Accelerating the Delfs–Galbraith Algorithm with Fast Subfield Root Detection

Maria Corte-Real Santos
University College London

Based on joint work with Craig Costello and Jia Shi

Isogeny-based Cryptography Workshop, Birmingham
March 18, 2022

Outline

Outline

The Supersingular Isogeny Problem

In its most general form, the *supersingular isogeny problem* asks to find an isogeny

$$\phi : E_1 \to E_2,$$

between two given supersingular elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$.

The Supersingular Isogeny Problem

In its most general form, the *supersingular isogeny problem* asks to find an isogeny

$$\phi : E_1 \to E_2,$$

between two given supersingular elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$.

The best known classical attack against this general problem is the **Delfs–Galbraith algorithm**.

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc.

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc. So, determining the concrete complexity of Delfs–Galbraith is important for the potential standardisation of these schemes.

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc. So, determining the concrete complexity of Delfs–Galbraith is important for the potential standardisation of these schemes.

Our contributions:

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc. So, determining the concrete complexity of Delfs–Galbraith is important for the potential standardisation of these schemes.

Our contributions:

- Provide an optimised implementation of the Delfs–Galbraith algorithm: Solver.

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc. So, determining the concrete complexity of Delfs–Galbraith is important for the potential standardisation of these schemes.

Our contributions:

- Provide an optimised implementation of the Delfs–Galbraith algorithm: Solver.
- Develop an efficient method to detect if a polynomial $f(X) \in \mathbb{F}_{p^d}[X]$ has a root in $\mathbb{F}_p$.

Motivation and Contributions

Difficulty of the supersingular isogeny problem affects the security of **B-SIDH**, **SQISign** (soundness), etc. So, determining the concrete complexity of Delfs–Galbraith is important for the potential standardisation of these schemes.

Our contributions:

- Provide an optimised implementation of the Delfs–Galbraith algorithm: Solver.
- Develop an efficient method to detect if a polynomial $f(X) \in \mathbb{F}_{p^d}[X]$ has a root in $\mathbb{F}_p$.
- Use this to introduce an improved attack, SuperSolver, with lower concrete complexity.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

Properties:

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

Properties:

- There are $\approx \frac{p}{12}$ vertices: this is the number of supersingular j-invariants (in $\mathbb{F}_{p^2}$).

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

Properties:

- There are $\approx \frac{p}{12}$ vertices: this is the number of supersingular j-invariants (in $\mathbb{F}_{p^2}$).
- $(\ell + 1)$-regular: one outgoing edge for each of the $\ell + 1$ cyclic subgroups of $E[\ell]$.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

Properties:

- There are $\approx \frac{p}{12}$ vertices: this is the number of supersingular j-invariants (in $\mathbb{F}_{p^2}$).
- $(\ell + 1)$-regular: one outgoing edge for each of the $\ell + 1$ cyclic subgroups of $E[\ell]$.
- Connected with diameter $O(\log p)$.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Let p be a large prime, $p \nmid \ell$.

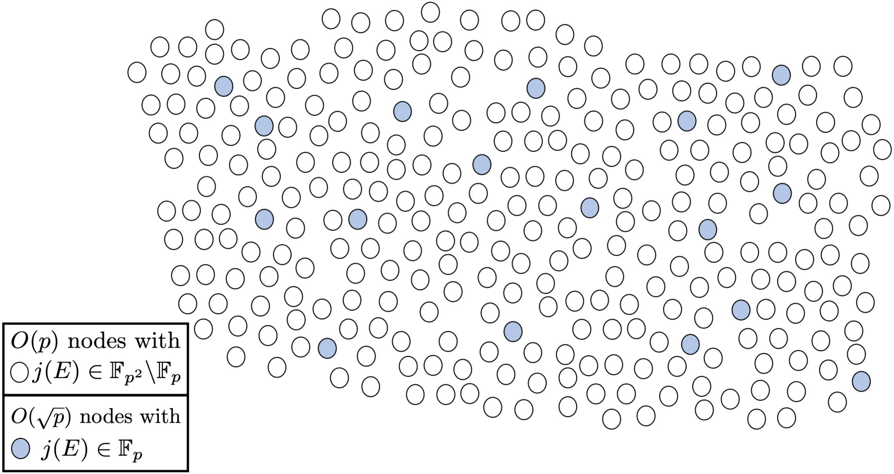
Vertices: $\bar{\mathbb{F}}_p$-isomorphism classes of supersingular elliptic curves E over $\bar{\mathbb{F}}_p$. These classes are represented by curves defined over $\mathbb{F}_{p^2}$ and are represented by a j-invariant in $\mathbb{F}_{p^2}$.

Edges: ℓ-isogenies defined over $\bar{\mathbb{F}}_p$.

