Efficient Detection of (N, N)-Splittings

Maria Corte-Real Santos
University College London

Based on joint work with Craig Costello and Sam Frengley

ISOCRYPT, KU Leuven
September 23, 2022

Outline

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\bar{\mathbb{F}}_p$-isomorphism and label these classes with:

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\bar{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Pairs of j-invariants $\left(j(E), j(E')\right)$

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\bar{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Pairs of j-invariants $\Big(j(E), j(E')\Big)$
2. Igusa–Clebsch invariants $I_2(C), I_4(C), I_6(C), I_{10}(C)$ (subscript denotes the weight of the invariant).

Abelian Surfaces

To generalise supersingular elliptic curves over $\bar{\mathbb{F}}_p$ ($p > 3$) to genus 2, we consider *superspecial (principally polarised) abelian surfaces* over $\bar{\mathbb{F}}_p$.

There are two types:

1. Products of supersingular elliptic curves $E \times E'$
2. Jacobians $\mathrm{Jac}(C)$ of genus 2 curves C

We consider them up to $\bar{\mathbb{F}}_p$-isomorphism and label these classes with:

1. Pairs of j-invariants $\Big(j(E), j(E')\Big)$
2. Igusa–Clebsch invariants $I_2(C), I_4(C), I_6(C), I_{10}(C)$ (subscript denotes the weight of the invariant).

For superspecial (p.p.) abelian surfaces, these invariants lie in $\mathbb{F}_{p^2}$.

(N, N)-Isogenies

An (N, N)-isogeny is an isogeny[1] $\phi\colon A \longrightarrow A'$, between p.p. abelian surfaces A, A' where:

- $\ker \phi \cong (\mathbb{Z}/N\mathbb{Z})^2$; and
- the isogeny respects the polarisations.

[1] i.e., surjective group homomorphism with finite kernel

General Isogeny Problem in Two Dimensions

In its most general form, the superspecial isogeny problem in two dimensions asks to find an isogeny

$$\phi\colon A \longrightarrow A',$$

between two superspecial (p.p.) abelian surfaces $A/\mathbb{F}_{p^2}$ and $A'/\mathbb{F}_{p^2}$.

General Isogeny Problem in Two Dimensions

In its most general form, the superspecial isogeny problem in two dimensions asks to find an isogeny

$$\phi\colon A \longrightarrow A',$$

between two superspecial (p.p.) abelian surfaces $A/\mathbb{F}_{p^2}$ and $A'/\mathbb{F}_{p^2}$.

The general isogeny problem can be viewed as finding a path between two nodes in the superspecial isogeny graph.

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

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

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

Let p be a large prime, $p \nmid N$. $\Gamma(N; \bar{\mathbb{F}}_p)$ is the graph with vertex set

$$\mathcal{S}(p) = \{\bar{\mathbb{F}}_p\text{-isomorphism classes of superspecial p.p. abelian surfaces}\},$$

and whose edges are (N, N)-isogenies over $\bar{\mathbb{F}}_p$.

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

Let p be a large prime, $p \nmid N$. $\Gamma(N; \bar{\mathbb{F}}_p)$ is the graph with vertex set

$$\mathcal{S}(p) = \{\bar{\mathbb{F}}_p\text{-isomorphism classes of superspecial p.p. abelian surfaces}\},$$

and whose edges are (N, N)-isogenies over $\bar{\mathbb{F}}_p$.

Properties:

- $\#\mathcal{S}(p) = O(p^3)$

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

Let p be a large prime, $p \nmid N$. $\Gamma(N; \bar{\mathbb{F}}_p)$ is the graph with vertex set

$$\mathcal{S}(p) = \{\bar{\mathbb{F}}_p\text{-isomorphism classes of superspecial p.p. abelian surfaces}\},$$

and whose edges are (N, N)-isogenies over $\bar{\mathbb{F}}_p$.

Properties:

- $\#\mathcal{S}(p) = O(p^3)$
- Classes $[A] \in \mathcal{S}(p)$ are represented by surfaces defined over $\mathbb{F}_{p^2}$.

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

Let p be a large prime, $p \nmid N$. $\Gamma(N; \bar{\mathbb{F}}_p)$ is the graph with vertex set

$$\mathcal{S}(p) = \{\bar{\mathbb{F}}_p\text{-isomorphism classes of superspecial p.p. abelian surfaces}\},$$

and whose edges are (N, N)-isogenies over $\bar{\mathbb{F}}_p$.

Properties:

- $\#\mathcal{S}(p) = O(p^3)$
- Classes $[A] \in \mathcal{S}(p)$ are represented by surfaces defined over $\mathbb{F}_{p^2}$.
- The graph is D_N-regular, where

$$D_N = N^3 \prod_{\substack{\text{primes} \\ \ell | N}} \frac{1}{\ell^3}(\ell + 1)(\ell^2 + 1).$$

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

Let p be a large prime, $p \nmid N$. $\Gamma(N; \bar{\mathbb{F}}_p)$ is the graph with vertex set

$$\mathcal{S}(p) = \{\bar{\mathbb{F}}_p\text{-isomorphism classes of superspecial p.p. abelian surfaces}\},$$

and whose edges are (N, N)-isogenies over $\bar{\mathbb{F}}_p$.

Properties:

- $\#\mathcal{S}(p) = O(p^3)$
- Classes $[A] \in \mathcal{S}(p)$ are represented by surfaces defined over $\mathbb{F}_{p^2}$.
- The graph is D_N-regular, where

$$D_N = N^3 \prod_{\substack{\text{primes} \\ \ell \mid N}} \frac{1}{\ell^3}(\ell + 1)(\ell^2 + 1).$$

- No analogy of Pizer's theorem - we work off the hypothesis that $\Gamma(N; \bar{\mathbb{F}}_p)$ is Ramanujan

The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

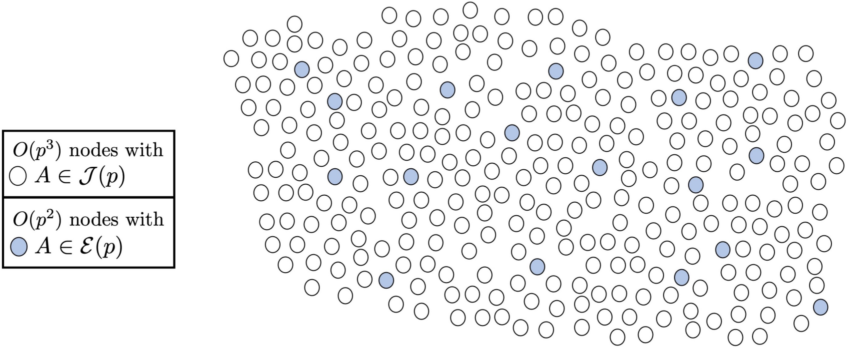
$\mathcal{S}(p)$ is equal to the disjoint union of:

$$\mathcal{J}(p) := \{[A] \in \mathcal{S}(p) \;:\; A \cong \mathrm{Jac}(C)\} \text{ and}$$
$$\mathcal{E}(p) := \{[A] \in \mathcal{S}(p) \;:\; A \cong E \times E' \text{ with } E,\ E' \text{ supersingular ECs}\}.$$

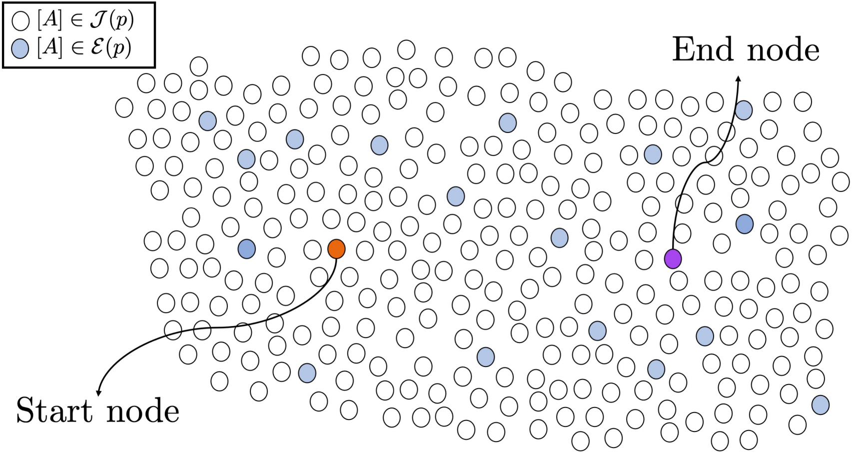
The Superspecial Isogeny Graph $\Gamma(N; \bar{\mathbb{F}}_p)$

$\mathcal{S}(p)$ is equal to the disjoint union of:

