Accelerating the Delfs–Galbraith Algorithm with Fast Subfield Root Detection

Maria Corte-Real Santos
University College London
www.mariascrs.com

Based on joint work with Craig Costello and Jia Shi

CRYPTO 2022

Motivation

- The supersingular isogeny problem is the foundational problem in isogeny-based cryptography, conjectured to be *post-quantum secure*.

Motivation

- The supersingular isogeny problem is the foundational problem in isogeny-based cryptography, conjectured to be *post-quantum secure*.
- The security of **SQISign** (signature scheme) and **B-SIDH** (key exchange) relies on its difficulty.

Motivation

- The supersingular isogeny problem is the foundational problem in isogeny-based cryptography, conjectured to be *post-quantum secure*.
- The security of **SQISign** (signature scheme) and **B-SIDH** (key exchange) relies on its difficulty.
- The best known classical attack against this general problem is the **Delfs–Galbraith algorithm**.

Contributions

Our contributions:

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

Contributions

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 $\mathbb{F}_p$.

Contributions

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.

Elliptic Curves and j-invariants

An elliptic curve E over $\mathbb{F}_{p^2}$ ($p \neq 2, 3$) is a smooth curve given by the equation

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_{p^2}$ and $4a^3 + 27b^2 \neq 0$.

Elliptic Curves and j-invariants

An elliptic curve E over $\mathbb{F}_{p^2}$ ($p \neq 2, 3$) is a smooth curve given by the equation

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_{p^2}$ and $4a^3 + 27b^2 \neq 0$. We consider *supersingular* elliptic curves.

Elliptic Curves and j-invariants

An elliptic curve E over $\mathbb{F}_{p^2}$ ($p \neq 2, 3$) is a smooth curve given by the equation

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_{p^2}$ and $4a^3 + 27b^2 \neq 0$. We consider *supersingular* elliptic curves.

We label our elliptic curves using *j-invariants*.

Elliptic Curves and j-invariants

An elliptic curve E over $\mathbb{F}_{p^2}$ ($p \neq 2, 3$) is a smooth curve given by the equation

$$E : y^2 = x^3 + ax + b,$$

where $a, b \in \mathbb{F}_{p^2}$ and $4a^3 + 27b^2 \neq 0$. We consider *supersingular* elliptic curves.

We label our elliptic curves using *j-invariants*.

Definition

Let $E : y^2 = x^3 + ax + b$ supersingular. Then, the j-invariant of E is

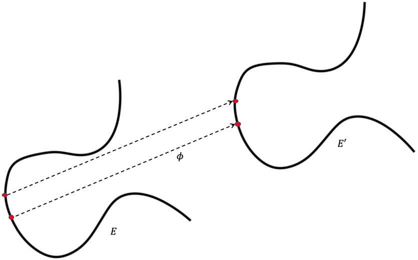
$$j(E) = 1728 \frac{4a^3}{4a^3 + 27b^2} \in \mathbb{F}_{p^2}.$$

Isogenies

Definition

Let E and E' be two elliptic curves, and let $\phi : E \rightarrow E'$ be a map between them. Then, ϕ is an *isogeny* if it is non-constant and $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$.

Isogenies are *group homomorphisms*, meaning that for $P, Q \in E$ we have $\phi(P \oplus_E Q) = \phi(P) \oplus_{E'} \phi(Q)$.

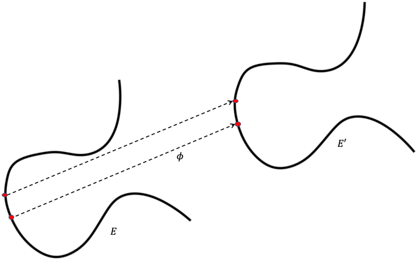


Isogenies

Definition

Let E and E' be two elliptic curves, and let $\phi : E \to E'$ be a map between them. Then, ϕ is an *isogeny* if it is non-constant and $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$.

Isogenies are *group homomorphisms*, meaning that for $P, Q \in E$ we have $\phi(P \oplus_E Q) = \phi(P) \oplus_{E'} \phi(Q)$.



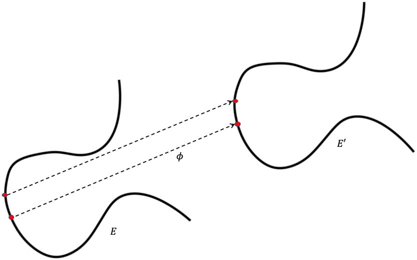
For (seperable) isogenies, the degree $\deg(\phi) = \#\ker(\phi)$.

