Leuven Isogeny Days 6, COSIC, 12th of September 2025

A presentation on work in progress

On the Concrete Classical Hardness of the Supersingular Isogeny Problem

Joint with Lorenz Panny and Alessandro Sferlazza

Ryan Rueger
IBM Research Zurich & Technical University of Munich

Concreteness and measuring the cost of attacks

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
    - cost of memory access is $O(\sqrt{M})$ (McEliece and NTRU explicitly mention this)

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
   - VLSI model with Area-Time cost measure
   - Can design ASICs or FPGAs (simple chips)
   - Performance can be evaluated through simulation
   - Used in analysis of SIKE and lead to suggested lower parameters

     [Longa-Wang-Szefer20]

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
4. Full best-effort implementation on powerful hardware

Concreteness and measuring the cost of attacks

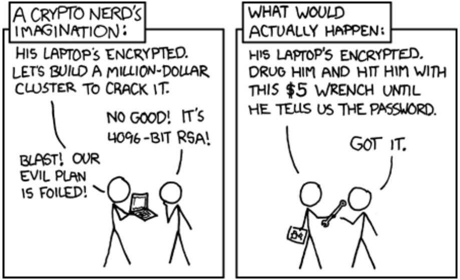
1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
4. Full best-effort implementation on powerful hardware
    - Allows one to test assumptions about e.g. memory constraints
    - Gives real-world numbers
    - In good cases, breaking small instances can be extrapolated to big instances
    - It can be used constructively for non-cryptographic sizes!
    - Example “Jessica” g6k lattice sieving tools

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
4. Full best-effort implementation on powerful hardware
5. Wrench attacks

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
4. Full best-effort implementation on powerful hardware
5. Wrench attacks



https://xkcd.com/538

Concreteness and measuring the cost of attacks

1. Asymptotic number of bit operations and memory accesses
2. Can be refined to expensive-memory model
3. Hardware implementations of specific subroutines
4. Full best-effort implementation on powerful hardware
5. Wrench attacks

This work

- We implemented a GPU accelerated isogeny-graph explorer, that can navigate to the $\mathbb{F}_p$ subgraph from a random starting curve in a few hours working over [redacted]-bit generic primes
- We also have machinery to perform vectorisation over $\mathbb{F}_p$
- ...and from this, to recover the full endomorphism ring in practically efficient time

Supersingular isogeny problems

Supersingular isogeny problems

Supersingular Isogeny Path Problem

Given two supersingular elliptic curves E_1, E_2 find an ℓ^k-isogeny $E_1 \to E_2$

Supersingular Endomorphism Problem

Given a supersingular elliptic curve E, compute a basis of the endomorphism ring

Supersingular isogeny problems

Supersingular Isogeny Path Problem

Given two supersingular elliptic curves E_1, E_2 find an ℓ^k-isogeny $E_1 \to E_2$

Supersingular Endomorphism Problem

Given a supersingular elliptic curve E, compute a basis of the endomorphism ring

They are equivalent ... but how would one solve them *in practice*?

Attacks against supersingular isogeny problems

Meet in the Middle (for isogeny path problem)

Idea

Between two random curves E_1, E_2 there exists an ℓ^k isogeny with $k = O(\log_\ell(p))$

Attacks against supersingular isogeny problems

Meet in the Middle (for isogeny path problem)

Idea

Between two random curves E_1, E_2 there exists an ℓ^k isogeny with $k = O(\log_\ell(p))$

Algorithm

Compute walks of length $\log_\ell(p)/2$ from E_1, E_2 until they meet

Attacks against supersingular isogeny problems

Meet in the Middle (for isogeny path problem)

Idea

Between two random curves E_1, E_2 there exists an ℓ^k isogeny with $k = O(\log_\ell(p))$

Algorithm

Compute walks of length $\log_\ell(p)/2$ from E_1, E_2 until they meet

Cost

$\tilde{O}(p^{1/2})$

Attacks against supersingular isogeny problems

Meet in the Middle (for isogeny path problem)

Idea

Between two random curves E_1, E_2 there exists an ℓ^k isogeny with $k = O(\log_\ell(p))$