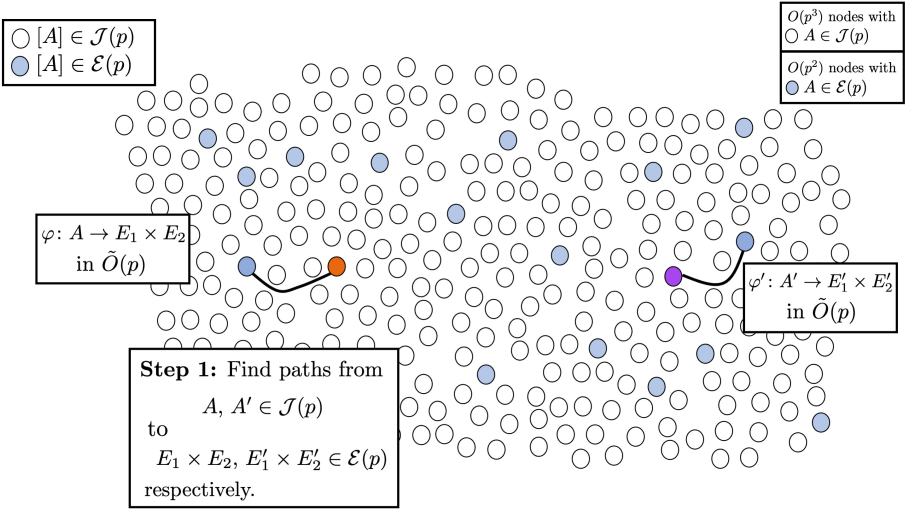
$$\mathcal{J}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong \mathrm{Jac}(C)\} \text{ and}$$
$$\mathcal{E}(p) := \{[A] \in \mathcal{S}(p) \ : \ A \cong E \times E' \text{ with } E,\ E' \text{ supersingular ECs}\}.$$



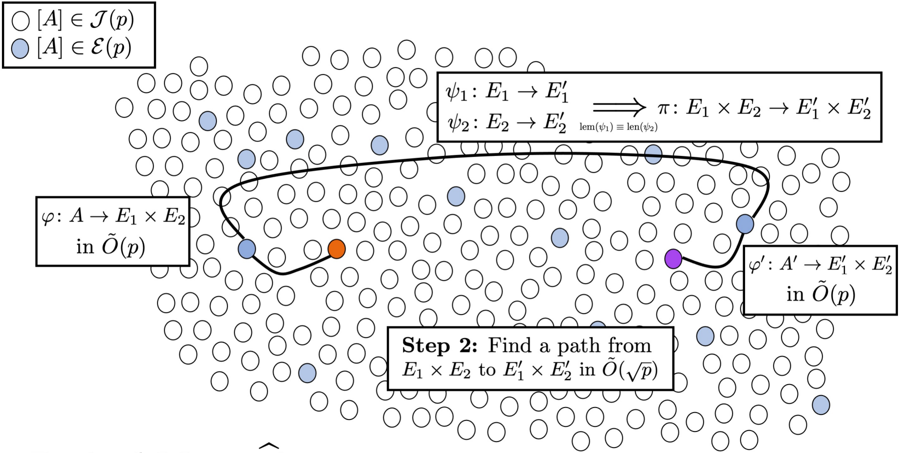
Attacking the General Isogeny Problem: Costello–Smith



Attacking the General Isogeny Problem: Costello–Smith



Attacking the General Isogeny Problem: Costello–Smith



Desired Map: $\widehat{\varphi'} \circ \pi \circ \varphi$

Attacking the General Isogeny Problem: Costello–Smith

The bottleneck of the attack is the first step: walking in $\Gamma_2(N; p)$ until finding $A \in \mathcal{J}(p)$ which is (N, N)-*split*.

Attacking the General Isogeny Problem: Costello–Smith

The bottleneck of the attack is the first step: walking in $\Gamma_2(N; p)$ until finding $A \in \mathcal{J}(p)$ which is (N, N)-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

Attacking the General Isogeny Problem: Costello–Smith

The bottleneck of the attack is the first step: walking in $\Gamma_2(N; p)$ until finding $A \in \mathcal{J}(p)$ which is (N, N)-*split*.

Definition

We say the Jacobian $\mathrm{Jac}(C)$ of a genus 2 curve C is (N, N)-*split* if there exists an (N, N)-isogeny[a] $\mathrm{Jac}(C) \to E \times E'$, where E, E' are elliptic curves.

[a] Separable, polarised, optimal

For this reason, we focus on the first step of the algorithm.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2, 2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2; p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2; p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 \colon A_1 \to A_2$.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2; p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 \colon A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2; p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 \colon A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Question: Taking steps in $\Gamma(2; p)$, can we detect whether the current node A_i is in (N, N)-split for $N > 2$?

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2; \bar{\mathbb{F}}_p)$ and detect $(2, 2)$-splittings.

First step in more detail:

1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2; p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 \colon A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Question: Taking steps in $\Gamma(2; p)$, can we detect whether the current node A_i is in (N, N)-split for $N > 2$?

Naive Answer: Compute all (N, N)-isogenies from A_i, but this is not efficient. Can we make the detection efficient?

Attacking the General Isogeny Problem: First step

Summary: Using Richelot isogenies, Costello–Smith take walks in $\Gamma(2;\bar{\mathbb{F}}_p)$ and detect $(2,2)$-splittings.

First step in more detail:

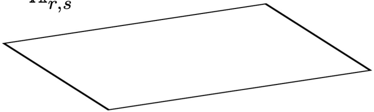
1. We start on a node $A_0 \in \mathcal{J}(p)$.
2. Take a step in $\Gamma(2;p)$ via a Richelot isogeny $\phi_1 \colon A_0 \to A_1$.
3. From the Richelot isogeny formulae, we can determine whether $A_1 \in \mathcal{E}(p)$. If not, take another step $\phi_2 \colon A_1 \to A_2$.
4. Repeat previous step until finding $A_i \in \mathcal{E}(p)$.

Question: Taking steps in $\Gamma(2;p)$, can we detect whether the current node A_i is in (N,N)-split for $N > 2$?

Naive Answer: Compute all (N,N)-isogenies from A_i, but this is not efficient. Can we make the detection efficient? If so, we could improve the concrete complexity of Costello–Smith attack.

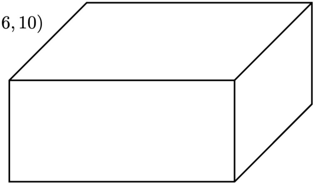
Detecting (N, N)-splittings

$\mathbb{A}^2_{r,s}$



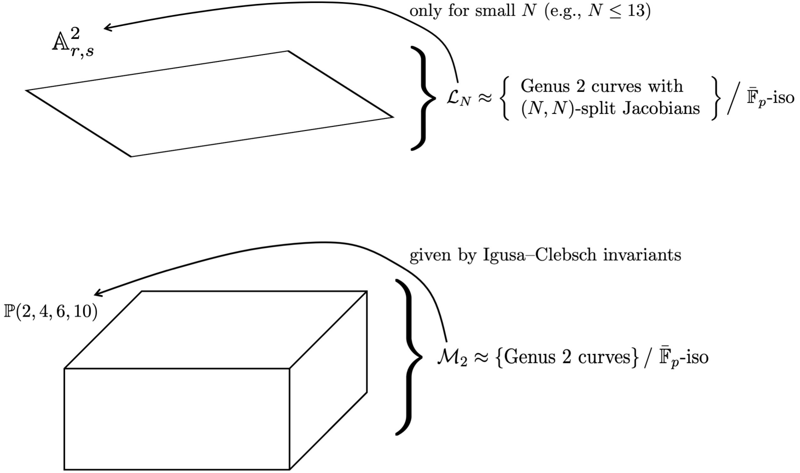
$$\mathcal{L}_N \approx \left\{ \begin{array}{c} \text{Genus 2 curves with} \\ (N, N)\text{-split Jacobians} \end{array} \right\} \Big/ \ \bar{\mathbb{F}}_p\text{-iso}$$

$\mathbb{P}(2, 4, 6, 10)$

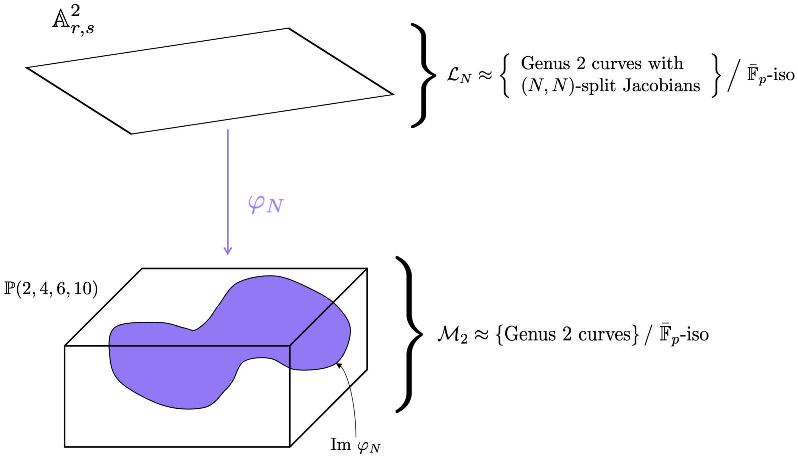


$$\mathcal{M}_2 \approx \{\text{Genus 2 curves}\} \ / \ \bar{\mathbb{F}}_p\text{-iso}$$