Isogenies

Definition

Let E and E' be two elliptic curves, and let $\phi : E \to E'$ be a map between them. Then, ϕ is an *isogeny* if it is non-constant and $\phi(\mathcal{O}_E) = \mathcal{O}_{E'}$.

Isogenies are *group homomorphisms*, meaning that for $P, Q \in E$ we have $\phi(P \oplus_E Q) = \phi(P) \oplus_{E'} \phi(Q)$.



For (seperable) isogenies, the degree $\deg(\phi) = \#\ker(\phi)$. We call an isogeny of degree ℓ an ℓ-isogeny.

The Supersingular Isogeny Problem

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

$$\phi : E_1 \longrightarrow E_2,$$

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

$$\phi : E_1 \longrightarrow E_2,$$

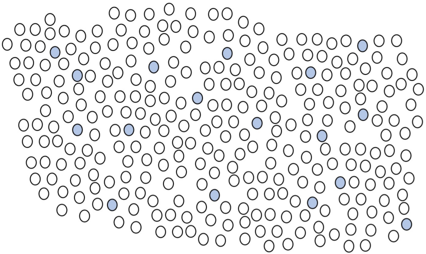
between two given supersingular elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$.

We do not assume:

- torsion point information
- degree of the isogeny
- starting curve is of a special form

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

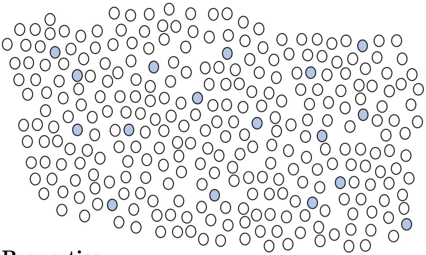
Let p be a large prime, $p \nmid \ell$.



Vertices: Isomorphism classes of supersingular elliptic curves E represented by a j-invariant in $\mathbb{F}_{p^2}$.
Edges: ℓ-isogenies

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

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



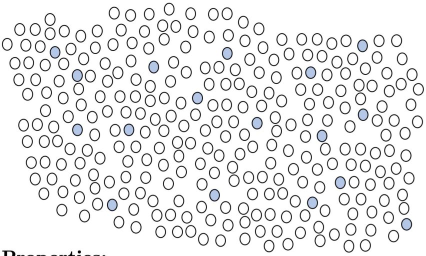
Vertices: Isomorphism classes of supersingular elliptic curves E represented by a j-invariant in $\mathbb{F}_{p^2}$.
Edges: ℓ-isogenies

Properties:

- A random walk of $\log(p)$ steps is almost as good as uniformly sampling vertices (Expander property)
- Path finding conjectured to be hard for classical and quantum computers

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

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



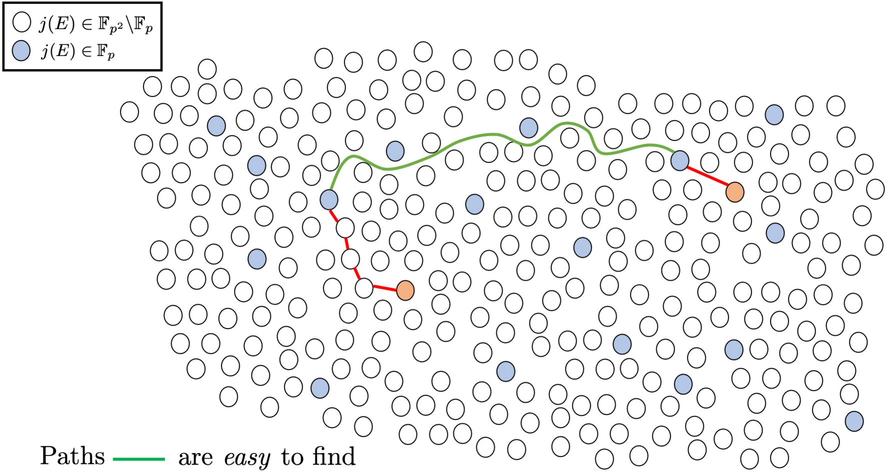
Vertices: Isomorphism classes of supersingular elliptic curves E represented by a j-invariant in $\mathbb{F}_{p^2}$.
Edges: ℓ-isogenies

