

# Point Counting Algorithms - A Survey

CSCI 599 Prof. Ming Deh-Huang

Heekwan Lee, Iftikhar A Burhanuddin

# Outline

# Outline

Computing the order of a group

# Outline

Computing the order of a group  
Naive Algorithm

# Outline

Computing the order of a group  
Naive Algorithm  
Baby Step Giant Step Algorithm

# Outline

Computing the order of a group  
Naive Algorithm  
Baby Step Giant Step Algorithm  
Schoof's Algorithm

# Computing the order of a group

- General abelian group

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time ... basis of RSA

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time ... basis of RSA
- $E(\mathbb{F}_q)$

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time ... basis of RSA
- $E(\mathbb{F}_q)$ 
  - $E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$  with  $n_1|n_2, n_1|q-1$ ... group maybe cyclic

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time ... basis of RSA
- $E(\mathbb{F}_q)$ 
  - $E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$  with  $n_1|n_2, n_1|q-1$ ... group maybe cyclic
  - Polynomial time

# Computing the order of a group

- General abelian group - exponential time using Pollard Rho

If we have information about the group structure:

- $\mathbb{Z}/n\mathbb{Z}$ 
  - $|\mathbb{Z}/n\mathbb{Z}| = n$
  - Constant time
- $\mathbb{Z}/m\mathbb{Z}^*$ 
  - $|\mathbb{Z}/m\mathbb{Z}^*| = \phi(m)$
  - Sub exponential time ... basis of RSA
- $E(\mathbb{F}_q)$ 
  - $E(\mathbb{F}_q) \cong \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$  with  $n_1|n_2, n_1|q-1$ ... group maybe cyclic
  - Polynomial time
  - Important operation in ECC ...one of the ways to generate *good* curves

# The Hasse Range

**Hasse (1934):** *Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  with  $q$  elements and given by the Weierstraß equation*

$$Y^2 = X^3 + aX + b$$

where  $a, b \in \mathbb{F}_q$  then

$$\begin{aligned} \#E(\mathbb{F}_q) &= q + 1 - t \\ |t| &\leq 2\sqrt{q} \end{aligned}$$

# Naive Algorithm

The number of points in  $E(\mathbb{F}_p)$  with given  $X$ -coordinate  $x \in \mathbb{F}_p$  is

# Naive Algorithm

The number of points in  $E(\mathbb{F}_p)$  with given  $X$ -coordinate  $x \in \mathbb{F}_p$  is

$$1 + \left( \frac{x^3 + ax + b}{p} \right)$$

# Naive Algorithm

The number of points in  $E(\mathbb{F}_p)$  with given  $X$ -coordinate  $x \in \mathbb{F}_p$  is

$$1 + \left( \frac{x^3 + ax + b}{p} \right)$$

Therefore

# Naive Algorithm

The number of points in  $E(\mathbb{F}_p)$  with given  $X$ -coordinate  $x \in \mathbb{F}_p$  is

$$1 + \left(\frac{x^3 + ax + b}{p}\right)$$

Therefore

$$\#E(\mathbb{F}_p) = 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + ax + b}{p}\right)\right)$$

# Naive Algorithm

The number of points in  $E(\mathbb{F}_p)$  with given  $X$ -coordinate  $x \in \mathbb{F}_p$  is

$$1 + \left(\frac{x^3 + ax + b}{p}\right)$$

Therefore

$$\begin{aligned}\#E(\mathbb{F}_p) &= 1 + \sum_{x \in \mathbb{F}_p} \left(1 + \left(\frac{x^3 + ax + b}{p}\right)\right) \\ &= 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{x^3 + ax + b}{p}\right)\end{aligned}$$

# Naive Algorithm

This is an  $O(p^{1+\varepsilon})$  method and can be used for  $p < 10000$  [HC]

# Naive Algorithm

This is an  $O(p^{1+\varepsilon})$  method and can be used for  $p < 10000$  [HC]