Algorithm

Compute walks of length $\log_\ell(p)/2$ from E_1, E_2 until they meet

Cost

$\tilde{O}(p^{1/2})$

Concretely

Requires $\tilde{O}(p^{1/2})$ memory and search is hard to parallelise

Attacks against supersingular isogeny problems

Meet in the Middle (for isogeny path problem)

Idea

Between two random curves E_1, E_2 there exists an ℓ^k isogeny with $k = O(\log_\ell(p))$

Algorithm

Compute walks of length $\log_\ell(p)/2$ from E_1, E_2 until they meet

Cost

$\tilde{O}(p^{1/2})$

Concretely

Requires $\tilde{O}(p^{1/2})$ memory and search is hard to parallelise

Can be improved with van Oorschot-Wiener Golden Collision Search

...to give cost $\tilde{O}(p^{3/4}/M^{1/2}/C)$ time and $\tilde{O}(M)$ memory using C cores

Attacks against supersingular isogeny problems

Delfs-Galbraith (for isogeny path problem)

Attacks against supersingular isogeny problems

Delfs-Galbraith (for isogeny path problem)

Idea

The $\mathbb{F}_p$-subgraph is of size $\tilde{O}(p^{1/2})$ and once we find it, we can stay inside it

Attacks against supersingular isogeny problems

Delfs-Galbraith (for isogeny path problem)

Idea

The $\mathbb{F}_p$-subgraph is of size $\tilde{O}(p^{1/2})$ and once we find it, we can stay inside it

Algorithm

Do random ℓ-isogeny walks $E_1 \to \cdots \to E_1', E_2 \to \cdots \to E_2'$ until E_1', E_2' are over $\mathbb{F}_p$

Find isogeny over $\mathbb{F}_p$ between $E_1' \to E_2'$ (e.g. with vOW Golden Collision search)

Attacks against supersingular isogeny problems

Delfs-Galbraith (for isogeny path problem)

Idea

The $\mathbb{F}_p$-subgraph is of size $\tilde{O}(p^{1/2})$ and once we find it, we can stay inside it

Algorithm

Do random ℓ-isogeny walks $E_1 \to \cdots \to E_1', E_2 \to \cdots \to E_2'$ until E_1', E_2' are over $\mathbb{F}_p$

Find isogeny over $\mathbb{F}_p$ between $E_1' \to E_2'$ (e.g. with vOW Golden Collision search)

Cost

$\tilde{O}(p^{1/2}) + \tilde{O}(p^{3/8}/M^{1/2})$ time and $\tilde{O}(M)$ memory

Attacks against supersingular isogeny problems

Delfs-Galbraith (for isogeny path problem)

Idea

The $\mathbb{F}_p$-subgraph is of size $\tilde{O}(p^{1/2})$ and once we find it, we can stay inside it

Algorithm

Do random ℓ-isogeny walks $E_1 \to \cdots \to E_1', E_2 \to \cdots \to E_2'$ until E_1', E_2' are over $\mathbb{F}_p$

Find isogeny over $\mathbb{F}_p$ between $E_1' \to E_2'$ (e.g. with vOW Golden Collision search)

Cost

$\tilde{O}(p^{1/2}) + \tilde{O}(p^{3/8}/M^{1/2})$ time and $\tilde{O}(M)$ memory

Concretely

First stage is naturally memoryless. Memory for second stage can be configured

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{\Delta}])$

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism
- ...of trace zero, and so induces orientation by $\mathbb{Z}[\sqrt{-dp}]$ on E (Converse is also true)

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism
- ...of trace zero, and so induces orientation by $\mathbb{Z}[\sqrt{-dp}]$ on E (Converse is also true)
- ...this has class number $\tilde{O}((dp)^{1/2})$, so good probability of finding these curves

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism
- ...of trace zero, and so induces orientation by $\mathbb{Z}[\sqrt{-dp}]$ on E (Converse is also true)
- ...this has class number $\tilde{O}((dp)^{1/2})$, so good probability of finding these curves
- We can detect whether ψ exists by checking $\Phi_d(j(E), j(E)^{(p)}) \stackrel{?}{=} 0$