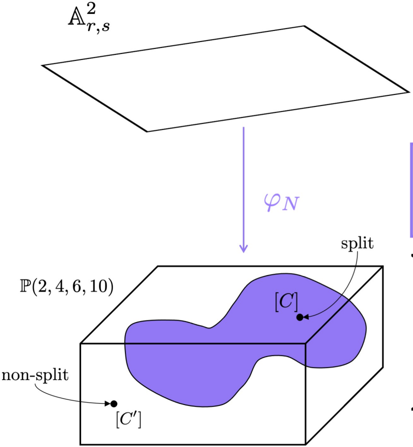
Detecting (N, N)-splittings



Detecting (N, N)-splittings



Detecting (N, N)-splittings

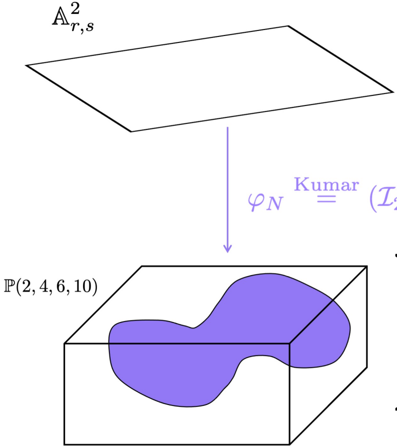


$\mathcal{L}_N \approx \left\{ \begin{matrix} \text{Genus 2 curves with} \\ (N,N)\text{-split Jacobians} \end{matrix} \right\} / \bar{\mathbb{F}}_p\text{-iso}$

Question: Is C (N, N)-split (over $\bar{\mathbb{F}}_p$)?
Answer: Yes $\iff$ $[C]$ in $\overline{\mathrm{Im}\ \varphi_\mathrm{N}}$

$\mathcal{M}_2 \approx \{\text{Genus 2 curves}\} / \bar{\mathbb{F}}_p\text{-iso}$

Detecting (N, N)-splittings



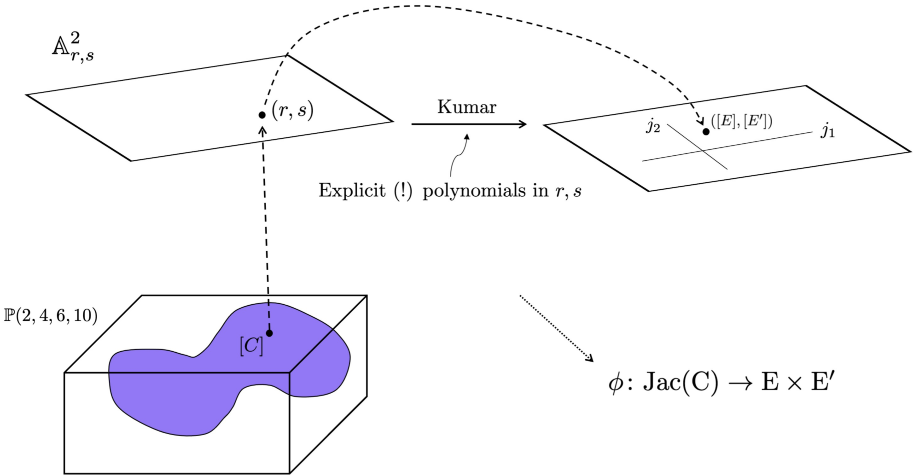
$\mathcal{L}_N \approx \left\{ \begin{array}{l} \text{Genus 2 curves with} \\ (N,N)\text{-split Jacobians} \end{array} \right\} / \bar{\mathbb{F}}_p\text{-iso}$

Explicit (!) polynomials in r, s

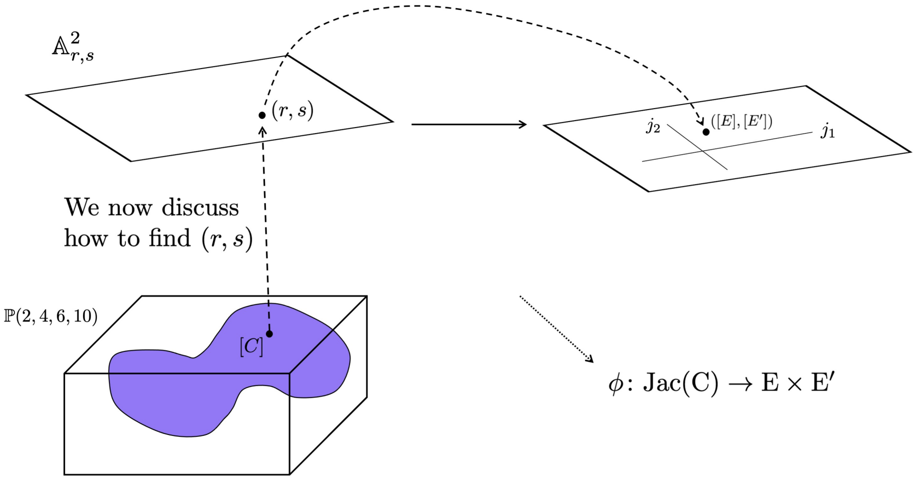
$\varphi_N \overset{\text{Kumar}}{=} (\mathcal{I}_2 : \mathcal{I}_4 : \mathcal{I}_6 : \mathcal{I}_{10})$, for $N \leq 11$.

$\mathcal{M}_2 \approx \{\text{Genus 2 curves}\} / \bar{\mathbb{F}}_p\text{-iso}$

Detecting (N, N)-splittings



Detecting (N, N)-splittings



Is C in the image?

We want to detect whether C is (N, N)-split, i.e., in the image of φ_N.

Is C in the image?

We want to detect whether C is (N,N)-split, i.e., in the image of φ_N. Let $I_2(C), I_4(C), I_6(C), I_{10}(C)$ be the Igusa–Clebsch invariants of C.

Is C in the image?

We want to detect whether C is (N, N)-split, i.e., in the image of φ_N. Let $I_2(C), I_4(C), I_6(C), I_{10}(C)$ be the Igusa–Clebsch invariants of C.

Method 1: Compute the equation F_N for the image of $\mathcal{L}_N$

Is C in the image?

We want to detect whether C is (N, N)-split, i.e., in the image of φ_N. Let $I_2(C), I_4(C), I_6(C), I_{10}(C)$ be the Igusa–Clebsch invariants of C.

Method 1: Compute the equation F_N for the image of $\mathcal{L}_N$

- Jac(C) is (N, N)-split $\iff F_N(I_2(C), I_4(C), I_6(C), I_{10}(C)) = 0$.

Is C in the image?

We want to detect whether C is (N, N)-split, i.e., in the image of φ_N. Let $I_2(C), I_4(C), I_6(C), I_{10}(C)$ be the Igusa–Clebsch invariants of C.

Method 1: Compute the equation F_N for the image of $\mathcal{L}_N$

- $\mathrm{Jac}(C)$ is (N, N)-split $\iff F_N(I_2(C), I_4(C), I_6(C), I_{10}(C)) = 0$.
- Computing F_N for $2 \leq N \leq 5$ has been done by Bruin–Doereksen [BD11] and Shaska and others [Sha04, SWWW08, MSV09].

Is C in the image?

We want to detect whether C is (N, N)-split, i.e., in the image of φ_N. Let $I_2(C), I_4(C), I_6(C), I_{10}(C)$ be the Igusa–Clebsch invariants of C.

Method 1: Compute the equation F_N for the image of $\mathcal{L}_N$

- Jac(C) is (N, N)-split $\iff F_N(I_2(C), I_4(C), I_6(C), I_{10}(C)) = 0$.
- Computing F_N for $2 \leq N \leq 5$ has been done by Bruin–Doereksen [BD11] and Shaska and others [Sha04, SWWW08, MSV09].
- The main problem is that F_N is *large* (with size growing rapidly with N), so the evaluation is inefficient.

Is C in the image?

For example, F_3 is given by:

$$256l_2^{16}l_4^{12} + 104960l_2^{14}l_4^{13} - 831744l_2^{12}l_4^{14} + 2771968l_2^{10}l_4^{15} - 5075200l_2^{8}l_4^{16} + 5528064l_2^{6}l_4^{17} - 3587840l_2^{4}l_4^{18} + 1286144l_2^{2}l_4^{19} - 196608l_4^{20} - 9216l_2^{15}l_4^{11}l_6^{1} - 4180992l_2^{13}l_4^{12}l_6^{1} + 28514304l_2^{11}l_4^{13}l_6^{1} - 79534080l_2^{9}l_4^{14}l_6^{1} + 116935680l_2^{7}l_4^{15}l_6^{1} - 95837184l_2^{5}l_4^{16}l_6^{1}$$
$$+ 41582592l_2^{3}l_4^{17}l_6^{1} - 7471104l_2^{1}l_4^{18}l_6^{1} + 152064l_2^{14}l_4^{10}l_6^{2} + 76792320l_2^{12}l_4^{11}l_6^{2} - 445381632l_2^{10}l_4^{12}l_6^{2} + 1021178880l_2^{8}l_4^{13}l_6^{2} - 1172067840l_2^{6}l_4^{14}l_6^{2} + 687868416l_2^{4}l_4^{15}l_6^{2} - 179159040l_2^{2}l_4^{16}l_6^{2} + 10616832l_4^{17}l_6^{2} - 1520640l_2^{13}l_4^{9}l_6^{3} - 861179904l_2^{11}l_4^{10}l_6^{3}$$
$$+ 4187842560l_2^{9}l_4^{11}l_6^{3} - 7720980480l_2^{7}l_4^{12}l_6^{3} + 6670494720l_2^{5}l_4^{13}l_6^{3} - 2614394880l_2^{3}l_4^{14}l_6^{3} + 339738624l_2^{1}l_4^{15}l_6^{3} + 10264320l_2^{12}l_4^{8}l_6^{4} + 6579532800l_2^{10}l_4^{9}l_6^{4} - 26374325760l_2^{8}l_4^{10}l_6^{4} + 38035215360l_2^{6}l_4^{11}l_6^{4} - 23555784960l_2^{4}l_4^{12}l_6^{4} + 5543976960l_2^{2}l_4^{13}l_6^{4}$$
$$- 238878720l_4^{14}l_6^{4} - 49268736l_2^{11}l_4^{7}l_6^{5} - 36167731200l_2^{9}l_4^{8}l_6^{5} + 117046093824l_2^{7}l_4^{9}l_6^{5} - 127410444288l_2^{5}l_4^{10}l_6^{5} + 52792197120l_2^{3}l_4^{11}l_6^{5} - 6210846720l_2^{1}l_4^{12}l_6^{5} + 172440576l_2^{10}l_4^{6}l_6^{6} + 147176911872l_2^{8}l_4^{7}l_6^{6} - 374650665984l_2^{6}l_4^{8}l_6^{6} + 293458775040l_2^{4}l_4^{9}l_6^{6}$$
$$- 73216327680l_2^{2}l_4^{10}l_6^{6} + 2866544640l_4^{11}l_6^{6} - 443418624l_2^{9}l_4^{5}l_6^{7} - 448927764480l_2^{7}l_4^{6}l_6^{7} + 869355804672l_2^{5}l_4^{7}l_6^{7} - 457930506240l_2^{3}l_4^{8}l_6^{7} + 57330892800l_2^{1}l_4^{9}l_6^{7} + 831409920l_2^{8}l_4^{4}l_6^{8} + 1026504036864l_2^{6}l_4^{5}l_6^{8} - 1446325735680l_2^{4}l_4^{6}l_6^{8} + 461961584640l_2^{2}l_4^{7}l_6^{8}$$
$$- 19349176320l_4^{8}l_6^{8} - 1108546560l_2^{7}l_4^{3}l_6^{9} - 1737898675200l_2^{5}l_4^{4}l_6^{9} + 1673703751680l_2^{3}l_4^{5}l_6^{9} - 270888468480l_2^{1}l_4^{6}l_6^{9} + 997691904l_2^{6}l_4^{2}l_6^{10} + 2117676648960l_2^{4}l_4^{3}l_6^{10} - 1268338507776l_2^{2}l_4^{4}l_6^{10} + 69657034752l_4^{5}l_6^{10} - 544195584l_2^{5}l_4^{1}l_6^{11} - 1758840127488l_2^{3}l_4^{2}l_6^{11}$$
$$+ 557256278016l_2^{1}l_4^{3}l_6^{11} + 136048896l_2^{4}l_6^{12} + 892480757760l_2^{2}l_4^{1}l_6^{12} - 104485552128l_4^{2}l_6^{12} - 208971104256l_2^{1}l_6^{13} + 22581504l_2^{15}l_4^{10}l_{10}^{1} + 13319292672l_2^{13}l_4^{11}l_{10}^{1} + 1614323354112l_2^{11}l_4^{12}l_{10}^{1} - 5919800246784l_2^{9}l_4^{13}l_{10}^{1} + 4099396158720l_2^{7}l_4^{14}l_{10}^{1} + 6877115861760l_2^{5}l_4^{15}l_{10}^{1}$$
$$- 10524909809664l_2^{3}l_4^{16}l_{10}^{1} + 3840532807680l_2^{1}l_4^{17}l_{10}^{1} - 677445120l_2^{14}l_4^{9}l_6^{1}l_{10}^{1} - 421643335680l_2^{12}l_4^{10}l_6^{1}l_{10}^{1} - 49309068432384l_2^{10}l_4^{11}l_6^{1}l_{10}^{1} + 151510642145280l_2^{8}l_4^{12}l_6^{1}l_{10}^{1} - 86071505001984l_2^{6}l_4^{13}l_6^{1}l_{10}^{1} - 94370665697280l_2^{4}l_4^{14}l_6^{1}l_{10}^{1}$$
$$+ 89993222111232l_2^{2}l_4^{15}l_6^{1}l_{10}^{1} - 11330304344064l_4^{16}l_6^{1}l_{10}^{1} + 9145509120l_2^{13}l_4^{8}l_6^{2}l_{10}^{1} + 6096314078208l_2^{11}l_4^{9}l_6^{2}l_{10}^{1} + 677114106877440l_2^{9}l_4^{10}l_6^{2}l_{10}^{1} - 1714742371519488l_2^{7}l_4^{11}l_6^{2}l_{10}^{1} + 779711791336704l_2^{5}l_4^{12}l_6^{2}l_{10}^{1} + 503515992158208l_2^{3}l_4^{13}l_6^{2}l_{10}^{1}$$
$$- 251704978440192l_2^{1}l_4^{14}l_6^{2}l_{10}^{1} - 73164072960l_2^{12}l_4^{7}l_6^{3}l_{10}^{1} - 53165449598976l_2^{10}l_4^{8}l_6^{3}l_{10}^{1} - 5505267460184064l_2^{8}l_4^{9}l_6^{3}l_{10}^{1} + 11256934815995904l_2^{6}l_4^{10}l_6^{3}l_{10}^{1} - 3950595178610688l_2^{4}l_4^{11}l_6^{3}l_{10}^{1} - 1292808013627392l_2^{2}l_4^{12}l_6^{3}l_{10}^{1}$$
$$+ 229563787640832l_4^{13}l_6^{3}l_{10}^{1} + 384111383040l_2^{11}l_4^{6}l_6^{4}l_{10}^{1} + 310864743516672l_2^{9}l_4^{7}l_6^{4}l_{10}^{1} + 29351137557800448l_2^{7}l_4^{8}l_6^{4}l_{10}^{1} - 47205144431829504l_2^{5}l_4^{9}l_6^{4}l_{10}^{1} + 12090771888611328l_2^{3}l_4^{10}l_6^{4}l_{10}^{1} + 1568284279308288l_2^{1}l_4^{11}l_6^{4}l_{10}^{1} - 1382800978944l_2^{10}l_4^{5}l_6^{5}l_{10}^{1}$$
$$- 1280057215481856l_2^{8}l_4^{6}l_6^{5}l_{10}^{1} - 107227854593823744l_2^{6}l_4^{7}l_6^{5}l_{10}^{1} + 131009780820123648l_2^{4}l_4^{8}l_6^{5}l_{10}^{1} - 22349349309874176l_2^{2}l_4^{9}l_6^{5}l_{10}^{1} - 691647917064192l_4^{10}l_6^{5}l_{10}^{1} + 3457002447360l_2^{9}l_4^{4}l_6^{6}l_{10}^{1} + 3788611835839488l_2^{7}l_4^{5}l_6^{6}l_{10}^{1} + 271863446778948096l_2^{5}l_4^{6}l_6^{6}l_{10}^{1}$$
$$- 240356478757773312l_2^{3}l_4^{7}l_6^{6}l_{10}^{1} + 23099294360272896l_2^{1}l_4^{8}l_6^{6}l_{10}^{1} - 5926289909760l_2^{8}l_4^{3}l_6^{7}l_{10}^{1} - 8060769746233344l_2^{6}l_4^{4}l_6^{7}l_{10}^{1} - 472377570351759360l_2^{4}l_4^{5}l_6^{7}l_{10}^{1} + 280666147116515328l_2^{2}l_4^{6}l_6^{7}l_{10}^{1} - 10295588364484608l_4^{7}l_6^{7}l_{10}^{1}$$
$$+ 6667076148480l_2^{7}l_4^{2}l_6^{8}l_{10}^{1} + 12082146413799168l_2^{5}l_4^{3}l_6^{8}l_{10}^{1} + 538369245404762112l_2^{3}l_4^{4}l_6^{8}l_{10}^{1} - 188897339981166640l_2^{1}l_4^{5}l_6^{8}l_{10}^{1} - 4444717432320l_2^{6}l_4^{1}l_6^{9}l_{10}^{1} - 12148972679172096l_2^{4}l_4^{2}l_6^{9}l_{10}^{1} - 363459183846211584l_2^{2}l_4^{3}l_6^{9}l_{10}^{1} + 55673663653675008l_4^{4}l_6^{9}l_{10}^{1}$$
$$+ 1333415229696l_2^{5}l_6^{10}l_{10}^{1} + 7373969886228480l_2^{3}l_4^{1}l_6^{10}l_{10}^{1} + 110410718818074624l_2^{1}l_4^{2}l_6^{10}l_{10}^{1} - 2046009960382464l_2^{2}l_6^{11}l_{10}^{1} - 33853318889472l_4^{1}l_6^{11}l_{10}^{1} + 829954952640l_2^{14}l_4^{8}l_{10}^{2} - 331561477643040l_2^{12}l_4^{9}l_{10}^{2} - 277722993887863200l_2^{10}l_4^{10}l_{10}^{2}$$
$$+ 2254138236578251296l_2^{8}l_4^{11}l_{10}^{2} - 5978653419073689504l_2^{6}l_4^{12}l_{10}^{2} + 7873496874124223616l_2^{4}l_4^{13}l_{10}^{2} - 5859528108438822912l_2^{2}l_4^{14}l_{10}^{2} + 1418239425570766848l_4^{15}l_{10}^{2} - 19918918863360l_2^{13}l_4^{7}l_6^{1}l_{10}^{2} + 6435225871444800l_2^{11}l_4^{8}l_6^{1}l_{10}^{2} + 7623595670662908288l_2^{9}l_4^{9}l_6^{1}l_{10}^{2}$$
$$- 50093733457558855872l_2^{7}l_4^{10}l_6^{1}l_{10}^{2} + 100464530313716070912l_2^{5}l_4^{11}l_6^{1}l_{10}^{2} - 86281638869222654976l_2^{3}l_4^{12}l_6^{1}l_{10}^{2} + 33536254113170472960l_2^{1}l_4^{13}l_6^{1}l_{10}^{2} + 209148648065280l_2^{12}l_4^{6}l_6^{2}l_{10}^{2} - 47682015912904032l_2^{10}l_4^{7}l_6^{2}l_{10}^{2} - 92763675562396294080l_2^{8}l_4^{8}l_6^{2}l_{10}^{2}$$
$$+ 480024550899866454048l_2^{6}l_4^{9}l_6^{2}l_{10}^{2} - 696807006985015640064l_2^{4}l_4^{10}l_6^{2}l_{10}^{2} + 345089664401090568192l_2^{2}l_4^{11}l_6^{2}l_{10}^{2} - 46347413831231864832l_4^{12}l_6^{2}l_{10}^{2} - 1254891888391680l_2^{11}l_4^{5}l_6^{3}l_{10}^{2} + 131894732684886912l_2^{9}l_4^{6}l_6^{3}l_{10}^{2} + 656855732954360345472l_2^{7}l_4^{7}l_6^{3}l_{10}^{2}$$
$$- 2580571446131453704704l_2^{5}l_4^{8}l_6^{3}l_{10}^{2} + 2551599910178391232512l_2^{3}l_4^{9}l_6^{3}l_{10}^{2} - 591214438416657088512l_2^{1}l_4^{10}l_6^{3}l_{10}^{2} + 4705844581468800l_2^{10}l_4^{4}l_6^{4}l_{10}^{2} + 291650742411226272l_2^{8}l_4^{5}l_6^{4}l_{10}^{2} - 2983483435458015826848l_2^{6}l_4^{6}l_6^{4}l_{10}^{2} + 8461165647508933711104l_2^{4}l_4^{7}l_6^{4}l_{10}^{2}$$
$$- 5197986514201121021952l_2^{2}l_4^{8}l_6^{4}l_{10}^{2} + 359459015816010203136l_4^{9}l_6^{4}l_{10}^{2} - 11294026995525120l_2^{9}l_4^{3}l_6^{5}l_{10}^{2} - 3457026379551234240l_2^{7}l_4^{4}l_6^{5}l_{10}^{2} + 9015964740354251736576l_2^{5}l_4^{5}l_6^{5}l_{10}^{2} - 17154557259672759883776l_2^{3}l_4^{6}l_6^{5}l_{10}^{2} + 5578614112858226786304l_2^{1}l_4^{7}l_6^{5}l_{10}^{2}$$