Properties:

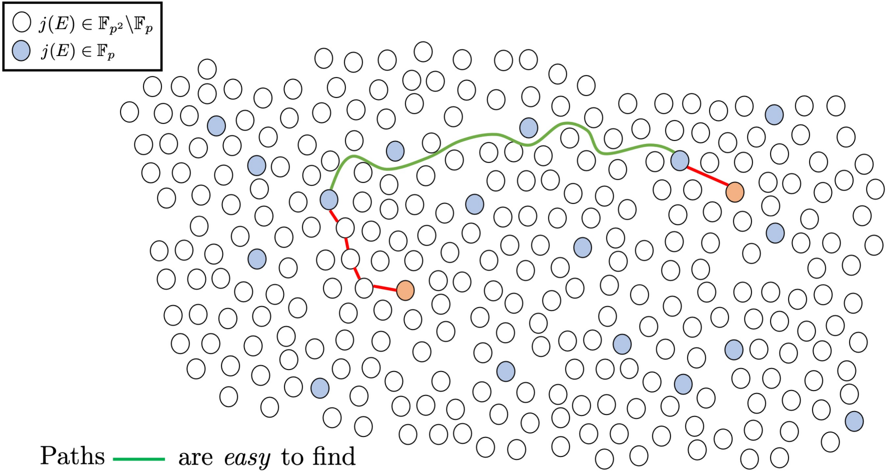
- There are $\approx \frac{p}{12}$ vertices: this is the number of supersingular j-invariants (in $\mathbb{F}_{p^2}$).
- $(\ell + 1)$-regular: one outgoing edge for each of the $\ell + 1$ cyclic subgroups of $E[\ell]$.
- Connected with diameter $O(\log p)$.
- Ramanujan graph: *rapid mixing*.

The Supersingular Isogeny Graph $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$



Outline

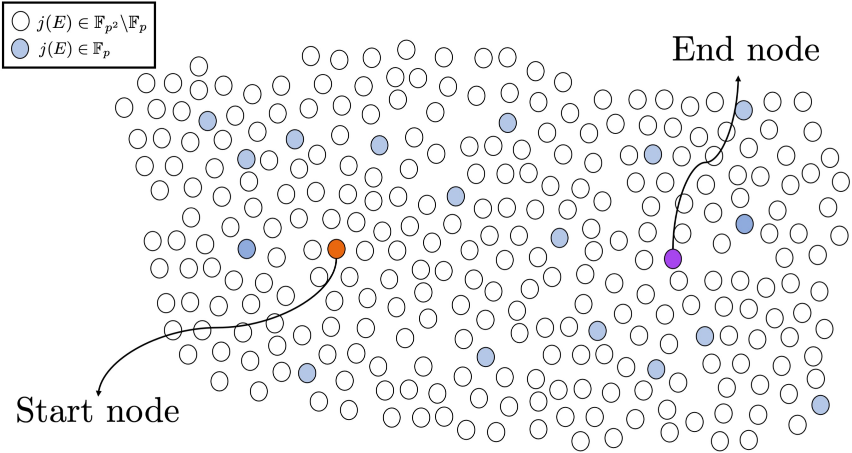
Key Observation



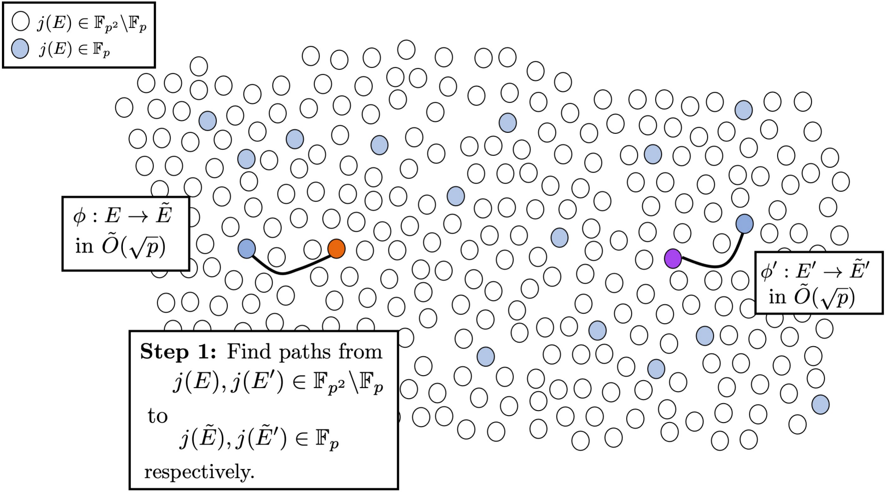
Paths —— are *easy* to find

Finding paths —— is the bottleneck

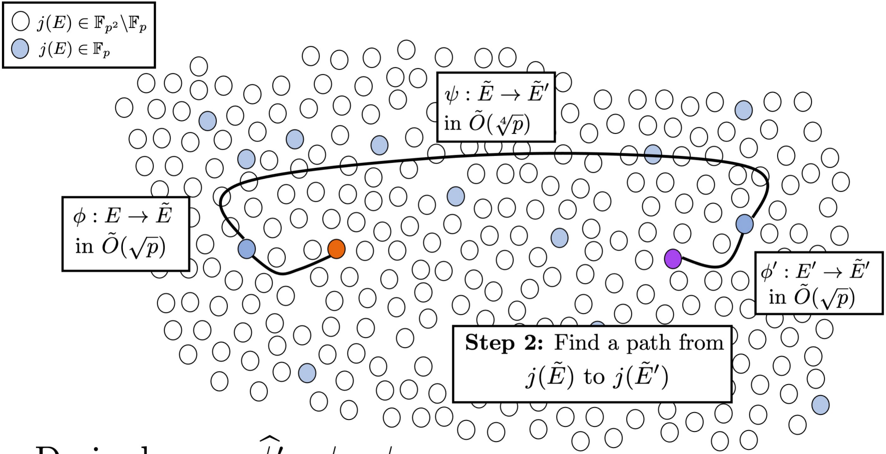
The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



Desired map: $\widehat{\phi'} \circ \psi \circ \phi$

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants.