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism
- ...of trace zero, and so induces orientation by $\mathbb{Z}[\sqrt{-dp}]$ on E (Converse is also true)
- ...this has class number $\tilde{O}((dp)^{1/2})$, so good probability of finding these curves
- We can detect whether ψ exists by checking $\Phi_d(j(E), j(E)^{(p)}) \stackrel{?}{=} 0$
- ...forces small d in practice

Attacks against supersingular isogeny problems

Generalised Delfs-Galbraith (for isogeny path problem)

Idea Replace $\mathbb{F}_p$-subgraph with oriented subgraph $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{\Delta}])$

However knowing an orientation is tantamount to knowing an endomorphism

- If you can (easily) find endomorphisms of a given curve, you can solve the isogeny problem already (OneEnd ⇔ EndRing ⇔ IsogenyPathProblem)

What orientations are easy to detect?

- We know the Frobenius isogeny for free $\pi : E \to E^{(p)}$
- If we have d-isogeny $\psi : E \to E^{(p)}$, then $\hat{\pi}\psi$ is an endomorphism
- ...of trace zero, and so induces orientation by $\mathbb{Z}[\sqrt{-dp}]$ on E (Converse is also true)
- ...this has class number $\tilde{O}((dp)^{1/2})$, so good probability of finding these curves
- We can detect whether ψ exists by checking $\Phi_d(j(E), j(E)^{(p)}) \stackrel{?}{=} 0$
- ...forces small d in practice

Cost $\tilde{O}(p/h(\mathbb{Z}[\sqrt{-dp}])) = \tilde{O}(p^{1/2})$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1), \mathbf{End}(E_2)$ using ideal-to-isogeny)

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1), \mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathrm{End}(E_1), \mathrm{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{-dp}])$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1)$, $\mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\varphi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1), \mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \dots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\varphi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(…Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1), \mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\varphi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$
- (Proven) Subexponential classical post-processing to then compute $\mathrm{End}(E)$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1)$, $\mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\phi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$
- (Proven) Subexponential classical post-processing to then compute $\mathrm{End}(E)$
- (Heuristic) $O(1)$ calls suffice to compute $\mathrm{End}(E)$ efficiently

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1)$, $\mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\phi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$
- (Proven) Subexponential classical post-processing to then compute $\mathrm{End}(E)$
- (Heuristic) $O(1)$ calls suffice to compute $\mathrm{End}(E)$ efficiently
- (Experimental) 4 calls suffice to compute $\mathrm{End}(E)$ efficiently

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1)$, $\mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{ss}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\varphi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$
- (Proven) Subexponential classical post-processing to then compute $\mathrm{End}(E)$
- (Heuristic) $O(1)$ calls suffice to compute $\mathrm{End}(E)$ efficiently
- (Experimental) 4 calls suffice to compute $\mathrm{End}(E)$ efficiently

Cost $\tilde{O}(p/h(\mathbb{Z}[\sqrt{-dp}])) = \tilde{O}(p^{1/2})$

Attacks against supersingular isogeny problems

Inseparable Endomorphisms (for endomorphism ring problem)

Idea

Collect lollipops to compute the endomorphism ring of E

(...Can then obtain isogeny $E_1 \to E_2$ from connecting ideal of $\mathbf{End}(E_1), \mathbf{End}(E_2)$ using ideal-to-isogeny)

Algorithm

- Walk through ℓ_j-isogeny graph with $\varphi : E \to \ldots \to F$ until F in $\mathrm{Ell}_p^{SS}(\mathbb{Z}[\sqrt{-dp}])$
  Let $\psi : F \to F^{(p)}$ be a d-isogeny then $\varphi = \hat{\varphi}\hat{\pi}\psi\varphi \in \mathrm{End}(E)$
- With different ℓ_j, two of these endomorphisms give a Bass order in $\mathrm{End}(E)$
- (Proven) Subexponential classical post-processing to then compute $\mathrm{End}(E)$
- (Heuristic) $O(1)$ calls suffice to compute $\mathrm{End}(E)$ efficiently
- (Experimental) 4 calls suffice to compute $\mathrm{End}(E)$ efficiently

Cost $\tilde{O}(p/h(\mathbb{Z}[\sqrt{-dp}])) = \tilde{O}(p^{1/2})$

Concretely Memoryless, but requires multiple calls to first step (and larger ℓ_j)