$$+ 16941040493287680l_2^{8}l_4^{2}l_6^{6}l_{10}^{2} + 11924054187176803680l_2^{6}l_4^{3}l_6^{6}l_{10}^{2} - 18130387929865453403136l_2^{4}l_4^{4}l_6^{6}l_{10}^{2} + 20629414334909533003776l_2^{2}l_4^{5}l_6^{6}l_{10}^{2} - 2460228863859181486080l_4^{6}l_6^{6}l_{10}^{2} - 14520891851389440l_2^{7}l_4^{1}l_6^{7}l_{10}^{2} - 21564868878583432704l_2^{5}l_4^{2}l_6^{7}l_{10}^{2}$$
$$+ 23398095828525487755264l_2^{3}l_4^{3}l_6^{7}l_{10}^{2} - 12963770211274304716800l_2^{1}l_4^{4}l_6^{7}l_{10}^{2} + 5445334444271040l_2^{6}l_6^{8}l_{10}^{2} + 20686517328243360384l_2^{4}l_4^{1}l_6^{8}l_{10}^{2} - 17587288756831258312704l_2^{2}l_4^{2}l_6^{8}l_{10}^{2} + 2951768088009919660032l_4^{3}l_6^{8}l_{10}^{2} - 8343272497393714176l_2^{3}l_6^{9}l_{10}^{2} + 5867149154142230691840l_2^{1}l_4^{1}l_6^{9}l_{10}^{2}$$
$$- 2285099025039360l_6^{10}l_{10}^{2} + 16268776981649280l_2^{13}l_4^{6}l_{10}^{3} + 24805141115326579440l_2^{11}l_4^{7}l_{10}^{3} + 7424575548105165856992l_2^{9}l_4^{8}l_{10}^{3} - 63830677396455144483408l_2^{7}l_4^{9}l_{10}^{3} + 163478193852927668601600l_2^{5}l_4^{10}l_{10}^{3} - 67181404446561091396608l_2^{3}l_4^{11}l_{10}^{3}$$
$$- 112537794474362231930880l_2^{1}l_4^{12}l_{10}^{3} - 292837985669687040l_2^{12}l_4^{5}l_6^{1}l_{10}^{3} - 535281027822695283840l_2^{10}l_4^{6}l_6^{1}l_{10}^{3} - 188230125255528599529408l_2^{8}l_4^{7}l_6^{1}l_{10}^{3} + 1321918217384252979598272l_2^{6}l_4^{8}l_6^{1}l_{10}^{3} - 2670553082037902148306432l_2^{4}l_4^{9}l_6^{1}l_{10}^{3} + 1080781518138930590846976l_2^{2}l_4^{10}l_6^{1}l_{10}^{3}$$
$$+ 467829765560544587218944l_4^{11}l_6^{1}l_{10}^{3} + 2196284892522652800l_2^{11}l_4^{4}l_6^{2}l_{10}^{3} + 4906162386527578347168l_2^{9}l_4^{5}l_6^{2}l_{10}^{3} + 2066374132888516002821088l_2^{7}l_4^{6}l_6^{2}l_{10}^{3} - 11373672332568284199569664l_2^{5}l_4^{7}l_6^{2}l_{10}^{3} + 16804586151308878201396224l_2^{3}l_4^{8}l_6^{2}l_{10}^{3} - 5228602103523200568311808l_2^{1}l_4^{9}l_6^{2}l_{10}^{3}$$
$$- 8785139570090611200l_2^{10}l_4^{3}l_6^{3}l_{10}^{3} - 24768842954191533178560l_2^{8}l_4^{4}l_6^{3}l_{10}^{3} - 12829997166657489137983104l_2^{6}l_4^{5}l_6^{3}l_{10}^{3} + 52295359302184507505293824l_2^{4}l_4^{6}l_6^{3}l_{10}^{3} - 50337523991591860203712512l_2^{2}l_4^{7}l_6^{3}l_{10}^{3} + 7929993708116801520205824l_4^{8}l_6^{3}l_{10}^{3}$$
$$+ 19766564032703875200l_2^{9}l_4^{2}l_6^{4}l_{10}^{3} + 74400680625120947458800l_2^{7}l_4^{3}l_6^{4}l_{10}^{3} + 49275552146140204730406144l_2^{5}l_4^{4}l_6^{4}l_{10}^{3} - 136907992026836672253299712l_2^{3}l_4^{5}l_6^{4}l_{10}^{3} + 70192431186766521673826304l_2^{1}l_4^{6}l_6^{4}l_{10}^{3} - 23719876839244650240l_2^{8}l_4^{1}l_6^{5}l_{10}^{3} - 132966636769128654239040l_2^{6}l_4^{2}l_6^{5}l_{10}^{3}$$
$$- 119850437391136595493622272l_2^{4}l_4^{3}l_6^{5}l_{10}^{3} + 198753997392344001530474496l_2^{2}l_4^{4}l_6^{5}l_{10}^{3} - 34674636419830654939496448l_4^{5}l_6^{5}l_{10}^{3} + 11859938419622325120l_2^{7}l_6^{6}l_{10}^{3} + 130879819699500885684480l_2^{5}l_4^{1}l_6^{6}l_{10}^{3} + 180215155544260371212473344l_2^{3}l_4^{2}l_6^{6}l_{10}^{3}$$
$$- 139172056834888597536129024l_2^{1}l_4^{3}l_6^{6}l_{10}^{3} - 54708260027772068504064l_2^{4}l_6^{7}l_{10}^{3} - 153075433353875029496254464l_2^{2}l_4^{1}l_6^{7}l_{10}^{3} + 29042357719222340071391232l_4^{2}l_6^{7}l_{10}^{3} + 56181198271377991197818880l_2^{1}l_6^{8}l_{10}^{3} + 179381568596787667440l_2^{12}l_4^{4}l_{10}^{4} - 10310985429287897554200l_2^{10}l_4^{5}l_{10}^{4}$$
$$- 34312667843627927307743319l_2^{8}l_4^{6}l_{10}^{4} + 796315712724979246147691736l_2^{6}l_4^{7}l_{10}^{4} - 1259826593347739038467698352l_2^{4}l_4^{8}l_{10}^{4} - 20344243618077880895097984l_2^{2}l_4^{9}l_{10}^{4} - 6324413850721664271452160l_4^{10}l_{10}^{4} - 2152578823161452009280l_2^{11}l_4^{3}l_6^{1}l_{10}^{4} + 479117447032084696558800l_2^{9}l_4^{4}l_6^{1}l_{10}^{4}$$
$$+ 802011529127059581156585984l_2^{7}l_4^{5}l_6^{1}l_{10}^{4} - 15323604635991919695054107328l_2^{5}l_4^{6}l_6^{1}l_{10}^{4} + 20799640382757665765942937600l_2^{3}l_4^{7}l_6^{1}l_{10}^{4} - 572690847823168270695358464l_2^{1}l_4^{8}l_6^{1}l_{10}^{4} + 9686604704226534041760l_2^{10}l_4^{2}l_6^{2}l_{10}^{4} - 5070368236070098695762072l_2^{8}l_4^{3}l_6^{2}l_{10}^{4} - 7625147434979773940119000728l_2^{6}l_4^{4}l_6^{2}l_{10}^{4}$$
$$+ 116494719221085818398366770144l_2^{4}l_4^{5}l_6^{2}l_{10}^{4} - 123426632204089848461637836928l_2^{2}l_4^{6}l_6^{2}l_{10}^{4} + 3951333456727127362355865600l_4^{7}l_6^{2}l_{10}^{4} - 19373209408453068083520l_2^{9}l_4^{1}l_6^{3}l_{10}^{4} + 22513456585517310751681728l_2^{7}l_4^{2}l_6^{3}l_{10}^{4} + 37884011811289309371575756736l_2^{5}l_4^{3}l_6^{3}l_{10}^{4}$$
$$- 438312829896484653721287297024l_2^{3}l_4^{4}l_6^{3}l_{10}^{4} + 312364817105781230834640224256l_2^{1}l_4^{5}l_6^{3}l_{10}^{4} + 14529907056339801062640l_2^{8}l_6^{4}l_{10}^{4} - 45749532319814748443413824l_2^{6}l_4^{1}l_6^{4}l_{10}^{4} - 104000451990854340623549473584l_2^{4}l_4^{2}l_6^{4}l_{10}^{4} + 818672079332554467681573850752l_2^{2}l_4^{3}l_6^{4}l_{10}^{4}$$
$$- 284249682355622621015443027968l_4^{4}l_6^{4}l_{10}^{4} + 35224486313399212035089664l_2^{5}l_6^{5}l_{10}^{4} + 149858496195783453346109865984l_2^{3}l_4^{1}l_6^{5}l_{10}^{4} - 612245404314136559879648354304l_2^{1}l_4^{2}l_6^{5}l_{10}^{4} - 88669301278410923407787237760l_2^{2}l_6^{6}l_{10}^{4} + 9165518530093743788178708480l_4^{1}l_6^{6}l_{10}^{4}$$
$$+ 1054871252290269557147664l_2^{11}l_4^{2}l_{10}^{5} + 577288230697319033544931620l_2^{9}l_4^{3}l_{10}^{5} + 58717469688192231349277967804l_2^{7}l_4^{4}l_{10}^{5} - 2820775415140689401632583611440l_2^{5}l_4^{5}l_{10}^{5} + 8721932098919675696971459596864l_2^{3}l_4^{6}l_{10}^{5} - 1491596402356017740874185902080l_2^{1}l_4^{7}l_{10}^{5} - 6329227513741617342885984l_2^{10}l_4^{1}l_6^{1}l_{10}^{5}$$
$$- 6286988918499042776419214928l_2^{8}l_4^{2}l_6^{1}l_{10}^{5} - 1082386496506677749243280264960l_2^{6}l_4^{3}l_6^{1}l_{10}^{5} + 49800946270566279856094859866112l_2^{4}l_4^{4}l_6^{1}l_{10}^{5} - 103399589598530590719457483392000l_2^{2}l_4^{5}l_6^{1}l_{10}^{5} + 8908376388256115488701595496448l_4^{6}l_6^{1}l_{10}^{5} + 9493841270612426014328976l_2^{9}l_6^{2}l_{10}^{5}$$
$$+ 23272031641459016746661145408l_2^{7}l_4^{1}l_6^{2}l_{10}^{5} + 7328383493076056345856313236336l_2^{5}l_4^{2}l_6^{2}l_{10}^{5} - 320340810135851745605350138657152l_2^{3}l_4^{3}l_6^{2}l_{10}^{5} + 411657840398330739755761223356416l_2^{1}l_4^{4}l_6^{2}l_{10}^{5} - 29110617891065891381589590400l_2^{6}l_6^{3}l_{10}^{5} - 21642421772385061394918633040384l_2^{4}l_4^{1}l_6^{3}l_{10}^{5}$$
$$+ 894262083973041858964328319501312l_2^{2}l_4^{2}l_6^{3}l_{10}^{5} - 540270356651972200647003820990464l_4^{3}l_6^{3}l_{10}^{5} + 23550833423951148019186352040768l_2^{3}l_6^{4}l_{10}^{5} - 917952869636731807668237883705344l_2^{1}l_4^{1}l_6^{4}l_{10}^{5} + 614341646859547117618140377088l_6^{5}l_{10}^{5} + 2584698285924232982401063716l_2^{10}l_{10}^{6}$$
$$+ 1198295880913809634821436851048l_2^{8}l_4^{1}l_{10}^{6} + 73484834778514733215983989169492l_2^{6}l_4^{2}l_{10}^{6} + 3518495667617387645821786802156232l_2^{4}l_4^{3}l_{10}^{6} - 16774053331541910064309653348741984l_2^{2}l_4^{4}l_{10}^{6} + 1498018164069949798113492863030784l_4^{5}l_{10}^{6} - 4687254536607984062192553308160l_2^{7}l_6^{1}l_{10}^{6}$$
$$- 638142104786507649257407979745024l_2^{5}l_4^{1}l_6^{1}l_{10}^{6} - 49368808851950957940226307659284864l_2^{3}l_4^{2}l_6^{1}l_{10}^{6} + 153691935954674129462393036347921920l_2^{1}l_4^{3}l_6^{1}l_{10}^{6} + 1343811007060021734262129698361056l_2^{4}l_6^{2}l_{10}^{6} + 232917437818562442249812346972033120l_2^{2}l_4^{1}l_6^{2}l_{10}^{6}$$
$$- 3296890672788378880374149570048806400l_4^{2}l_6^{2}l_{10}^{6} - 356161297463842575248378721456422400l_2^{1}l_6^{3}l_{10}^{6} + 58879180974264105377458087129861488l_2^{5}l_{10}^{7} + 3697541916321297635392988322593326800l_2^{3}l_4^{1}l_{10}^{7} + 16793765487257250745071728222413248000l_2^{1}l_4^{2}l_{10}^{7}$$
$$- 12329531453117341331968481122874328000l_2^{2}l_6^{1}l_{10}^{7} - 67401304166884125252205065992797440000l_4^{1}l_6^{1}l_{10}^{7} - 3883782492464424496895506065638925150000l_{10}^{8}$$