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

- symmetric in X and Y

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

- symmetric in X and Y
- of degree N_ℓ in both X and Y, where

$$N_\ell := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } \ell.$$

$N_\ell = \ell + 1$ for ℓ prime.

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

- symmetric in X and Y
- of degree N_ℓ in both X and Y, where

$$N_\ell := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } \ell.$$

$N_\ell = \ell + 1$ for ℓ prime.

$\Phi_\ell(j_1, j_2) = 0 \iff j_1, j_2$ are j-invariants of ℓ-isogenous elliptic curves.

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

- symmetric in X and Y
- of degree N_ℓ in both X and Y, where

$$N_\ell := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } \ell.$$

$N_\ell = \ell + 1$ for ℓ prime.

$\Phi_\ell(j_1, j_2) = 0 \iff j_1, j_2$ are j-invariants of ℓ-isogenous elliptic curves.

This tells us that the roots of $\Phi_{\ell,p}(X, j)$ are neighbours of j in $\mathcal{X}(\mathbb{F}_p, \ell)$.

Modular Polynomial

The modular polynomial (of level ℓ) $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ parameterizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. It is:

- symmetric in X and Y
- of degree N_ℓ in both X and Y, where

$$N_\ell := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } \ell.$$

$N_\ell = \ell + 1$ for ℓ prime.

$\Phi_\ell(j_1, j_2) = 0 \iff j_1, j_2$ are j-invariants of ℓ-isogenous elliptic curves.

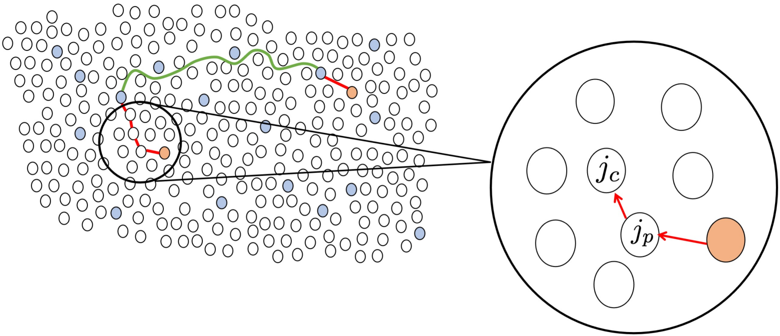
This tells us that the roots of $\Phi_{\ell,p}(X, j)$ are neighbours of j in $\mathcal{X}(\mathbb{F}_p, \ell)$. Reducing coefficients $\bmod p$ we can work with $\Phi_{\ell,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Taking a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Taking a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$:

Taking a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

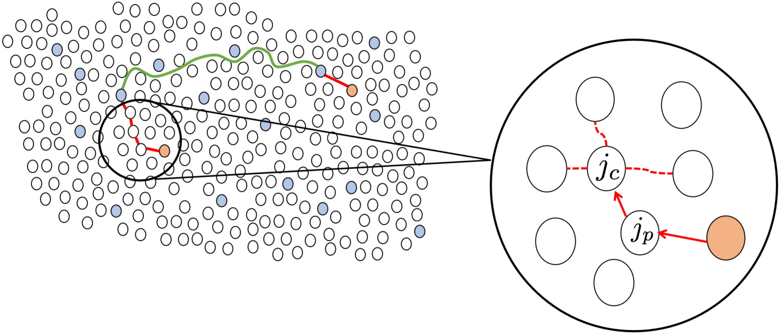
Taking a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$:



1. Store the current and previous j-invariants j_c and j_p.

Taking a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

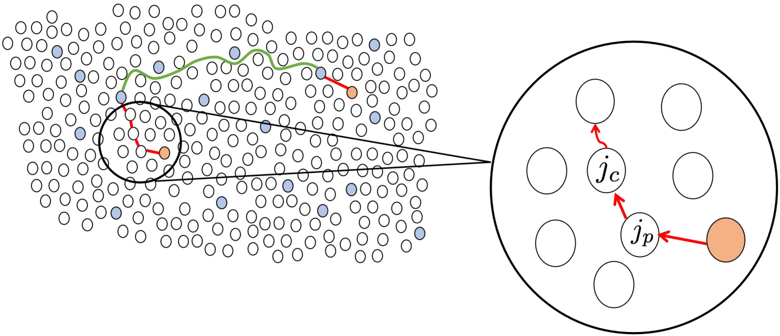
Taking a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$:



2. Find the $N_\ell - 1$ roots of $\Phi_{\ell,p}(X, j_c)/(X - j_p)$.

Taking a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$

Taking a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$:



3. Choose one of these and walk to the corresponding node.

Concrete Complexity of Delfs–Galbraith

Solver is an optimised implementation of the Delfs–Galbraith algorithm with $\ell = 2$.