Properties:

- A random walk of $\log(p)$ steps is almost as good as uniformly sampling vertices (Expander property)
- Path finding conjectured to be hard for classical and quantum computers

Finding a path between two nodes j_1, j_2 = Finding an isogeny between E_1, E_2

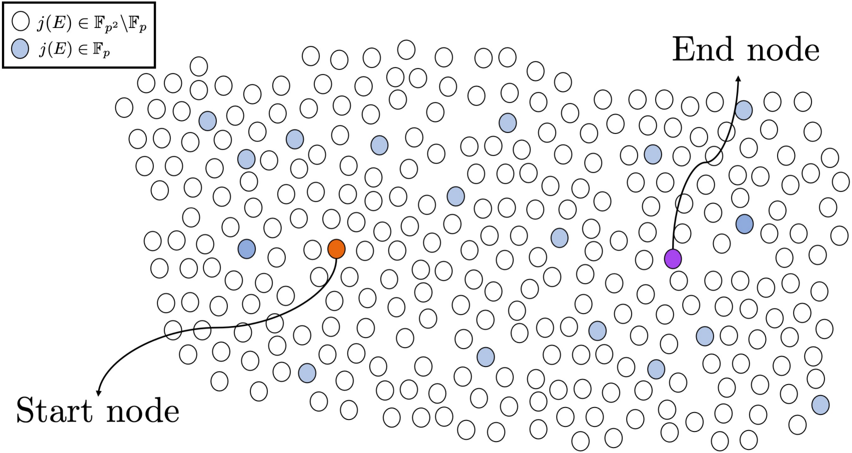
Key Observation



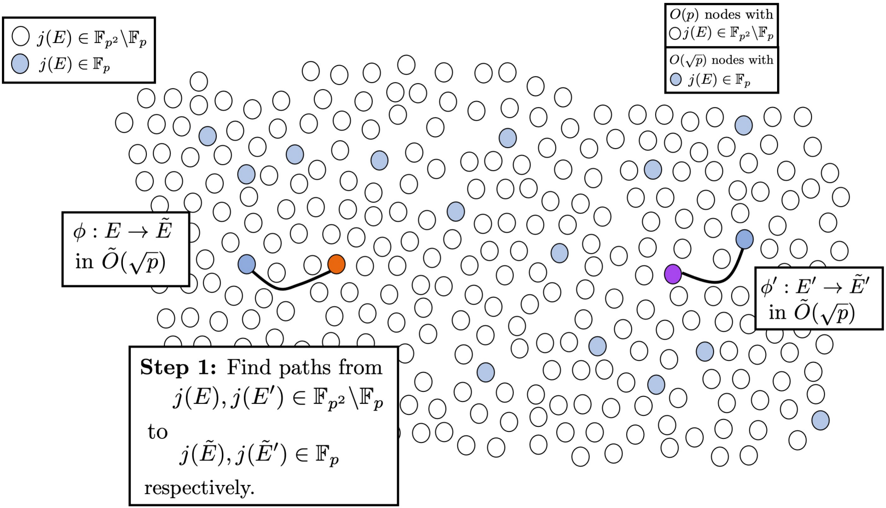
Paths —— are *easy* to find

Finding paths —— is the bottleneck

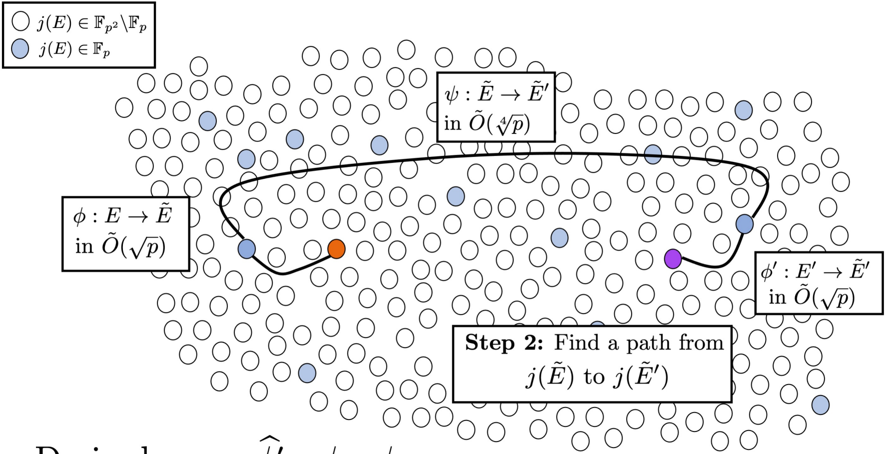
The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



The Delfs–Galbraith Algorithm



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

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

To take a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$, we use *modular polynomials*.

