- $\left(\frac{D}{N}\right) = 1$
- $\left(\frac{p^*}{N}\right) = 1$ *for all* p^* *dividing* D.

Finding a solution to $U^2 + |D|V^2 = 4N$.

Globally:

- Keep the prime factorization of each D on the list $\mathcal{L}$.
- Only consider D which are B-smooth for some B.
- Only consider D which are associated to low class numbers.

For each N:

- For each D, test for necessary conditions for the existence of a solution.
- Find $\sqrt{D}$ modulo N by finding the square roots of primes dividing D.
- Remember the square roots of the primes for future use.
- Find (and remember) a quadratic nonresidue g.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.

- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.

- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.
  - $\gcd(2154, 3) = 3$. Take $2154/3 = 927$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.
  - $\gcd(2154, 3) = 3$. Take $2154/3 = 927$.
  - $\gcd(927, 3) = 3$. Take $927/3 = 309$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.
  - $\gcd(2154, 3) = 3$. Take $2154/3 = 927$.
  - $\gcd(927, 3) = 3$. Take $927/3 = 309$.
  - $\gcd(309, 3) = 3$. Take $309/3 = 103$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 614889782588491410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 614889782588491410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.
  - $\gcd(2154, 3) = 3$. Take $2154/3 = 927$.
  - $\gcd(927, 3) = 3$. Take $927/3 = 309$.
  - $\gcd(309, 3) = 3$. Take $309/3 = 103$.
  - $\gcd(103, 3) = 1$.

Removing small factors

- Precompute the product $\mathcal{P}$ of all primes less than S.
- Divide m by its gcd with $\mathcal{P}$. Do this repeatedly until they are coprime.
- There is a clever way to treat multiple m at once.

- Let $\mathcal{P} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot \cdots \cdot 47 = 61488978258849 1410$.
- Let $m_1 = 1295$, $m_2 = 2781$, and $M = m_1 m_2 = 3601395$.
- $\mathcal{P} = 61488978258849 1410 \equiv 3119655 \pmod{M}$
- $\mathcal{P} \equiv 3119655 \equiv 0 \pmod{1295}$.
  - $\gcd(0, 1295) = 1295$.
  - $q = 1$.
- $\mathcal{P} \equiv 3119655 \equiv 2154 \pmod{2781}$.
  - $\gcd(2154, 3) = 3$. Take $2154/3 = 927$.
  - $\gcd(927, 3) = 3$. Take $927/3 = 309$.
  - $\gcd(309, 3) = 3$. Take $309/3 = 103$.
  - $\gcd(103, 3) = 1$.
  - $q = 103$.

Summary

- Look at next D.
- Check necessary conditions. If fail, go back to start.
- Find $\sqrt{D}$ modulo N.
  - by finding $\sqrt{q^*} \pmod{N}$ for all $q^* | D$
  - or just accessing it from memory if it has been done before
- Solve for $U^2 + |D|V^2 = 4N$.
- Let $m = N + 1 \pm U$ (or more if $D = -3, -4$).
- Add the m's to the list to be factored.
- If list is not large enough, go back to start.
- If list is large enough, do the batch factoring.
- For each m, check if the remaining q is pseudoprime.
- If yes, try to prove q is prime.

What to do when the list $\mathcal{L}$ runs out?

- If we are at the top of the recursion tree, then expand the list to include higher class numbers.
- Otherwise, mark that instance as FAIL and resume the computations one level up.

This results in solving one polynomial of huge degree. The rest have small degrees.

Other option explored:

- Expand the list while in the middle of the tree.

Assume that we have the corresponding (D, m, q) from Phase 1.

PHASE II: Find E and P.

- Find a root j modulo N of the Hilbert class polynomial of $\mathbb{Q}(\sqrt{D})$.
  - involves computing a class polynomial (of degree h)
  - involves finding a root of this polynomial modulo N
- Find an elliptic curve E modulo N with j-invariant j and a point P satisfying the conditions in the theorem.
  - finding the twist involves finding a quadratic non-residue

Finding a root j of the Hilbert class polynomial.

- Remember and reuse previous computations.
- Prioritize discriminants D which give polynomials of lower degree (during the first phase).
- Use class polynomials obtained from other invariants.

Trick to find one root of a totally split polynomial f mod p:

- Basic trick:
  $\gcd\left(f, X^{(p-1)/2} - 1\right)$ or $\gcd\left(f, X^{(p-1)/2} + 1\right)$ modulo p.
  This splits the roots into two "buckets".
- New trick:
  $\gcd\left(f, X^{(p-1)/n} - \zeta_n^k\right)$ modulo p for all k.
  This splits the roots into k "buckets". (where $k|(p-1)$).

Finding an elliptic curve E and a point P.

- Use the quadratic nonresidue g to find twists of elliptic curves.

Average runtime for a sample size of 100 primes, single thread.

	1000 bits	2000 bits	3000 bits	4000 bits
Magma 2.11-13	$\geq$ 60.00	-	-	-
ecpp-dj-1.04	2.43	57.4	383	1731
Pari 2.10 `isprime`	3.02	33.9	175	751
Primo 4.21	3.90	28.0	124	438
Pari 2.10 `ecpp`	3.04	24.7	113	442

Runtime for each phase, single thread:

		1000 bits	2000 bits	3000 bits	4000 bits
Primo 4.21	I	3.06	20.4	91.1	340
	II	0.84	7.5	32.6	98
Pari ecpp	*	0.31	2.3	4.7	26
	I	1.03	12.0	63.4	260
	II	1.71	10.4	44.7	157

(*) is the average time spent for precomputations

Average runtime for each step of Pari ecpp:

	1000 bits	2000 bits	3000 bits	4000 bits
$\sqrt{D} \pmod N$	0.12	4.52	32.2	168
factoring smalls	0.76	4.10	11.8	23
pseudoprimality	0.15	3.33	19.4	68
class polynomial	1.14	3.02	5.0	7
root mod N	0.22	4.94	31.1	128
find E and P	0.35	2.42	8.3	21

- Phase I operations are done more frequently as N becomes higher. It is harder to express $m = qs$ as $m \approx N$ becomes bigger.
- There is a big spike at 4000 bits for computing the roots. This is because the degree can go higher than 40.

Further

- Use intermediate fields of H_K/K to obtain lower degree polynomials and find their roots instead.
  - We will seldom need to find roots of a polynomial of huge degree.
  - Because of this, we can afford a longer list $\mathcal{L}$.
  - We go up the recursion tree less frequently because we have a higher chance to succeed.
- Parallel computation.
  - Phase II is done. (Pari/GP Atelier 2018)
  - Phase I is trickier.

New `libpari` commands

- `Fp_sqrt_i`
  find the square root of x given a quadratic nonresidue mod p
- `cornacchia2_sqrt`
  finds the solution to $U^2 + |D|V^2 = 4N$ given $\sqrt{D} \pmod{N}$
- `disc_best_modinv`
  find the invariant to use with `polclass` such that the height of the output is low
- `FpX_oneroot_split`
  finds one root of a totally split polynomial over $F_p[X]$
- removing small factors by batch

Thank you for listening!

References


A. Atkin and F. Morain, *Elliptic curves and primality proving*, Mathematics of Computation **61** (1993), no. 203, 29–68.


J. Franke, T. Kleinjung, F. Morain, and T. Wirth, *Proving the primality of very large numbers with fastecpp*, pp. 194–207, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.


F. Morain, *Implementing the asymptotically fast version of the elliptic curve primality proving algorithm*.