> **Why $\ell = 2$?** Taking a step in $\mathcal{X}(\mathbb{F}_p, 2)$ means computing a square root.

Concrete Complexity of Delfs–Galbraith

Solver is an optimised implementation of the Delfs–Galbraith algorithm with $\ell = 2$.

Why $\ell = 2$? Taking a step in $\mathcal{X}(\mathbb{F}_p, 2)$ means computing a square root.

We use Solver to find the concrete complexity of Delfs–Galbraith.

Concrete Complexity of Delfs–Galbraith

Solver is an optimised implementation of the Delfs–Galbraith algorithm with $\ell = 2$.

> **Why $\ell = 2$?** Taking a step in $\mathcal{X}(\mathbb{F}_p, 2)$ means computing a square root.

We use Solver to find the concrete complexity of Delfs–Galbraith.

Experimentally, given a node $j \in \mathbb{F}_{p^2} \backslash \mathbb{F}_p$, the average number of $\mathbb{F}_p$ multiplications needed to find a path to a node $j' \in \mathbb{F}_p$ is

$$c \cdot \sqrt{p} \cdot \log_2 p,$$

with $0.75 \leq c \leq 1.05$.

Outline

Overview

SuperSolver is a **new attack** that improves on the *concrete* complexity of the Delfs–Galbraith algorithm.

Overview

SuperSolver is a **new attack** that improves on the *concrete* complexity of the Delfs–Galbraith algorithm. It changes the first step: the subfield search.

Overview

SuperSolver is a **new attack** that improves on the *concrete* complexity of the Delfs–Galbraith algorithm. It changes the first step: the subfield search.

At each step, we want to know if the current node j_c is ℓ-isogenous to a $j \in \mathbb{F}_p$.

Overview

SuperSolver is a **new attack** that improves on the *concrete* complexity of the Delfs–Galbraith algorithm. It changes the first step: the subfield search.

At each step, we want to know if the current node j_c is ℓ-isogenous to a $j \in \mathbb{F}_p$.

Key Observation

At each step, the precise values of the ℓ-isogenous neighbours do not need to be known, only whether it lies in $\mathbb{F}_p$.

Overview

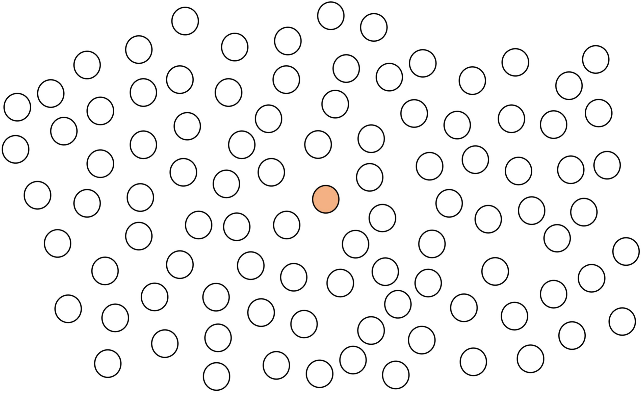
At each step of the random walk in $\mathcal{X}(\bar{\mathbb{F}}_p, 2)$, SuperSolver inspects the ℓ-isogeny graph with fast subfield root detection for ℓ in a carefully chosen set, to efficiently detect whether j_c has an ℓ-isogenous neighbour in $\mathbb{F}_p$.

Overview

At each step of the random walk in $\mathcal{X}(\bar{\mathbb{F}}_p, 2)$, SuperSolver inspects the ℓ-isogeny graph with **fast subfield root detection** for ℓ in a carefully chosen set, to efficiently detect whether j_c has an ℓ-isogenous neighbour in $\mathbb{F}_p$.

Overview

At each step of the random walk in $\mathcal{X}(\bar{\mathbb{F}}_p, 2)$, SuperSolver inspects the ℓ-isogeny graph with **fast subfield root detection** for ℓ in a **carefully chosen set**, to efficiently detect whether j_c has an ℓ-isogenous neighbour in $\mathbb{F}_p$.



Fast Subfield Root Detection

Recall to take a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ we find the roots of

$$\Phi_{\ell,p}(X, j_c) \in \mathbb{F}_{p^2}[X].$$

Fast Subfield Root Detection

Recall to take a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ we find the roots of

$$\Phi_{\ell,p}(X, j_c) \in \mathbb{F}_{p^2}[X].$$

We want a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

Fast Subfield Root Detection

Recall to take a step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ we find the roots of

$$\Phi_{\ell,p}(X, j_c) \in \mathbb{F}_{p^2}[X].$$

We want a fast way of detecting whether it has a root in $\mathbb{F}_p$ *without* finding roots.