Is C in the image?

N	Weighted degree of F_N	Number of monomials in F_N	Average bitlength of the coefficients of F_N
2	30	34	~ 16.6
3	80	318	~ 64.3
4	180	2699	~ 197
5	480	43410	~ 617

Table: The number of monomials in the defining equation for the image of $\mathcal{L}_N$ in $\mathbb{P}(2,4,6,10)$.

Is C in the image?

Method 2: Computing resultants
We normalise the Igusa–Clebsh invariants $I_2(C)$, $I_4(C)$, $I_6(C)$, $I_{10}(C)$ as:

$$\alpha_1(C) = \frac{I_4(C)}{I_2(C)^2}, \quad \alpha_2(C) = \frac{I_2(C)I_4(C)}{I_6(C)}, \quad \alpha_3(C) = \frac{I_4(C)I_6(C)}{I_{10}(C)}$$

Is C in the image?

Method 2: Computing resultants
We normalise the Igusa–Clebsh invariants $I_2(C)$, $I_4(C)$, $I_6(C)$, $I_{10}(C)$ as:

$$\alpha_1(C) = \frac{I_4(C)}{I_2(C)^2}, \quad \alpha_2(C) = \frac{I_2(C)I_4(C)}{I_6(C)}, \quad \alpha_3(C) = \frac{I_4(C)I_6(C)}{I_{10}(C)}$$

Kumar [Kum15] gives us the map

$$\varphi_N = \Big(\mathcal{I}_2(r,s) : \mathcal{I}_4(r,s) : \mathcal{I}_6(r,s) : \mathcal{I}_{10}(r,s)\Big).$$

Is C in the image?

Method 2: Computing resultants
We normalise the Igusa–Clebsh invariants $I_2(C)$, $I_4(C)$, $I_6(C)$, $I_{10}(C)$ as:

$$\alpha_1(C) = \frac{I_4(C)}{I_2(C)^2}, \quad \alpha_2(C) = \frac{I_2(C)I_4(C)}{I_6(C)}, \quad \alpha_3(C) = \frac{I_4(C)I_6(C)}{I_{10}(C)}$$

Kumar [Kum15] gives us the map

$$\varphi_N = \Big(\mathcal{I}_2(r,s) : \mathcal{I}_4(r,s) : \mathcal{I}_6(r,s) : \mathcal{I}_{10}(r,s)\Big).$$

We chose the same normalisation of the $\mathcal{I}_k(r,s)$ to give us $i_1(r,s)$, $i_2(r,s)$ and $i_3(r,s)$.