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

To take a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$, we use *modular polynomials*.
The modular polynomial $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ is symmetric 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.

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

To take a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$, we use *modular polynomials*. The modular polynomial $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ is symmetric 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.

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

To take a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$, we use *modular polynomials*.
The modular polynomial $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ is symmetric 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 $\Phi_\ell(X, j)$ are neighbours of j in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$.

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

To take a self-avoiding step in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$, we use *modular polynomials*. The modular polynomial $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ is symmetric of degree N_ℓ in both X and Y where

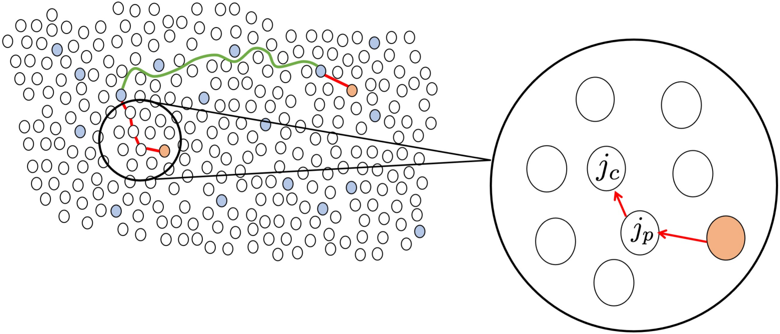
$$N_\ell := \prod_{i=1}^{n} (\ell_i + 1)\ell_i^{e_i - 1}, \text{ for prime decomposition } \prod_{i=1}^{n} \ell_i^{e_i} \text{ of } \ell.$$

$N_\ell = \ell + 1$ for ℓ prime.

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

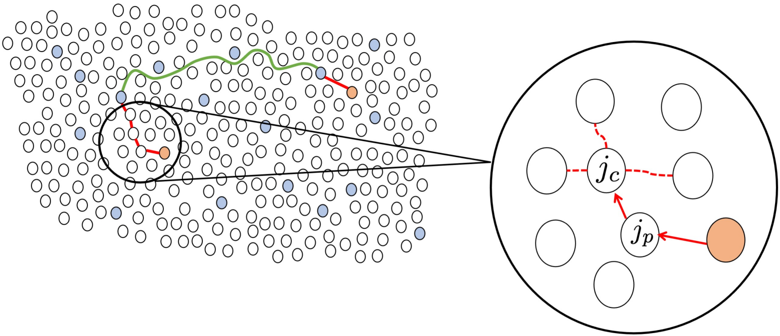
This tells us that the roots of $\Phi_\ell(X, j)$ are neighbours of j in $\mathcal{X}(\bar{\mathbb{F}}_p, \ell)$. Reducing coefficients $\bmod p$ we can work with $\Phi_{\ell,p}(X, Y) \in \mathbb{F}_p[X, Y]$.

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



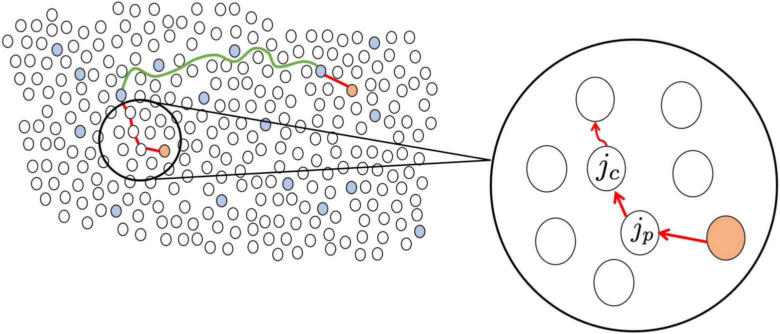
1. Store the current and previous j-invariants j_c and j_p.

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



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

Concrete Complexity of Delfs–Galbraith

We use our optimised implementation of the Delfs–Galbraith algorithm, Solver, to determine the concrete complexity of the first (bottleneck) step.