On a PIII, 700 MHz 128 MB primes of the form  $k \cdot 2^n + 1$

| $y^2 = x^3 + ax + b$                  | $char \mathbb{F}_p$ [# Digits]        | $\#E(\mathbb{F}_p)$ | Time                   |
|---------------------------------------|---------------------------------------|---------------------|------------------------|
| $y^2 = x^3 + 3x + 1$                  | $3 * 2^1 + 1 = 7$ [1]                 | 12                  | 10 ms                  |
| $y^2 = x^3 + 33x + 60$                | $9 * 2^3 + 1 = 73$ [2]                | 69                  | 10 ms                  |
| $y^2 = x^3 + 261x + 332$              | $7 * 2^6 + 1 = 449$ [3]               | 420                 | 20 ms                  |
| $y^2 = x^3 + 6186x + 3381$            | $31 * 2^8 + 1 = 7937$ [4]             | 8040                | 250 ms                 |
| $y^2 = x^3 + 51914x + 20880$          | $1 * 2^{16} + 1 = 65537$ [5]          | 65280               | 2, 103 ms              |
| $y^2 = x^3 + 288149x + 134680$        | $3 * 2^{18} + 1 = 786433$ [6]         | 786205              | 27, 479 ms             |
| $y^2 = x^3 + 4078904x + 2662894$      | $11 * 2^{19} + 1 = 5767169$ [7]       | 5765985             | 3 mn, 20, 878 ms       |
| $y^2 = x^3 + 2903151x + 19802935$     | $11 * 2^{21} + 1 = 23068673$ [8]      | 23060676            | 13 mn, 35, 462 ms      |
| $y^2 = x^3 + 13523297x + 33527882$    | $5 * 2^{25} + 1 = 167772161$ [9]      | 167776175           | 1 h, 37 mn, 12, 817 ms |
| $y^2 = x^3 + 439167662x + 2995643896$ | $13 * 2^{28} + 1 = 3489660929 =$ [10] | 3489664260          | > 11 h, 15 mn          |

but for larger fields...we'll use more efficient algos

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$m = tg + b$$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$m = tg + b$$

$$O = [m]P = [tg]P + [b]P$$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$m = tg + b$$

$$O = [m]P = [tg]P + [b]P$$

$$-[tg]P = [b]P$$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$\begin{aligned}m &= tg + b \\O &= [m]P = [tg]P + [b]P \\-[tg]P &= [b]P(*)\end{aligned}$$

We want to minimize  $b + g$  subject to  $b.g = m$ ,

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$\begin{aligned}m &= tg + b \\O &= [m]P = [tg]P + [b]P \\-[tg]P &= [b]P(*)\end{aligned}$$

We want to minimize  $b + g$  subject to  $b.g = m$ ,  $\Rightarrow b = g = q^{\frac{1}{4}}$

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$\begin{aligned}m &= tg + b \\O &= [m]P = [tg]P + [b]P \\-[tg]P &= [b]P(*)\end{aligned}$$

We want to minimize  $b + g$  subject to  $b.g = m$ ,  $\Rightarrow b = g = q^{\frac{1}{4}}$

Plugging value of  $b$  and  $g$  in (\*) gives,

# Baby Step Giant Step Algorithm

This is a generic technique which can be applied to finite abelian groups to determine group order, compute discrete log, ...

If the size of search space is  $N$ , this method answers the question in  $O(\sqrt{N})$

Hasse tells us that the order of the error is  $4\sqrt{q}$  and hence we can compute  $\#E(\mathbb{F}_q)$  in  $O(q^{\frac{1}{4}+\varepsilon})$

Say  $P \in E(\mathbb{F}_q)$  and  $\#E(\mathbb{F}_q) = m$  and

$$\begin{aligned}m &= tg + b \\O &= [m]P = [tg]P + [b]P \\-[tg]P &= [b]P(*)\end{aligned}$$

We want to minimize  $b + g$  subject to  $b.g = m$ ,  $\Rightarrow b = g = q^{\frac{1}{4}}$

Plugging value of  $b$  and  $g$  in (\*) gives,  $-[tq^{\frac{1}{4}}]P = [b]P$

# Baby Step Giant Step Algorithm

**BSGS algorithm:**

# Baby Step Giant Step Algorithm

**BSGS algorithm:**

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

# Baby Step Giant Step Algorithm

## **BSGS algorithm:**

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

# Baby Step Giant Step Algorithm

## BSGS algorithm:

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1. Select a random point  $P$  in  $E(\mathbb{F}_q)$  and compute  $[b]P$ , where  $0 < b < q^{\frac{1}{4}}$  and store them in some *convenient* manner