Attacks against supersingular isogeny problems

1. Meet in the middle Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(p^{1/2})$
2. vOW Golden Collision Finding Time $\tilde{O}(p^{3/4}/M^{1/2})$, Memory $\tilde{O}(M)$
3. (Generalised) Delfs-Galbraith Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(M)$
4. Inseparable Endomorphisms Time $\tilde{O}(p^{1/2})$, Memory $o(1)$

Attacks against supersingular isogeny problems

1. Meet in the middle Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(p^{1/2})$
2. vOW Golden Collision Finding Time $\tilde{O}(p^{3/4}/M^{1/2})$, Memory $\tilde{O}(M)$
3. (Generalised) Delfs-Galbraith Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(M)$
4. Inseparable Endomorphisms Time $\tilde{O}(p^{1/2})$, Memory $o(1)$

Irrespective of post-processing need efficient graph exploration for 3. and 4.

Attacks against supersingular isogeny problems

1. Meet in the middle Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(p^{1/2})$
2. vOW Golden Collision Finding Time $\tilde{O}(p^{3/4}/M^{1/2})$, Memory $\tilde{O}(M)$
3. (Generalised) Delfs-Galbraith Time $\tilde{O}(p^{1/2})$, Memory $\tilde{O}(M)$
4. Inseparable Endomorphisms Time $\tilde{O}(p^{1/2})$, Memory $o(1)$

Irrespective of post-processing need efficient graph exploration for 3. and 4.

This is really the asymptotic bottleneck

...so how do we do this, for real?

Concrete Questions and Improvements

How to traverse the isogeny graph?

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc) [Chi-Domínguez25]

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc)˙ [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc). [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc). [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Which ℓ?

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc) [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Which ℓ?

- Need $O(\ell^3)$ $\mathbb{F}_p$-multiplications to compute a root of a degree-ℓ polynomial
  ...so $O(\ell^2)$ $\mathbb{F}_p$-multiplications per node visited
  ...so choose $\ell = 2$

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc) [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Which ℓ?

- Need $O(\ell^3)$ $\mathbb{F}_p$-multiplications to compute a root of a degree-ℓ polynomial
  ...so $O(\ell^2)$ $\mathbb{F}_p$-multiplications per node visited
  ...so choose $\ell = 2$

Better detection i.e. SuperSolver [Corte-Real Santos-Costello-Shi21]

- Core Insight one step costs half a square root
  ...which costs $O(\log(p))$ $\mathbb{F}_p$-multiplications

Concrete Questions and Improvements

How to traverse the isogeny graph?

- Using modular polynomials
- Using torsion (as is available in SQISign etc) [Chi-Domínguez25]
- Using radical isogenies [Chi-Domínguez25]

Which ℓ?

- Need $O(\ell^3)$ $\mathbb{F}_p$-multiplications to compute a root of a degree-ℓ polynomial
  ...so $O(\ell^2)$ $\mathbb{F}_p$-multiplications per node visited
  ...so choose $\ell = 2$

Better detection i.e. SuperSolver [Corte-Real Santos-Costello-Shi21]

- Core Insight one step costs half a square root
  ...which costs $O(\log(p))$ $\mathbb{F}_p$-multiplications
- Therefore any test for orientability that uses constant number of $\mathbb{F}_p$ multiplications will eventually become more efficient as p grows

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Naively

checking whether ℓ-neighbours of E are d-isogenous to their $\mathbb{F}_p$-conjugate

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Naively

checking whether ℓ-neighbours of E are d-isogenous to their $\mathbb{F}_p$-conjugate

- Find neighbours: compute solutions of $\Phi_\ell(j(E), z) = 0 \pmod p$
- Is oriented: verify whether any solution z satisfies $\Phi_d(z, z^p) = 0 \pmod p$

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Idea Avoid root computations

$$
\begin{aligned}
&\Phi_\ell(j(E), z) = 0, \quad \Phi_d(z, z^p) = 0 \\
\rightsquigarrow\ &\Phi_\ell(j(E), x + \tau y) = 0, \quad \Phi_d(x + \tau y, x - \tau y) = 0
\end{aligned}
$$