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(r), \mathrm{res}_2(r))$.

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(r), \mathrm{res}_2(r))$. If degree is 0, then Jac(C) is not (N,N)-split.

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then Jac(C) is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(r), \mathrm{res}_2(r))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and r_0 is a root of the GCD. Then solve for s_0.

This method is more efficient (and requires less memory).

Is C in the image?

Method 2: Computing resultants

Suppose there exist a simultaneous solution $r_0, s_0 \in \bar{\mathbb{F}}_p$ of

$$\begin{cases} f_1(r,s) = i_1(r_0,s_0) - \alpha_1(C) \\ f_2(r,s) = i_2(r_0,s_0) - \alpha_2(C) \\ f_3(r,s) = i_3(r_0,s_0) - \alpha_3(C) \end{cases}$$

such that the denominators of $f_i(r,s)$ do not vanish at (r_0, s_0). Then $\mathrm{Jac}(C)$ is (N,N)-split.

We determine if $\exists r_0, s_0$ by:

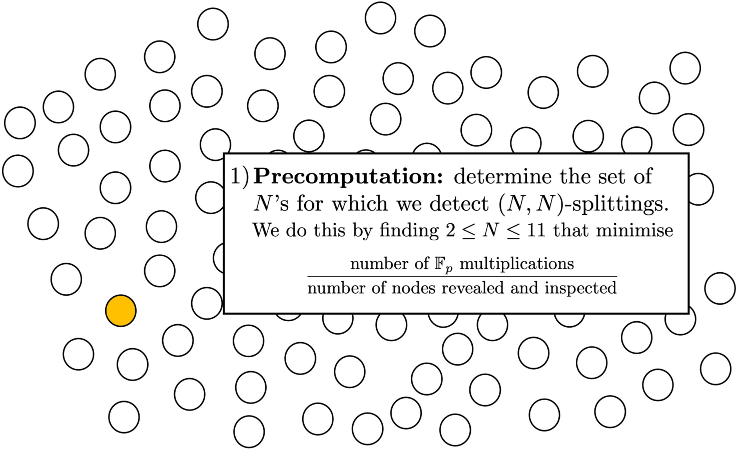
(1) Computing resultants of (the numerators of) $f_1(r,s)$, $f_2(r,s)$ and $f_2(r,s)$, $f_3(r,s)$ (with respect to r) to get $\mathrm{res}_1(s), \mathrm{res}_2(s)$.

(2) Compute $\gcd(\mathrm{res}_1(r), \mathrm{res}_2(r))$. If degree is 0, then Jac(C) is not (N,N)-split. Otherwise, Jac(C) is (N,N)-split and r_0 is a root of the GCD. Then solve for s_0.

This method is more efficient (and requires less memory). In fact, we obtain a more efficient method by precomputing the resultants generically.

Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.

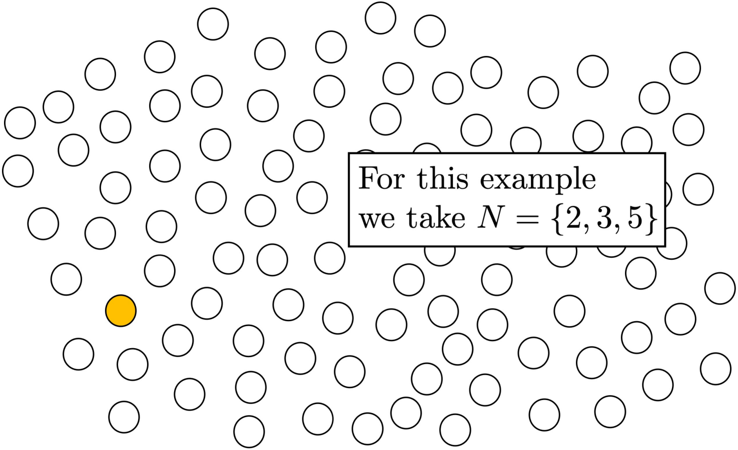


1) **Precomputation:** determine the set of N's for which we detect (N, N)-splittings. We do this by finding $2 \leq N \leq 11$ that minimise

$$\frac{\text{number of } \mathbb{F}_p \text{ multiplications}}{\text{number of nodes revealed and inspected}}$$

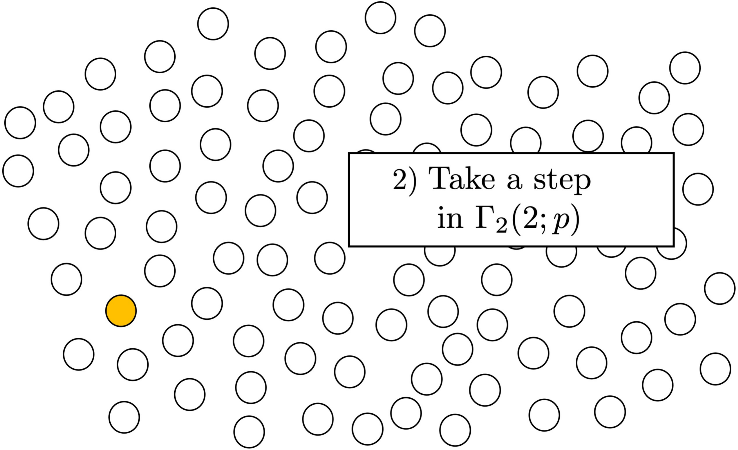
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



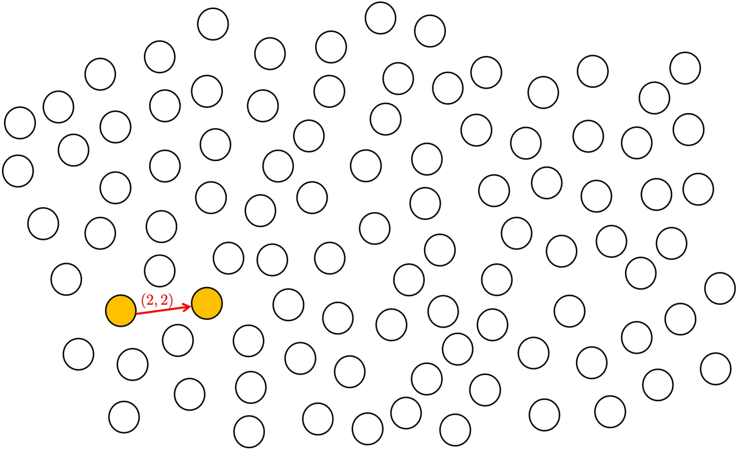
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



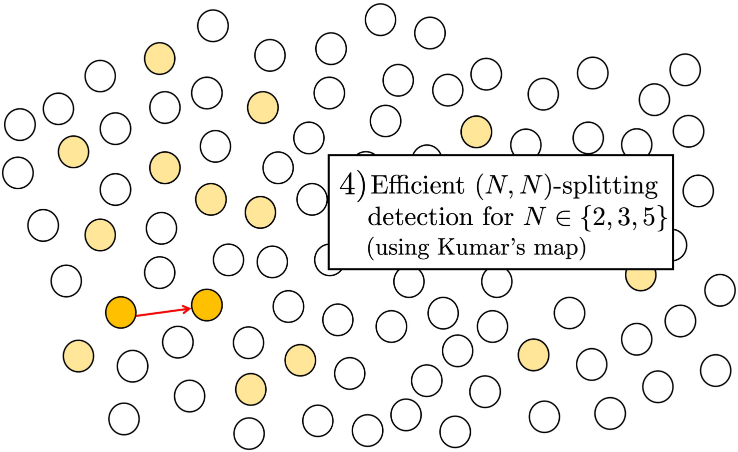
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



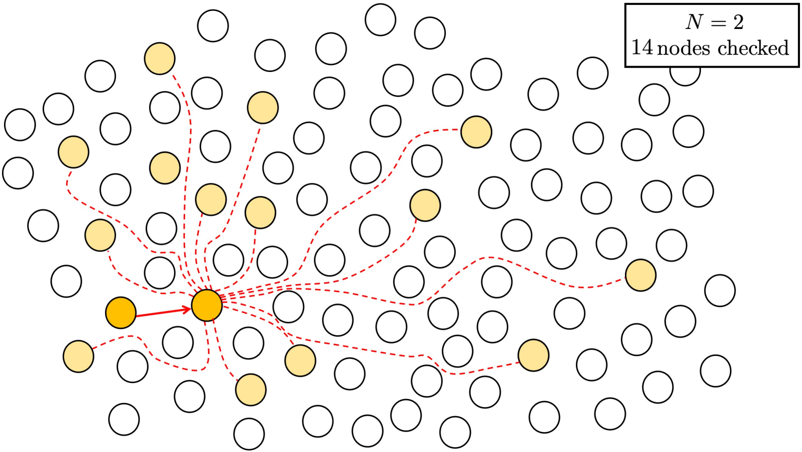
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