Lemma

Let π be the p-power Frobenius map and f a polynomial in $\mathbb{F}_{p^2}[X]$. Then, $\gcd(f, \pi(f))$ is the largest divisor of f defined over $\mathbb{F}_p$.
In particular, if

$$\deg\left(\gcd(f, \pi(f))\right) = \begin{cases} 1, & f \text{ has a root in } \mathbb{F}_p \\ 0, & f \text{ does not have a root in } \mathbb{F}_p \end{cases}.$$

Fast Subfield Root Detection

Problem: In general $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ and we want to avoid costly multiplications in $\mathbb{F}_{p^2}$.

Fast Subfield Root Detection

Problem: In general $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ and we want to avoid costly multiplications in $\mathbb{F}_{p^2}$.

Observation

For polynomials $f_1, f_2 \in \mathbb{F}_{p^2}[X]$, if

$$g_1 = af_1 + bf_2, \text{ and } g_2 = cf_1 + df_2,$$

with $a, b, c, d \in \mathbb{F}_{p^2}$ such that $ad - bc \neq 0$ with we have

$$\gcd(f_1, f_2) = \gcd(g_1, g_2).$$

Fast Subfield Root Detection

Problem: In general $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ and we want to avoid costly multiplications in $\mathbb{F}_{p^2}$.

Solution: Let $\alpha \in \mathbb{F}_{p^2}$ be such that $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$. For $f(X) := \Phi_{\ell,p}(X, j_c)$, if

$$g_1 = \frac{1}{2}\Big(f + \pi(f)\Big), \text{ and } g_2 = \frac{\alpha}{2}\Big(f - \pi(f)\Big),$$

then $g_1, g_2 \in \mathbb{F}_p[X]$ and $\gcd(f, \pi(f)) = \gcd(g_1, g_2)$.

Fast Subfield Root Detection

Problem: In general $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ and we want to avoid costly multiplications in $\mathbb{F}_{p^2}$.

Solution: Let $\alpha \in \mathbb{F}_{p^2}$ be such that $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$. For $f(X) := \Phi_{\ell,p}(X, j_c)$, if

$$g_1 = \frac{1}{2}\Big(f + \pi(f)\Big), \text{ and } g_2 = \frac{\alpha}{2}\Big(f - \pi(f)\Big),$$

then $g_1, g_2 \in \mathbb{F}_p[X]$ and $\gcd(f, \pi(f)) = \gcd(g_1, g_2)$.

We can avoid **all** multiplications over $\mathbb{F}_{p^2}$: if we write the coefficients of $f(X)$ as $a_k^{(1)} + a_k^{(2)}\alpha$ (say $\alpha^2 = -1$), then

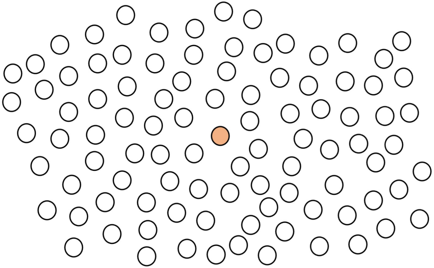
$$g_1(X) = \sum_{k=0}^{n} a_k^{(1)} X^k, \text{ and } g_2(X) = \sum_{k=0}^{n} a_k^{(2)} X^k.$$

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.



List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

1. Determine the cost per node revealed of taking a step in the 2-isogeny graph: cost_2

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

1. Determine the cost per node revealed of taking a step in the 2-isogeny graph: cost_2
2. Determine the cost per node inspected in the ℓ-isogeny graph: cost_ℓ.

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

1. Determine the cost per node revealed of taking a step in the 2-isogeny graph: cost_2
2. Determine the cost per node inspected in the ℓ-isogeny graph: cost_ℓ.
3. Determine a list $L = [\ell_1, \ldots, \ell_n]$ of $\ell_i > 2$ with $\text{cost}_\ell < \text{cost}_2$

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

1. Determine the cost per node revealed of taking a step in the 2-isogeny graph: cost_2
2. Determine the cost per node inspected in the ℓ-isogeny graph: cost_ℓ.
3. Determine a list $L = [\ell_1, \ldots, \ell_n]$ of $\ell_i > 2$ with $\text{cost}_\ell < \text{cost}_2$
4. Find the subset of L that minimises the total cost of each step:

$$\text{cost} = \frac{\text{total \# of } \mathbb{F}_p \text{ multiplications}}{\text{total \# of nodes revealed}}.$$

List of Optimal ℓ's

Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.

We want to compute a list of ℓ's that minimise $\#\mathbb{F}_p$ multiplications per node inspected.

1. Determine the cost per node revealed of taking a step in the 2-isogeny graph: cost_2
2. Determine the cost per node inspected in the ℓ-isogeny graph: cost_ℓ.
3. Determine a list $L = [\ell_1, \ldots, \ell_n]$ of $\ell_i > 2$ with $\text{cost}_\ell < \text{cost}_2$
4. Find the subset of L that minimises the total cost of each step:

$$\text{cost} = \frac{\text{total \# of } \mathbb{F}_p \text{ multiplications}}{\text{total \# of nodes revealed}}.$$

Calculating the list of optimal ℓ's can be done in precomputation.