Now write

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Idea Avoid root computations

$$\Phi_\ell(j(E), z) = 0, \quad \Phi_d(z, z^p) = 0$$

$$\rightsquigarrow \Phi_\ell(j(E), x + \tau y) = 0, \quad \Phi_d(x + \tau y, x - \tau y) = 0$$

Now write

$$\Phi_\ell(j(E), x + \tau y) = f_0(x, y) + \tau f_1(x, y) \quad \text{where} \quad f_0, f_1 \in \mathbb{F}_p[x, y]$$

$$\Phi_d(x + \tau y, x - \tau y) = h_0(x, y) \quad \text{where} \quad h_0 \in \mathbb{F}_p[x, y]$$

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Idea Avoid root computations

$$\Phi_\ell(j(E), z) = 0, \quad \Phi_d(z, z^p) = 0$$

$$\rightsquigarrow \Phi_\ell(j(E), x + \tau y) = 0, \quad \Phi_d(x + \tau y, x - \tau y) = 0$$

Now write

$$\Phi_\ell(j(E), x + \tau y) = f_0(x, y) + \tau f_1(x, y) \quad \text{where} \quad f_0, f_1 \in \mathbb{F}_p[x, y]$$

$$\Phi_d(x + \tau y, x - \tau y) = h_0(x, y) \quad \text{where} \quad h_0 \in \mathbb{F}_p[x, y]$$

Then compute the resultant

$$r_0(y) = \mathrm{Res}_x(f_0, h_0), \quad r_1(y) = \mathrm{Res}_x(f_1, h_0)$$

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Idea Avoid root computations

$$\Phi_\ell(j(E), z) = 0, \quad \Phi_d(z, z^p) = 0$$
$$\rightsquigarrow \Phi_\ell(j(E), x + \tau y) = 0, \quad \Phi_d(x + \tau y, x - \tau y) = 0$$

Now write

$$\Phi_\ell(j(E), x + \tau y) = f_0(x, y) + \tau f_1(x, y) \quad \text{where} \quad f_0, f_1 \in \mathbb{F}_p[x, y]$$
$$\Phi_d(x + \tau y, x - \tau y) = h_0(x, y) \quad \text{where} \quad h_0 \in \mathbb{F}_p[x, y]$$

Then compute the resultant

$$r_0(y) = \mathrm{Res}_x(f_0, h_0), \quad r_1(y) = \mathrm{Res}_x(f_1, h_0)$$

$r_0(y) = r_1(y) = 0$ has $\mathbb{F}_p$-solution $\iff$ $\Phi_\ell(j(E), z) = \Phi_d(z, z^p) = 0$ has $\mathbb{F}_{p^2}$-solution

NeighbourIsOriented

NeighbourInFp + Generalised Delfs-Galbraith

Idea Avoid root computations

$$\Phi_\ell(j(E), z) = 0, \quad \Phi_d(z, z^p) = 0$$
$$\rightsquigarrow \Phi_\ell(j(E), x + \tau y) = 0, \quad \Phi_d(x + \tau y, x - \tau y) = 0$$

Now write

$$\Phi_\ell(j(E), x + \tau y) = f_0(x, y) + \tau f_1(x, y) \quad \text{where} \quad f_0, f_1 \in \mathbb{F}_p[x, y]$$
$$\Phi_d(x + \tau y, x - \tau y) = h_0(x, y) \quad \text{where} \quad h_0 \in \mathbb{F}_p[x, y]$$

Then compute the resultant

$$r_0(y) = \mathrm{Res}_x(f_0, h_0), \quad r_1(y) = \mathrm{Res}_x(f_1, h_0)$$

$r_0(y) = r_1(y) = 0$ has $\mathbb{F}_p$-solution $\iff$ $\Phi_\ell(j(E), z) = \Phi_d(z, z^p) = 0$ has $\mathbb{F}_{p^2}$-solution

Finally check whether $r_0(y) = r_1(y) = 0$ has solution by computing $\gcd(r_0(y), r_1(y))$