Concrete Complexity of Delfs–Galbraith

We use our optimised implementation of the Delfs–Galbraith algorithm, Solver, to determine the concrete complexity of the first (bottleneck) step.

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 $0.75 \leq c \leq 1.05$.

SuperSolver Overview

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

SuperSolver 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.

SuperSolver 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.

Key Observation

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

SuperSolver 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.

Key Observation

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

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

SuperSolver Overview

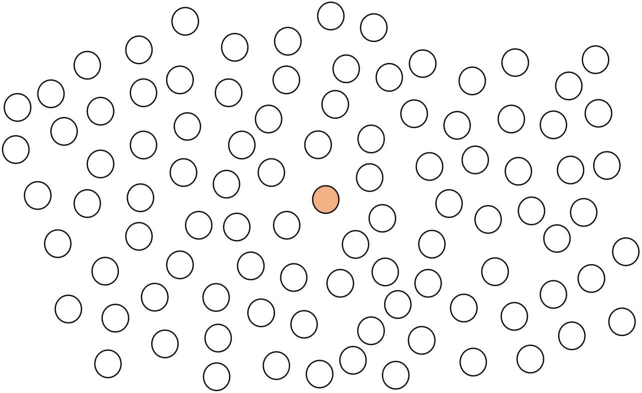
At each step of the random walk in $\mathcal{X}(\bar{\mathbb{F}}_p, 2)$, SuperSolver inspects the ℓ-isogeny graph with fast subfield root detection for ℓ in a carefully chosen set, to efficiently detect whether j_c has an ℓ-isogenous neighbour in $\mathbb{F}_p$.

SuperSolver 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 $\mathbb{F}_p$.

SuperSolver 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

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

Fast Subfield Root Detection

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

Lemma

Let $\pi : a \mapsto a^p$ be the p-power Frobenius map and f a polynomial in $\mathbb{F}_{p^2}[X]$.

Fast Subfield Root Detection

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

Lemma

Let $\pi : a \mapsto a^p$ be the p-power Frobenius map and f a polynomial in $\mathbb{F}_{p^2}[X]$.

- If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.

Fast Subfield Root Detection

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

Lemma

Let $\pi : a \mapsto a^p$ be the p-power Frobenius map and f a polynomial in $\mathbb{F}_{p^2}[X]$.

- If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.
- If $\deg\big(\gcd(f, \pi(f))\big) = 0$, f does not have a root in $\mathbb{F}_p$.

Fast Subfield Root Detection

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

Lemma

Let $\pi : a \mapsto a^p$ be the p-power Frobenius map and f a polynomial in $\mathbb{F}_{p^2}[X]$.

- If $\deg\big(\gcd(f, \pi(f))\big) = 1$, f has a root in $\mathbb{F}_p$.
- If $\deg\big(\gcd(f, \pi(f))\big) = 0$, f does not have a root in $\mathbb{F}_p$.

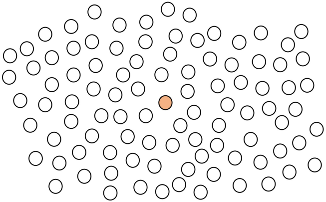
We also show how to transform $f, \pi(f) \in \mathbb{F}_{p^2}[X]$ to give $g_1, g_2 \in \mathbb{F}_p[X]$ with the same gcd and avoid *all* costly multiplications in $\mathbb{F}_{p^2}$.

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.

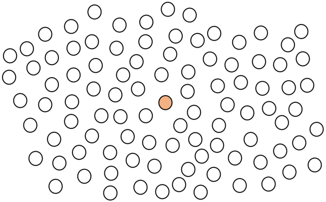
List of Optimal ℓ's

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

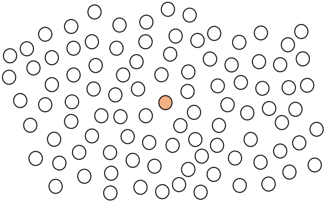
Though the inspection of the neighbours of j_c in the ℓ-isogeny graph increases the total number of $\mathbb{F}_p$ multiplications at each step, more nodes are checked.



We compute the list of optimal ℓ's to minimise the number of $\mathbb{F}_p$ multiplications per node 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 compute the list of optimal ℓ's to minimise the number of $\mathbb{F}_p$ multiplications per node revealed. The key is that calculating the list of optimal ℓ's can be done in *precomputation*.

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.
- Affects schemes such as B-SIDH and SQISign, which have Delfs–Galbraith as their best attack.