Outline

Worked Example: Precomputation

Let $p = 2^{20} - 3$.

- Construct the extension field $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$, where α^2 is the first non-square in $-1, -2, 2, -3, 3, \dots$.

Worked Example: Precomputation

Let $p = 2^{20} - 3$.

- Construct the extension field $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$, where α^2 is the first non-square in $-1, -2, 2, -3, 3, \ldots$.
- Reduces the coefficients of $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ mod p to obtain $\Phi_{\ell,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

Worked Example: Precomputation

Let $p = 2^{20} - 3$.

- Construct the extension field $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$, where α^2 is the first non-square in $-1, -2, 2, -3, 3, \ldots$.
- Reduces the coefficients of $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ mod p to obtain $\Phi_{\ell,p}(X, Y) \in \mathbb{F}_p[X, Y]$.
- For SuperSolver, compute a list of optimal ℓ's L.

Worked Example: Precomputation

Let $p = 2^{20} - 3$.

- Construct the extension field $\mathbb{F}_{p^2} = \mathbb{F}_p(\alpha)$, where α^2 is the first non-square in $-1, -2, 2, -3, 3, \ldots$.
- Reduces the coefficients of $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ mod p to obtain $\Phi_{\ell,p}(X, Y) \in \mathbb{F}_p[X, Y]$.
- For SuperSolver, compute a list of optimal ℓ's L.

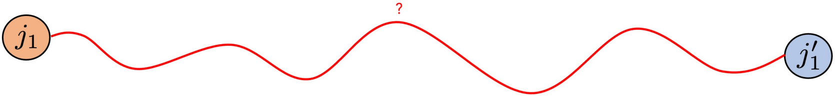
Sample our start and end node:

Start Node: $j_1 = 129007\alpha + 818380$

End Node: $j_2 = 97589\alpha + 660383$

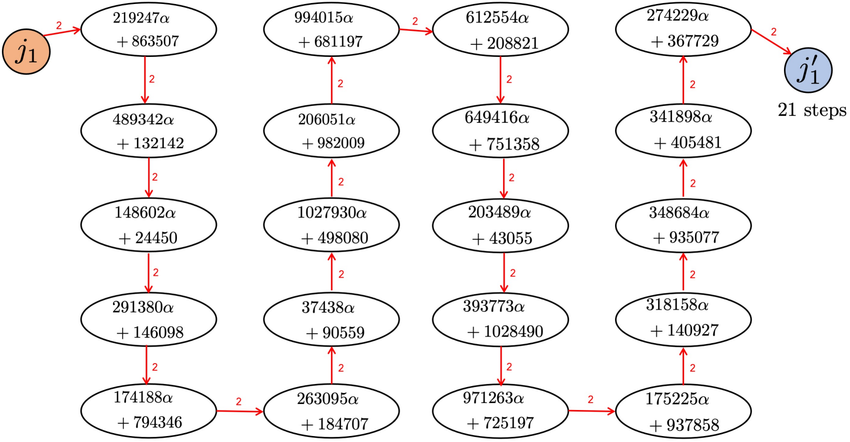
Worked Example: Solver

Path from $j_1 = 129007\alpha + 818380$ to subfield node.



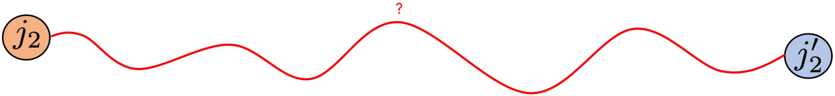
Worked Example: Solver

Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 760776$.



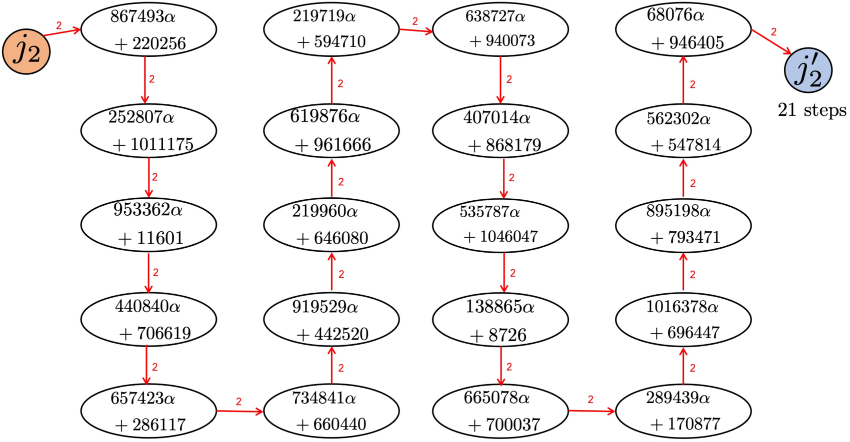
Worked Example: Solver

Path from $j_2 = 97589\alpha + 660383$ to subfield node.