Generalised NeighbourInFp: NeighbourIsOriented

Indeed this generalises NeighbourInFp

1. $\ell = 1, d = 1$ Checks whether E is in $\mathbb{F}_p$
2. $\ell > 1, d = 1$ Checks whether E has ℓ-neighbours in $\mathbb{F}_p$ (SuperSolver)
3. $\ell = 1, d > 1$ Checks whether E is $\mathbb{Z}[\sqrt{-dp}]$-oriented (Generalised Delfs-Galbraith)
4. $\ell > 1, d > 1$ Checks whether E has ℓ-neighbour that is $\mathbb{Z}[\sqrt{-dp}]$-oriented

Generalised NeighbourInFp: NeighbourIsOriented

Indeed this generalises NeighbourInFp

1. $\ell = 1, d = 1$ Checks whether E is in $\mathbb{F}_p$
2. $\ell > 1, d = 1$ Checks whether E has ℓ-neighbours in $\mathbb{F}_p$ (SuperSolver)
3. $\ell = 1, d > 1$ Checks whether E is $\mathbb{Z}[\sqrt{-dp}]$-oriented (Generalised Delfs-Galbraith)
4. $\ell > 1, d > 1$ Checks whether E has ℓ-neighbour that is $\mathbb{Z}[\sqrt{-dp}]$-oriented

Initial analysis indicates that 4. is likely too expensive (Work in progress!)

Resultant algorithms are more complex to analyse

Generalised NeighbourInFp: NeighbourIsOriented

Indeed this generalises NeighbourInFp

1. $\ell = 1, d = 1$ Checks whether E is in $\mathbb{F}_p$
2. $\ell > 1, d = 1$ Checks whether E has ℓ-neighbours in $\mathbb{F}_p$ (SuperSolver)
3. $\ell = 1, d > 1$ Checks whether E is $\mathbb{Z}[\sqrt{-dp}]$-oriented (Generalised Delfs-Galbraith)
4. $\ell > 1, d > 1$ Checks whether E has ℓ-neighbour that is $\mathbb{Z}[\sqrt{-dp}]$-oriented

Initial analysis indicates that 4. is likely too expensive (Work in progress!)

Resultant algorithms are more complex to analyse

There is interesting concurrent work that might improve on this

Minimising costs of checking NeighbourIsOriented

Given a set of tests $T_{\ell,c}$, we must minimise

$$\min_{I}\left(\frac{\mathrm{cost}(\sqrt{p})/2 + \sum_{(\ell,c)\in I} \mathrm{cost}(T_{\ell,c})}{\sum_{(\ell,c)\in I} \mathrm{p.success}(T_{\ell,c})}\right)$$

Minimising costs of checking NeighbourIsOriented

Given a set of tests $T_{\ell,c}$, we must minimise

$$\min_I \left(\frac{\mathrm{cost}(\sqrt{p})/2 + \sum_{(\ell,c)\in I} \mathrm{cost}(T_{\ell,c})}{\sum_{(\ell,c)\in I} \mathrm{p.success}(T_{\ell,c})} \right)$$

For a small set of $\ell, c = 1, 2, \ldots$, the costs and success probabilities can be explicitly computed (In theory...)

Minimising costs of checking NeighbourIsOriented

Given a set of tests $T_{\ell,c}$, we must minimise

$$\min_I \left(\frac{\mathrm{cost}(\sqrt{p})/2 + \sum_{(\ell,c)\in I} \mathrm{cost}(T_{\ell,c})}{\sum_{(\ell,c)\in I} \mathrm{p.success}(T_{\ell,c})} \right)$$

For a small set of $\ell, c = 1, 2, \ldots$, the costs and success probabilities can be explicitly computed (In theory...)

Example Supersolver did this for
$c = 1, \ell = 3, 5, 7, 9, 11, 13, 17, 19, 15, 23, 25, 29, 21, 31, 27, 37, 41, 43, 33, 35$