For more details, see our full paper at eprint/2021/1488

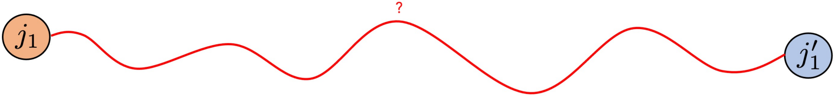
Additional Slides

Worked Example

Let $p = 2^{20} - 3$. Sample our start and end node:

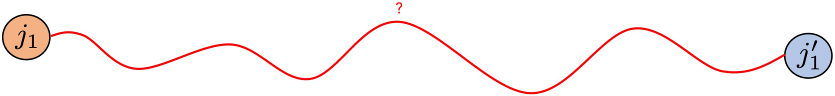
Start Node: $j_1 = 129007\alpha + 818380$

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



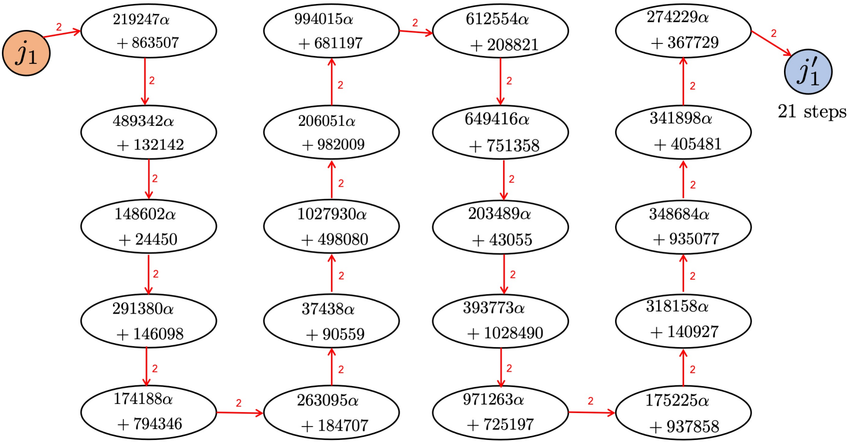
Worked Example: Solver

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



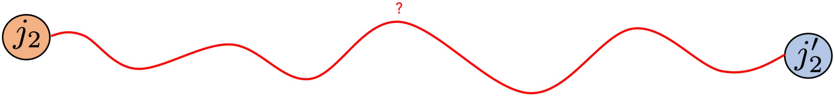
Worked Example: Solver

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



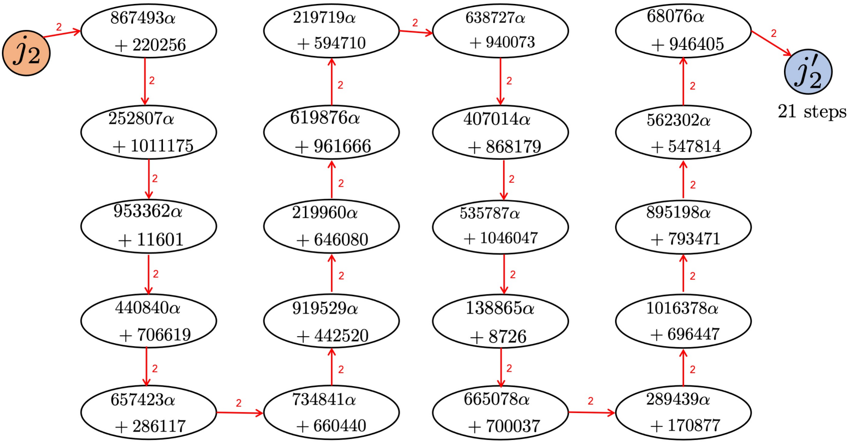
Worked Example: Solver

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



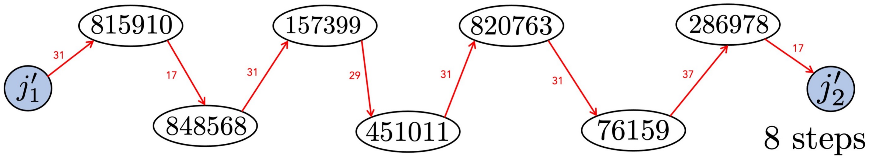
Worked Example: Solver

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



Worked Example: Solver

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



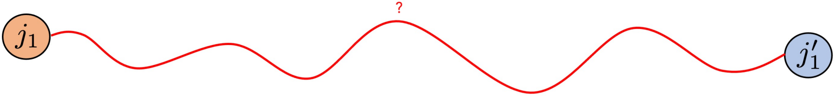
In total, the path has $21 + 21 + 8 =$ **50 steps**.