# Baby Step Giant Step Algorithm

## BSGS algorithm:

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1. Select a random point  $P$  in  $E(\mathbb{F}_q)$  and compute  $[b]P$ , where  $0 < b < q^{\frac{1}{4}}$  and store them in some *convenient* manner
2. Next compute  $-[tq^{\frac{1}{4}}]P$  and check if it is in the table.
3. If yes obtain group order by using the indices,

# Baby Step Giant Step Algorithm

## BSGS algorithm:

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1. Select a random point  $P$  in  $E(\mathbb{F}_q)$  and compute  $[b]P$ , where  $0 < b < q^{\frac{1}{4}}$  and store them in some *convenient* manner
2. Next compute  $-[tq^{\frac{1}{4}}]P$  and check if it is in the table.
3. If yes obtain group order by using the indices, otherwise increment  $t$  and go to step 2

# Baby Step Giant Step Algorithm

## BSGS algorithm:

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1. Select a random point  $P$  in  $E(\mathbb{F}_q)$  and compute  $[b]P$ , where  $0 < b < q^{\frac{1}{4}}$  and store them in some *convenient* manner
2. Next compute  $-[tq^{\frac{1}{4}}]P$  and check if it is in the table.
3. If yes obtain group order by using the indices, otherwise increment  $t$  and go to step 2

There are some bugs...Mestre's enhancement

# Baby Step Giant Step Algorithm

This is an  $O(p^{\frac{1}{4}+\varepsilon})$  method and can be used for  $p < 10^{30}$  [PGP]

# Baby Step Giant Step Algorithm

This is an  $O(p^{\frac{1}{4}+\varepsilon})$  method and can be used for  $p < 10^{30}$  [PGP]

On a PIII, 700 MHz, 128 MB primes of the form  $k \cdot 2^n + 1$

# Baby Step Giant Step Algorithm

This is an  $O(p^{\frac{1}{4}+\varepsilon})$  method and can be used for  $p < 10^{30}$  [PGP]

On a PIII, 700 MHz, 128 MB primes of the form  $k \cdot 2^n + 1$

| $y^2 = x^3 + ax + b$               | $char \mathbb{F}_p$ [# Digits]   | $\#E(\mathbb{F}_p)$ | <i>Time</i>      |
|------------------------------------|----------------------------------|---------------------|------------------|
| $y^2 = x^3 + 3x + 1$               | $3 * 2^1 + 1 = 7$ [1]            | 12                  | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 33x + 60$             | $9 * 2^3 + 1 = 73$ [2]           | 69                  | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 261x + 332$           | $7 * 2^6 + 1 = 449$ [3]          | 420                 | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 6186x + 3381$         | $31 * 2^8 + 1 = 7937$ [4]        | 8040                | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 51914x + 20880$       | $1 * 2^{16} + 1 = 65537$ [5]     | 65280               | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 288149x + 134680$     | $3 * 2^{18} + 1 = 786433$ [6]    | 786205              | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 4078904x + 2662894$   | $11 * 2^{19} + 1 = 5767169$ [7]  | 5765985             | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 2903151x + 19802935$  | $11 * 2^{21} + 1 = 23068673$ [8] | 23060676            | 0 – 30 <i>ms</i> |
| $y^2 = x^3 + 13523297x + 33527882$ | $5 * 2^{25} + 1 = 167772161$ [9] | 167776175           | 0 – 30 <i>ms</i> |