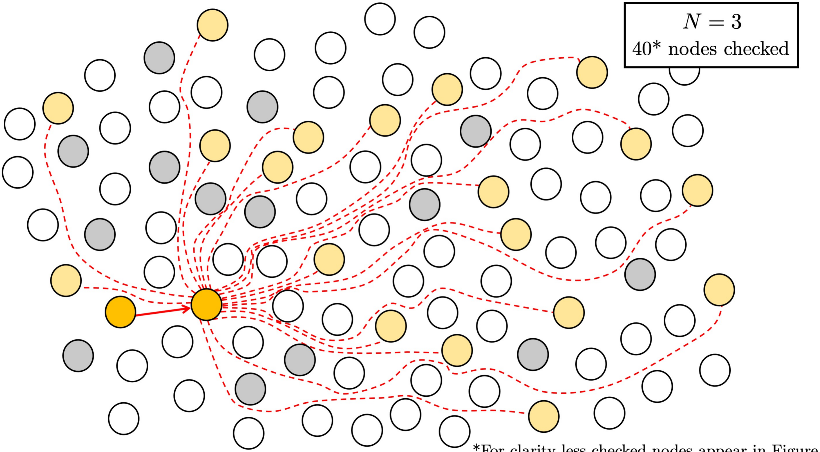
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



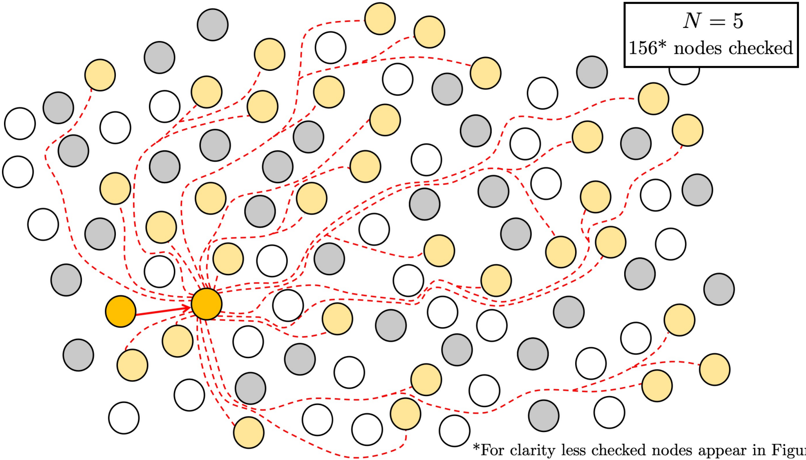
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



Attacking the General Isogeny Problem: Revisted

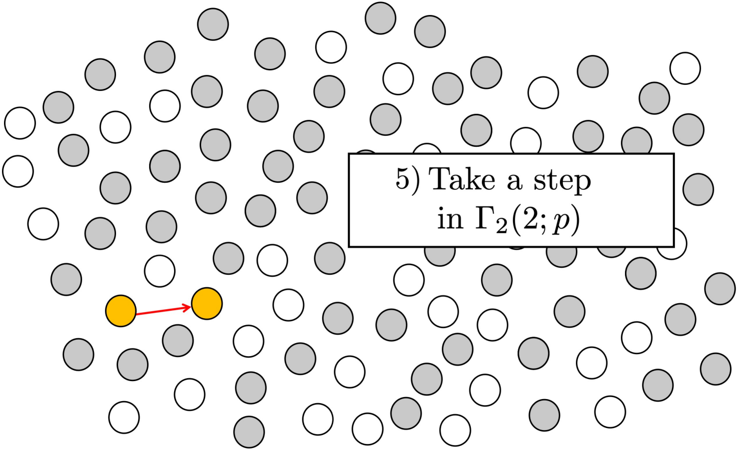
We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



*For clarity less checked nodes appear in Figure

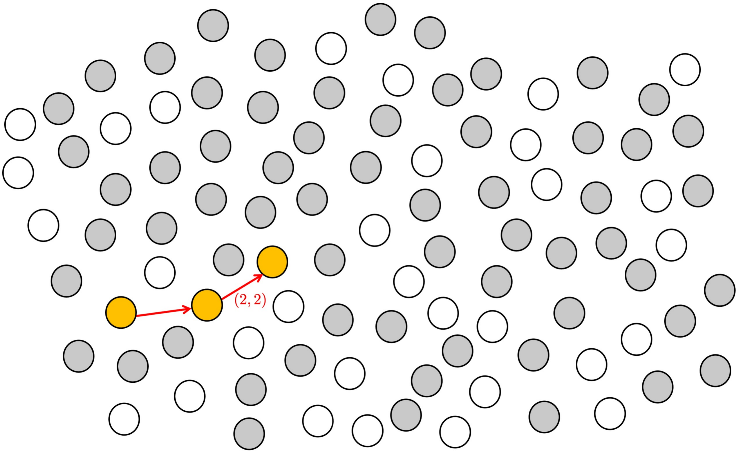
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



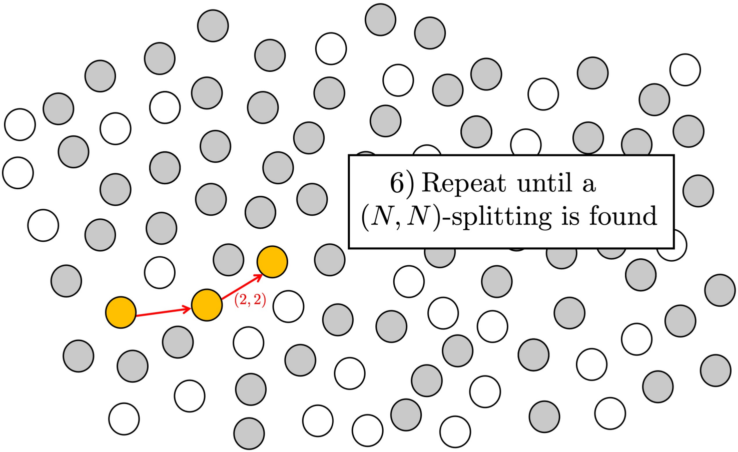
Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



Attacking the General Isogeny Problem: Revisted

We now apply efficient splitting detection to the Costello–Smith algorithm and decreasing its concrete complexity.



Preliminary Experiments

We implemented and optimised the first step of Costello–Smith attack with *and* without detection of (N, N)-spliting. We ran these (for primes p of bitsizes $50 - 1000$) until reaching 10^8 $\mathbb{F}_p$ multiplications.

Preliminary Experiments

We implemented and optimised the first step of Costello–Smith attack with *and* without detection of (N, N)-spliting. We ran these (for primes p of bitsizes $50 - 1000$) until reaching 10^8 $\mathbb{F}_p$ multiplications. We counted the number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

Preliminary Experiments

We implemented and optimised the first step of Costello–Smith attack with *and* without detection of (N, N)-spliting. We ran these (for primes p of bitsizes $50 - 1000$) until reaching 10^8 $\mathbb{F}_p$ multiplications. We counted the number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

	Walks in $\Gamma_2(2; p)$ without additional searching [CS20]			Walks in $\Gamma_2(2; p)$ w. split searching in $\Gamma_2(N; p)$ **This work**			
prime p	bits p	nodes per 10^8 muls	muls per node	set $N \in \{\dots\}$	nodes per 10^8 muls	muls per node	**imprv. factor**
$2^{11} \cdot 3^{24} - 1$	50	172712	579	$\{2, 3\}$	2830951	35	**16.5**
$2^{27} \cdot 3^{77} - 1$	150	63492	1575	$\{3, 4\}$	1858912	54	**29.2**
$2^{181} \cdot 3^{43} - 1$	250	34083	2934	$\{4, 6\}$	1771608	56	**52.4**
$2^{113} \cdot 3^{244} - 1$	500	20239	4941	$\{4, 6\}$	1667360	60	**82.4**
$2^{107} \cdot 3^{437} - 1$	800	13228	7560	$\{4, 6\}$	1548504	65	**116.3**
$2^{721} \cdot 3^{176} - 1$	1000	8814	11346	$\{4, 6\}$	1403752	71	**159.8**

Preliminary Experiments

We implemented and optimised the first step of Costello–Smith attack with *and* without detection of (N, N)-spliting. We ran these (for primes p of bitsizes $50 - 1000$) until reaching 10^8 $\mathbb{F}_p$ multiplications. We counted the number of nodes revealed and $\mathbb{F}_p$ multiplications per node revealed.

	Walks in $\Gamma_2(2;p)$ without additional searching [CS20]			Walks in $\Gamma_2(2;p)$ w. split searching in $\Gamma_2(N;p)$ **This work**			
prime p	bits p	nodes per 10^8 muls	muls per node	set $N \in \{\dots\}$	nodes per 10^8 muls	muls per node	**imprv. factor**
$2^{11} \cdot 3^{24} - 1$	50	172712	579	$\{2,3\}$	2830951	35	**16.5**
$2^{27} \cdot 3^{77} - 1$	150	63492	1575	$\{3,4\}$	1858912	54	**29.2**
$2^{181} \cdot 3^{43} - 1$	250	34083	2934	$\{4,6\}$	1771608	56	**52.4**
$2^{113} \cdot 3^{244} - 1$	500	20239	4941	$\{4,6\}$	1667360	60	**82.4**
$2^{107} \cdot 3^{437} - 1$	800	13228	7560	$\{4,6\}$	1548504	65	**116.3**
$2^{721} \cdot 3^{176} - 1$	1000	8814	11346	$\{4,6\}$	1403752	71	**159.8**

Any questions?