Worked Example: SuperSolver

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

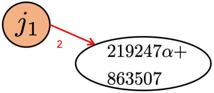
Worked Example: SuperSolver

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 $j_1' = 35387$.

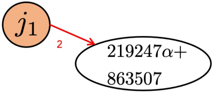


3-isogenous neighbour in $\mathbb{F}_p$?

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

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 $j'_1 = 35387$.

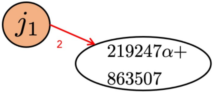


3-isogenous neighbour in $\mathbb{F}_p$?

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

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 $j_1' = 35387$.



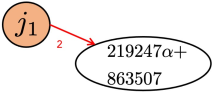
3-isogenous neighbour in $\mathbb{F}_p$?

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

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

Worked Example: SuperSolver

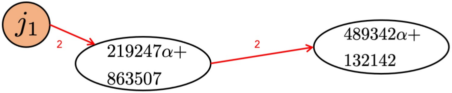
The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



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

Worked Example: SuperSolver

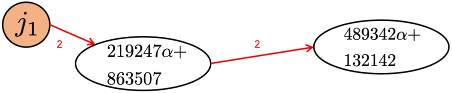
The list of optimal ℓ's is precomputed as $L = \{3, 5\}$.
Path from $j_1 = 129007\alpha + 818380$ to subfield node $j_1' = 35387$.



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 $j_1' = 35387$.

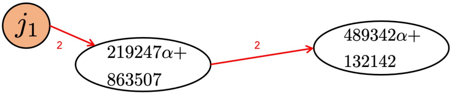


3-isogenous neighbour in $\mathbb{F}_p$?

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

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 $j_1' = 35387$.



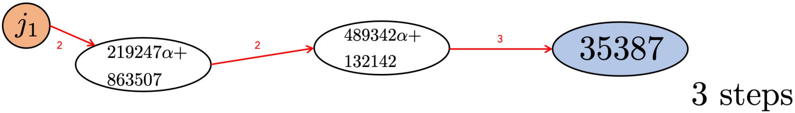
3-isogenous neighbour in $\mathbb{F}_p$?

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

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

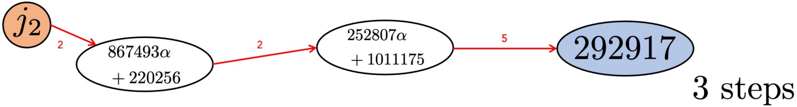
Worked Example: SuperSolver

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



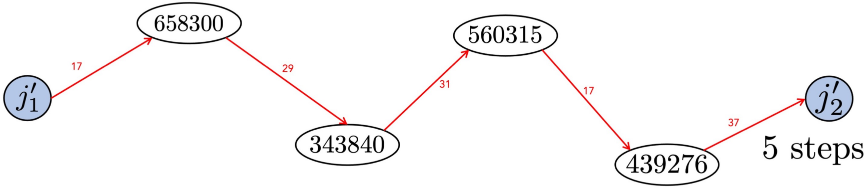
Worked Example: SuperSolver

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 $j_2' = 292917$.



In total, the path has $3 + 3 + 5 = \mathbf{11}$ **steps**.

Concrete Complexity of Delfs–Galbraith

Solver is an optimised implementation of the Delfs–Galbraith algorithm.

- **Choice of $\ell = 2$:** Taking a step in $\mathcal{X}(\mathbb{F}_p, 2)$ means computing a square root.
- **Square root finding in $\mathbb{F}_{p^2}$:** Use Scott's 'Tricks of the trade' paper to find square roots in $\mathbb{F}_{p^2}$ with only two $\mathbb{F}_p$ exponentiations (and a few $\mathbb{F}_p$ multiplications).
- **Random walks in 2-isogeny graph**: Depth-first search with bounded depth.

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 $0.75 \leq c \leq 1.05$.

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 $\pi : a \mapsto a^p$ 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

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

Fast Subfield Root Detection

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

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

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 $\gcd(f, \pi(f)) = \gcd(g_1, g_2)$.

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 compute the 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:

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

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