# Baby Step Giant Step Algorithm

|  |
|--|
| $y^2 = x^3 + 46257640654x + 380391811$<br>$char \mathbb{F}_p [\# \text{ Digits}] = 49 * 2^{30} + 1 = 52613349377$ [11]<br>$\#E(\mathbb{F}_p) = 52613138470$<br>$Time = 10 \text{ ms}$  |
| $y^2 = x^3 + 173075890854290x + 15340676211085$<br>$char \mathbb{F}_p [\# \text{ Digits}] = 35 * 2^{45} + 1 = 1231453023109121$ [16]<br>$\#E(\mathbb{F}_p) = 1231453038889714$<br>$Time = 130 \text{ ms}$                        |
| $y^2 = x^3 + 19997399169037516x + 25109674361472332$<br>$char \mathbb{F}_p [\# \text{ Digits}] = 17 * 2^{51} + 1 = 38280596832649217$ [17]<br>$\#E(\mathbb{F}_p) = 38280596799790406$<br>$Time = 251 \text{ ms}$                 |
| $y^2 = x^3 + 3310642384618942998x + 2216029861822881760$<br>$char \mathbb{F}_p [\# \text{ Digits}] = 29 * 2^{57} + 1 = 4179340454199820289$ [19]<br>$\#E(\mathbb{F}_p) = 4179340454698527751$<br>$Time = 1,041 \text{ ms}$       |
| $y^2 = x^3 + 105468761489682535187x + 40246170019083146395$<br>$char \mathbb{F}_p [\# \text{ Digits}] = 9 * 2^{65} + 1 = 332041393326771929089$ [21]<br>$\#E(\mathbb{F}_p) = 332041393291195225233$<br>$Time = 8,041 \text{ ms}$ |

# Baby Step Giant Step Algorithm

|  |
|--|
| $y^2 = x^3 + 36475090356678594228271x + 3888898517015898167970$<br>$\text{char } \mathbb{F}_p [\# \text{ Digits}] = 39 * 2^{70} + 1 = 46043073207979040833537$ [23]<br>$\#E(\mathbb{F}_p) = 46043073207990880985350$<br>$\text{Time} = \mathbf{10,976 \text{ ms}}$                                 |
| $y^2 = x^3 + 112471453303800447633864x + 90963216517616084604445$<br>$\text{char } \mathbb{F}_p [\# \text{ Digits}] = 5 * 2^{75} + 1 = 188894659314785808547841$ [24]<br>$\#E(\mathbb{F}_p) = 188894659314576663574008$<br>$\text{Time} = \mathbf{21,511 \text{ ms}}$                              |
| $y^2 = x^3 + 1120655742151729212438458x + 2083002591035693767455878$<br>$\text{char } \mathbb{F}_p [\# \text{ Digits}] = 15 * 2^{78} + 1 = 4533471823554859405148161$ [25]<br>$\#E(\mathbb{F}_p) = 4533471823556601964187121$<br>$\text{Time} = \mathbf{1 \text{ mn}, 3,861 \text{ ms}}$           |
| $y^2 = x^3 + 153540579541521345473887025x + 171108961758054635503737369$<br>$\text{char } \mathbb{F}_p [\# \text{ Digits}] = 5 * 2^{85} + 1 = 193428131138340667952988161 =$ [27]<br>$\#E(\mathbb{F}_p) = 193428131138339230512213786$<br>$\text{Time} = \mathbf{1 \text{ mn}, 50,830 \text{ ms}}$ |
| $y^2 = x^3 + 94010262021911566563859251729x + 83424307937653472480453887283$<br>$\text{char } \mathbb{F}_p [\# \text{ Digits}] = 27 * 2^{92} + 1 = 133697524242821069689105416193 =$ [30]<br>$\#E(\mathbb{F}_p) = \mathbf{\text{not enough memory}}$   |

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

Consider the  $q$ -th power Frobenius isogeny acting on  $E$

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

Consider the  $q$ -th power Frobenius isogeny acting on  $E$

$$\begin{aligned} \phi : E(\overline{\mathbb{F}}_q) &\rightarrow E^q(\overline{\mathbb{F}}_q) \\ (x, y) &\mapsto (x^q, y^q) \\ O &\mapsto O \end{aligned}$$

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

Consider the  $q$ -th power Frobenius isogeny acting on  $E$

$$\begin{aligned} \phi : E(\overline{\mathbb{F}}_q) &\rightarrow E^q(\overline{\mathbb{F}}_q) \\ (x, y) &\mapsto (x^q, y^q) \\ O &\mapsto O \end{aligned}$$