Minimising costs of checking NeighbourIsOriented

Given a set of tests $T_{\ell,c}$, we must minimise

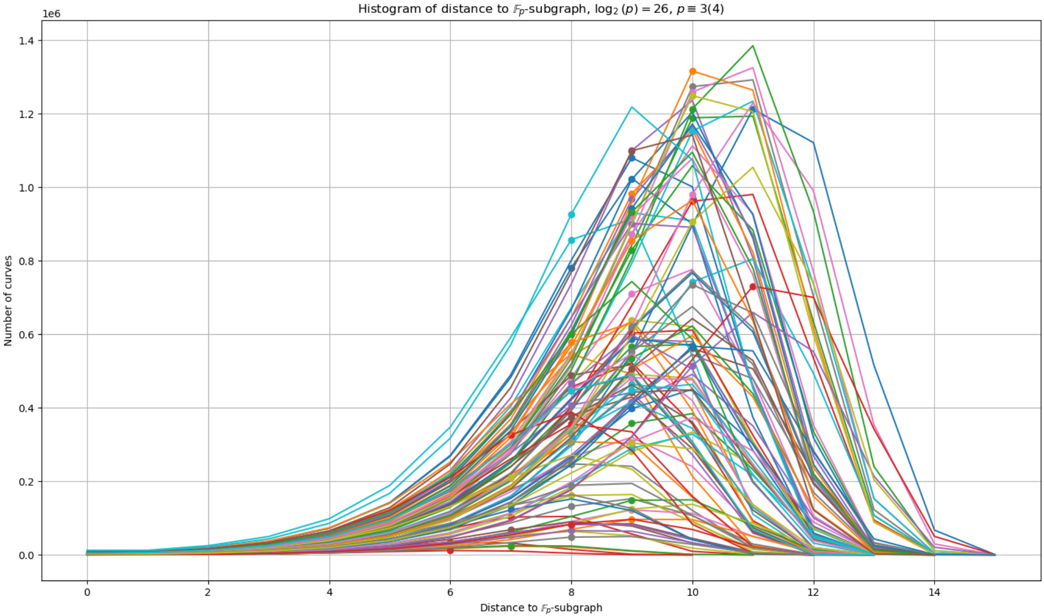
$$\min_I \left(\frac{\text{cost}(\sqrt{p})/2 + \sum_{(\ell,c)\in I} \text{cost}(T_{\ell,c})}{\sum_{(\ell,c)\in I} \text{p.success}(T_{\ell,c})} \right)$$

For a small set of $\ell, c = 1, 2, ...$, the costs and success probabilities can be explicitly computed (In theory...)

Example Supersolver did this for
$c = 1, \ell = 3, 5, 7, 9, 11, 13, 17, 19, 15, 23, 25, 29, 21, 31, 27, 37, 41, 43, 33, 35$

We are trying to get better estimates for both asymptotic and concrete costs (Work in progress!)

Bad Neighbourhoods and Rerandomisation: Dandelions



Why GPUs

Why GPUs

What is a GPU?

Example NVIDIA L40s

- 19,000 cuda cores
- 32 bit architecture
- Streaming multi-processors each manage 128 cuda cores
- ~50 MB Cache, ~50 GB RAM
- CUDA software allows for writing code for both CPU and GPU simultaneously

Why GPUs

What is a GPU?

Example NVIDIA L40s

- 19,000 cuda cores
- 32 bit architecture
- Streaming multi-processors each manage 128 cuda cores
- ~50 MB Cache, ~50 GB RAM
- CUDA software allows for writing code for both CPU and GPU simultaneously

What can('t) you do with a gpu?

- SIMD = Single Instruction Multiple Data (data dependent branching is bad)
- RAM Thrashing/ Cache Limitations

Why GPUs

What is a GPU?

Example NVIDIA L40s

- 19,000 cuda cores
- 32 bit architecture
- Streaming multi-processors each manage 128 cuda cores
- ~50 MB Cache, ~50 GB RAM
- CUDA software allows for writing code for both CPU and GPU simultaneously

What can('t) you do with a gpu?