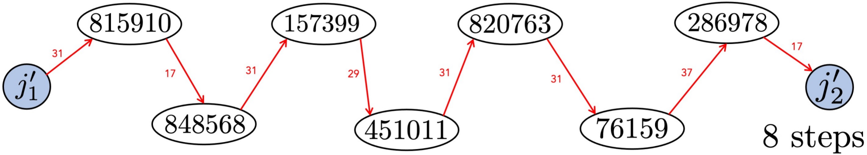
Worked Example: Solver

Path from $j_2 = 97589\alpha + 660383$ to subfield node $j_2' = 35387$.



Worked Example: Solver

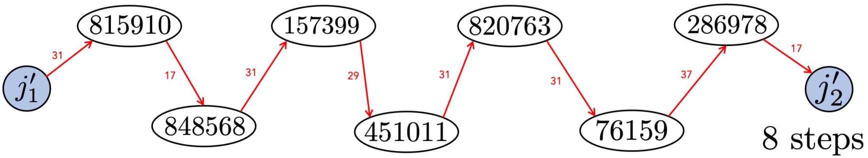
Path between subfield nodes $j'_1 = 760776$ and $j'_2 = 35387$.



We take steps in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ with $\ell \in \{17, 29, 31, 37\}$.

Worked Example: Solver

Path between subfield nodes $j_1' = 760776$ and $j_2' = 35387$.



We take steps in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ with $\ell \in \{17, 29, 31, 37\}$.

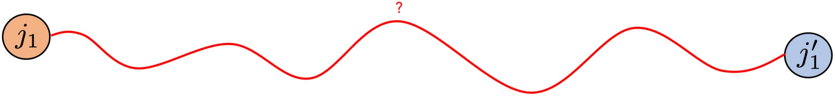
In total, the path has $21 + 21 + 8 =$ **50 steps**.

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.

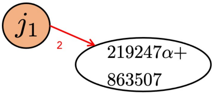
Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node.



Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.

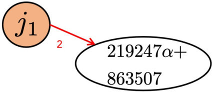


3-isogenous neighbour in $\mathbb{F}_p$?

$$\Phi_{3,p}(X, 219247\alpha + 863507) = X^4 + (212814\alpha + 479338)X^3 + (408250\alpha + 920025)X^2 + (811739\alpha + 93038)X + 942336\alpha + 847782$$

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.

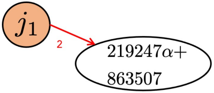


3-isogenous neighbour in $\mathbb{F}_p$?

$$g_1 = X^4 + 479338X^3 + 920025X^2 + 93038X + 847782$$
$$g_2 = 425628X^3 + 816500X^2 + 574905X + 836099$$

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



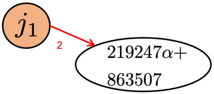
3-isogenous neighbour in $\mathbb{F}_p$?

$$g_1 = X^4 + 479338X^3 + 920025X^2 + 93038X + 847782$$
$$g_2 = 425628X^3 + 816500X^2 + 574905X + 836099$$

$$\boxed{\gcd(g_1, g_2) = 1 \implies \text{no 3-isogenous neighbour in } \mathbb{F}_p}$$

Worked Example: SuperSolver

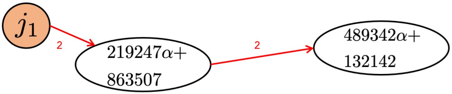
The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



3-isogenous neighbour in $\mathbb{F}_p$? No.
Similarly, no 5-isogenous neighbour in $\mathbb{F}_p$.

Worked Example: SuperSolver

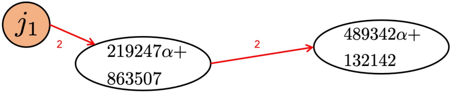
The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



3-isogenous neighbour in $\mathbb{F}_p$?

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.

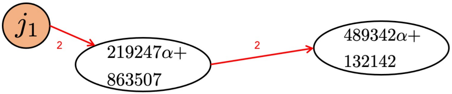


3-isogenous neighbour in $\mathbb{F}_p$?

$$\Phi_{3,p}(X, 489342\alpha + 132142) = X^4 + (872004\alpha + 13960)X^3 + (1031755\alpha + 822066)X^2 + (969683\alpha + 747785)X + 813010\alpha + 255391.$$

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



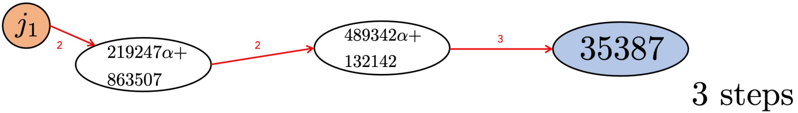
3-isogenous neighbour in $\mathbb{F}_p$?

$$g_1 = X^4 + 13960X^3 + 822066X^2 + 747785X + 255391$$
$$g_2 = 695435X^3 + 1014937X^2 + 890793X + 577447$$

$$\gcd(g_1, g_2) = X + 1013186 \implies \text{3-isogenous neighbour in } \mathbb{F}_p$$
$$-1013186 = 35387$$

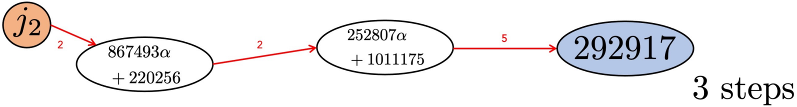
Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



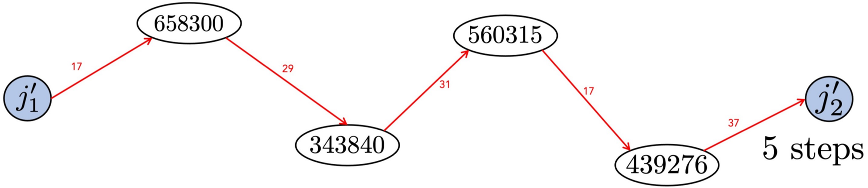
Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_2 = 97589\alpha + 660383$ to subfield node $j_2' = 292917$.