Say  $\hat{\phi}$  denotes the dual isogeny of  $\phi$

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

Consider the  $q$ -th power Frobenius isogeny acting on  $E$

$$\begin{aligned} \phi : E(\overline{\mathbb{F}}_q) &\rightarrow E^q(\overline{\mathbb{F}}_q) \\ (x, y) &\mapsto (x^q, y^q) \\ O &\mapsto O \end{aligned}$$

Say  $\hat{\phi}$  denotes the dual isogeny of  $\phi$

$$\begin{aligned} \hat{\phi} : E^q(\overline{\mathbb{F}}_q) &\rightarrow E(\overline{\mathbb{F}}_q) \\ O &\mapsto O \end{aligned}$$

# Schoof's Algorithm

Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}_q$  of char  $p > 3$  with  $q$  elements. The number of  $\mathbb{F}_q$  rational points on  $E$  is denoted by  $\#E(\mathbb{F}_q)$

Consider the  $q$ -th power Frobenius isogeny acting on  $E$

$$\begin{aligned} \phi : E(\overline{\mathbb{F}}_q) &\rightarrow E^q(\overline{\mathbb{F}}_q) \\ (x, y) &\mapsto (x^q, y^q) \\ O &\mapsto O \end{aligned}$$

Say  $\hat{\phi}$  denotes the dual isogeny of  $\phi$

$$\begin{aligned} \hat{\phi} : E^q(\overline{\mathbb{F}}_q) &\rightarrow E(\overline{\mathbb{F}}_q) \\ O &\mapsto O \end{aligned}$$

Action of Frobenius on the  $\mathbb{F}_q$  rational points is trivial!

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is,

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

$$\phi^2(P) - [t]\phi(P) + [q](P) = O$$

.

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

$$\phi^2(P) - [t]\phi(P) + [q](P) = O$$

.

**Question:** Why not select a *suitable*  $P = (x, y) \in E(\overline{\mathbb{F}}_q)$  and plug into any of the above equations and determine  $t$ ?

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

$$\phi^2(P) - [t]\phi(P) + [q](P) = O$$

.

**Question:** Why not select a *suitable*  $P = (x, y) \in E(\overline{\mathbb{F}}_q)$  and plug into any of the above equations and determine  $t$ ?

$$\phi^2(P) + [q](P) = [t]\phi(P)$$

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

$$\phi^2(P) - [t]\phi(P) + [q](P) = O$$

.

**Question:** Why not select a *suitable*  $P = (x, y) \in E(\overline{\mathbb{F}}_q)$  and plug into any of the above equations and determine  $t$ ?

$$\phi^2(P) + [q](P) = [t]\phi(P) \quad \mathbf{ECDLP} :-(\$$

The *trace* of the Frobenius isogeny is defined as the number  $t$  such that  $\phi + \hat{\phi} = [t]$ , that is, for  $P \in E(\overline{\mathbb{F}}_q)$

$$\phi(P) - [t](P) + \hat{\phi}(P) = O$$

Also the trace also satisfies the equation  $\#E(\mathbb{F}_q) = q + 1 - t$

Applying  $\phi$  to both sides and using the fact that  $\phi\hat{\phi} = [q]$ , we obtain

$$\phi^2(P) - [t]\phi(P) + [q](P) = O$$

**Question:** Why not select a *suitable*  $P = (x, y) \in E(\overline{\mathbb{F}}_q)$  and plug into any of the above equations and determine  $t$ ?

$$\phi^2(P) + [q](P) = [t]\phi(P) \quad \mathbf{ECDLP} \text{ :-} ($$

Say  $P \in E[l]$ , is an  $l$ -torsion point, then  $\phi^2(P) - [t_l]\phi(P) + [q_l](P) = O$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\phi^2(P) + [q_l](P) = [\tau]\phi(P)$$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q)\end{aligned}$$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1.  $l \leftarrow 2$  and  $l_0 = 2$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1.  $l \leftarrow 2$  and  $l_0 = 2$
2. Find the smallest prime  $l_n$  greater than  $l_{n-1}$  and different from  $p$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1.  $l \leftarrow 2$  and  $l_0 = 2$
2. Find the smallest prime  $l_n$  greater than  $l_{n-1}$  and different from  $p$
3. Compute  $t_n \equiv t \pmod{l_n}$