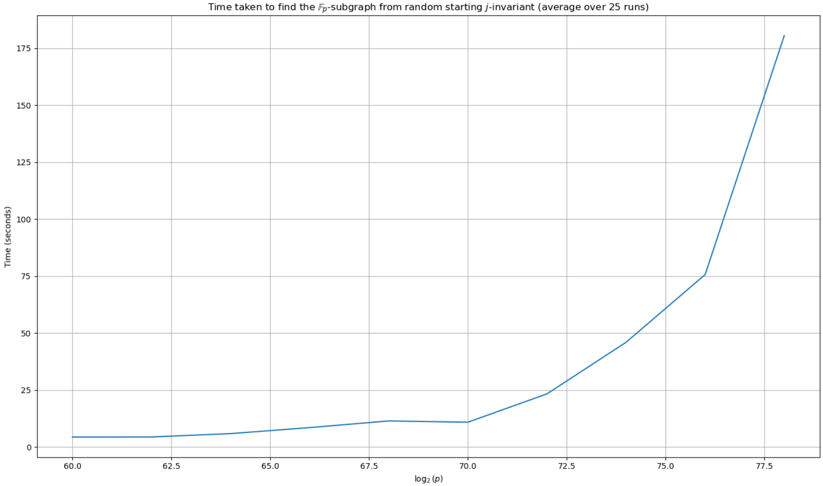
- SIMD = Single Instruction Multiple Data (data dependent branching is bad)
- RAM Thrashing/ Cache Limitations

GPUs aren't just fast

- Specialised hardware: NSA will do similar things
- Similar problems (limited memory, simple logic, not superscalar)
- Close(r) to silicon

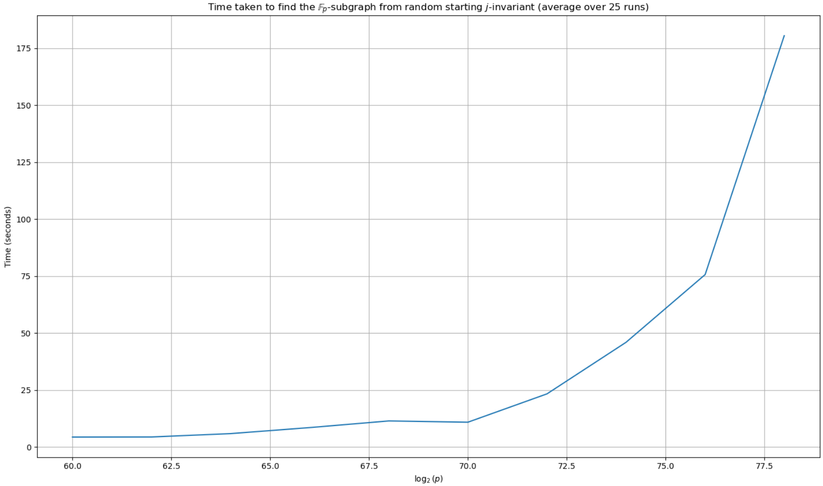
Some timings

Very Preliminary timings (WIP!)



Some timings

Very Preliminary timings (WIP!)



Larger instances We can break 95bit instances in just a few hours on average

Many Questions and Todos

Many Questions and Todos

Will cache turn out to really be a problem?

Many Questions and Todos

Will cache turn out to really be a problem?

Clustering of the $\mathbb{F}_p$ subgraph (Adventures in Supersingularland)

Many Questions and Todos

Will cache turn out to really be a problem?

Clustering of the $\mathbb{F}_p$ subgraph (Adventures in Supersingularland)

Can we efficiently detect oriented neighbours?

Many Questions and Todos

Will cache turn out to really be a problem?

Clustering of the $\mathbb{F}_p$ subgraph (Adventures in Supersingularland)

Can we efficiently detect oriented neighbours?

Use of radical isogenies and torsion to walk through the graph [Chi-Domínguez25]

Summary

Practical

We wrote a GPU accelerated implementation of the SuperSolver variant of the Delfs-Galbraith attack to break instances of the supersingular isogeny problem for 95 bit generic primes in a few hours

This attack would cost ~30 USD on AWS and ~5 USD in the Hetztner cloud

Theoretical

We combined the theory of SuperSolver's NeighbourInFp detection with Generalised Delfs-Galbraith to obtain a NeighbourIsOriented detection subroutine, but we think that this turns out to be an unfavourable generalisation

Thank you for your attention

Slides https://rueg.re/lid25

RFC: SQISign Challenges

Many schemes have public challenges that give prize money for breaking their non-cryptographic parameters (e.g. RSA, SIKE)

Perhaps we want to do this for SQISign? Or give a general isogeny challenge?

An apparent obstruction is the trusted setup required to generate these challenges and then a zero-knowledge protocol to prove that a solution has been found

Hopefully we will soon have a better idea of how expensive attacks are in practice, and can start thinking about setting challenges