Worked Example: SuperSolver

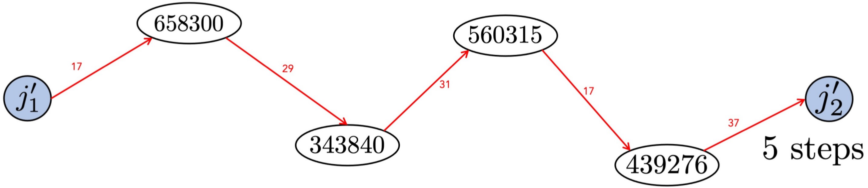
The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path between subfield nodes $j_1' = 35387$ and $j_2' = 292917$.



We take steps in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ with $\ell \in \{17, 29, 31, 37\}$.

Worked Example: SuperSolver

The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path between subfield nodes $j_1' = 35387$ and $j_2' = 292917$.



We take steps in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$ with $\ell \in \{17, 29, 31, 37\}$.

In total, the path has $3 + 3 + 5 =$ **11 steps**.

Outline

Results

Experiments on small primes and many j-invariants.

Results

Experiments on small primes and many j-invariants. SuperSolver finds a subfield node with much fewer (on average, half) $\mathbb{F}_p$ multiplications and by visiting less nodes.

Results

Experiments on small primes and many *j*-invariants. SuperSolver finds a subfield node with much fewer (on average, half) $\mathbb{F}_p$ multiplications and by visiting less nodes.

Example: For $p = 2^{24} - 3$, averaging over 5000 pseudo-random supersingular *j*-invarants in $\mathbb{F}_{p^2}$, we get:

Solver used *112878* $\mathbb{F}_p$ multiplications and walked on *1897* nodes.

SuperSolver used *53900* $\mathbb{F}_p$ multiplications and walked on *318* nodes.

Results

Experiments on small primes and many j-invariants. SuperSolver finds a subfield node with much fewer (on average, half) $\mathbb{F}_p$ multiplications and by visiting less nodes.

Experiments on cryptographic sized primes and one j-invariant. We ran SuperSolver and Solver until the number of $\mathbb{F}_p$ multiplications used exceeded 10^8, recording the total number of nodes covered.

Results

Experiments on small primes and many j-invariants. SuperSolver finds a subfield node with much fewer (on average, half) $\mathbb{F}_p$ multiplications and by visiting less nodes.

Experiments on cryptographic sized primes and one j-invariant. We ran SuperSolver and Solver until the number of $\mathbb{F}_p$ multiplications used exceeded 10^8, recording the total number of nodes covered.

Examples:

For $p = 2^{50} - 27$, SuperSolver covers between *3 and 4 times* the number of nodes that Solver does.

For $p = 2^{800} - 105$, SuperSolver covers between *18 and 19 times* the number of nodes.

Conclusions

What does this mean for isogeny-based cryptography?

- We improve the concrete complexity of Delfs–Galbraith - asymptotic complexity is unchanged.

Conclusions

What does this mean for isogeny-based cryptography?

- We improve the concrete complexity of Delfs–Galbraith - asymptotic complexity is unchanged.
- No direct impact on SIDH and SIKE - there are faster claw-finding algorithms.

Conclusions

What does this mean for isogeny-based cryptography?

- We improve the concrete complexity of Delfs–Galbraith - asymptotic complexity is unchanged.
- No direct impact on SIDH and SIKE - there are faster claw-finding algorithms.
- Affects other proposals, such as B-SIDH and SQISign, with Delfs–Galbraith as their best attack.

Open Problems

Relating to SuperSolver:

Open Problems

Relating to SuperSolver:

- Can we combine $\Phi_m(X, j)$ and $\Phi_n(X, j)$ so that we can detect whether j has an nm-isogenous neighbour doing operations with Φ_m and Φ_n only?

Open Problems

Relating to SuperSolver:

- Can we combine $\Phi_m(X, j)$ and $\Phi_n(X, j)$ so that we can detect whether j has an *nm*-isogenous neighbour doing operations with Φ_m and Φ_n only?
- What does a *quantum version* of SuperSolver look like?

Open Problems

Relating to SuperSolver:

- Can we combine $\Phi_m(X, j)$ and $\Phi_n(X, j)$ so that we can detect whether j has an nm-isogenous neighbour doing operations with Φ_m and Φ_n only?
- What does a *quantum version* of SuperSolver look like?
- Other applications of subfield detection