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1.  $l \leftarrow 2$  and  $l_0 = 2$
2. Find the smallest prime  $l_n$  greater than  $l_{n-1}$  and different from  $p$
3. Compute  $t_n \equiv t \pmod{l_n}$
4. If  $\prod_{i=1}^n l_i < 4\sqrt{q}$ ; increment  $n$ , goto step 2

where  $t_l \equiv t \pmod{l}$ ,  $q_l \equiv q \pmod{l}$

And Hasse tells us that  $|t| \leq 2\sqrt{q}$ . So if we find  $t \pmod{l}$  for sufficiently many -  $O(\log q / \log \log q)$  primes  $l$  then using CRT we can recover  $t$ .

$$\begin{aligned}\phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Rightarrow (x^{q^2}, y^{q^2}) + [q_l](x, y) &= [\tau](x^q, y^q) \\ \text{where } \tau &\in \{0, 1, \dots, l-1\}\end{aligned}$$

**Schoof's algorithm:** [BSS], [KG]

INPUT: An elliptic curve  $E$  over a finite field  $\mathbb{F}_q$

OUTPUT: The order of  $E(\mathbb{F}_q)$

1.  $l \leftarrow 2$  and  $l_0 = 2$
2. Find the smallest prime  $l_n$  greater than  $l_{n-1}$  and different from  $p$
3. Compute  $t_n \equiv t \pmod{l_n}$
4. If  $\prod_{i=1}^n l_i < 4\sqrt{q}$ ; increment  $n$ , goto step 2
5. Solve the system of congruences  $t \equiv t_i \pmod{l_i}$  for  $1 \leq i \leq n$  to find  $t$ , and hence  $\#E(\mathbb{F}_q) = q + 1 - t$

So can we efficiently find  $\tau$  by trial and error?

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

By repeated application of the group law, we can express the multiplication by  $m$  by rational functions

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

By repeated application of the group law, we can express the multiplication by  $m$  by rational functions

**Thm [BSS]:** Let  $E$  be an elliptic curve defined over a field  $K$ , and let  $m$  be a positive integer. There exist polynomials  $\psi_m, \theta_m, \omega_m \in K[X, Y]$  such that, for  $P = (x, y) \in E(\overline{K})$  such that  $[m]P \neq O$ , we have

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

By repeated application of the group law, we can express the multiplication by  $m$  by rational functions

**Thm [BSS]:** Let  $E$  be an elliptic curve defined over a field  $K$ , and let  $m$  be a positive integer. There exist polynomials  $\psi_m, \theta_m, \omega_m \in K[X, Y]$  such that, for  $P = (x, y) \in E(\overline{K})$  such that  $[m]P \neq O$ , we have

$$[m]P = \left( \frac{\theta_m(x, y)}{\psi_m(x, y)^2}, \frac{\omega_m(x, y)}{\psi_m(x, y)^3} \right)$$

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

By repeated application of the group law, we can express the multiplication by  $m$  by rational functions

**Thm [BSS]:** Let  $E$  be an elliptic curve defined over a field  $K$ , and let  $m$  be a positive integer. There exist polynomials  $\psi_m, \theta_m, \omega_m \in K[X, Y]$  such that, for  $P = (x, y) \in E(\overline{K})$  such that  $[m]P \neq O$ , we have

$$[m]P = \left( \frac{\theta_m(x, y)}{\psi_m(x, y)^2}, \frac{\omega_m(x, y)}{\psi_m(x, y)^3} \right)$$

The polynomial  $\psi_m(x, y)$  is called the  $m$ th division polynomial of the curve  $E$ . Also the sequences  $\theta_m$  and  $\omega_m$  can be expressed in terms of the sequence  $\psi_m$

So can we efficiently find  $\tau$  by trial and error?

Schoof's idea so instead of computing with points explicitly...why not work with the group law.

By repeated application of the group law, we can express the multiplication by  $m$  by rational functions

**Thm [BSS]:** Let  $E$  be an elliptic curve defined over a field  $K$ , and let  $m$  be a positive integer. There exist polynomials  $\psi_m, \theta_m, \omega_m \in K[X, Y]$  such that, for  $P = (x, y) \in E(\overline{K})$  such that  $[m]P \neq O$ , we have

$$[m]P = \left( \frac{\theta_m(x, y)}{\psi_m(x, y)^2}, \frac{\omega_m(x, y)}{\psi_m(x, y)^3} \right)$$

The polynomial  $\psi_m(x, y)$  is called the  $m$ th division polynomial of the curve  $E$ . Also the sequences  $\theta_m$  and  $\omega_m$  can be expressed in terms of the sequence  $\psi_m$

**Thm:** Let  $P \in E(\overline{K}) - \{O\}$ , and let  $m \geq 1$ .  $P \in E[m] \Leftrightarrow \psi_m(P) = 0$

So for  $P = (x, y) \in E[l]$

$$\phi^2(P) + [q_l](P) = [\tau]\phi(P)$$

So for  $P = (x, y) \in E[l]$

$$\begin{aligned} \phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ \Downarrow \\ (X^{q^2}, Y^{q^2}) + [q_l](X, Y) &\equiv [\tau](X^q, Y^q) \end{aligned}$$

So for  $P = (x, y) \in E[l]$

$$\begin{aligned} \phi^2(P) + [ql](P) &= [\tau]\phi(P) \\ &\Downarrow \\ (X^{q^2}, Y^{q^2}) + [ql](X, Y) &\equiv [\tau](X^q, Y^q) \end{aligned}$$

in the ring  $\mathbb{F}_q[X, Y]$

So for  $P = (x, y) \in E[l]$

$$\begin{aligned} \phi^2(P) + [q_l](P) &= [\tau]\phi(P) \\ &\Downarrow \\ (X^{q^2}, Y^{q^2}) + [q_l](X, Y) &\equiv [\tau](X^q, Y^q) \end{aligned}$$

in the ring  $\mathbb{F}_q[X, Y] / \langle \psi_l(X) \rangle$ ,

So for  $P = (x, y) \in E[l]$

$$\begin{aligned} \phi^2(P) + [ql](P) &= [\tau]\phi(P) \\ &\Downarrow \\ (X^{q^2}, Y^{q^2}) + [ql](X, Y) &\equiv [\tau](X^q, Y^q) \end{aligned}$$

in the ring  $\mathbb{F}_q[X, Y] / \langle \psi_l(X), Y^2 - X^3 - AX - B \rangle$

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic.

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive.

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the the ring requires 1.5 MB.

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the the ring requires 1.5 MB.

Elkies' and Atkins' suggestions resulted in SEA algorithm.

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the the ring requires 1.5 MB.

Elkies' and Atkins' suggestions resulted in SEA algorithm. It uses  $l$ -division polynomials of degree  $O(l)$  found using modular polynomials

# Time and Space Complexity

[RS] Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the the ring requires 1.5 MB.

Elkies' and Atkins' suggestions resulted in SEA algorithm. It uses  $l$ -division polynomials of degree  $O(l)$  found using modular polynomials and runs in  $O(\log^{4+\varepsilon} q)$

# Time and Space Complexity

**[RS]** Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the the ring requires 1.5 MB.

Elkies' and Atkins' suggestions resulted in SEA algorithm. It uses  $l$ -division polynomials of degree  $O(l)$  found using modular polynomials and runs in  $O(\log^{4+\varepsilon} q)$

**Records [FGH]:** Morain in 1995  $q = 10^{499} + 153$ , Vercauteren in 1999  $q = 2^{1999}$

# Time and Space Complexity

**[RS]** Running time is  $O(\log^8 q)$  with naive arithmetic. Working with  $l$ -division polynomials of  $O(l^2)$  degree is expensive. With  $p \approx 10^{200}$ , representing one element in the ring requires 1.5 MB.

Elkies' and Atkins' suggestions resulted in SEA algorithm. It uses  $l$ -division polynomials of degree  $O(l)$  found using modular polynomials and runs in  $O(\log^{4+\varepsilon} q)$

**Records [FGH]:** Morain in 1995  $q = 10^{499} + 153$ , Vercauteren in 1999  $q = 2^{1999}$

... standby for part deux