Parallel strategies for SIDH:

Towards computing SIDH twice as fast

Francisco Rodríguez-Henríquez

Cinvestav

Joint work with:
Daniel Cervantes-Vázquez, José Eduardo Ochoa-Jiménez

School of Mathematics and Statistics
Carleton University

May 20, 2020

Motivation

Motivation

- In 2011, Jao and De Feo proposed the Supersingular Isogeny-based Diffie-Hellman key exchange protocol (SIDH).
- The Supersingular Isogeny Key Encapsulation (SIKE) protocol, which can be seen as a descendant of SIDH, is one of the candidates considered in the second round of the NIST post-quantum cryptography standardization project.
- SIDH and SIKE operate on supersingular elliptic curves defined over $\mathbb{F}_{p^2}$, where p is a large prime number of the form, $p = 2^{e_A} 3^{e_B} - 1$.
- The time costs of SIDH and SIKE are dominated by the computation of large smooth-degree isogenies and scalar multiplications.

Motivation

- SIDH has been studied and implemented in an impressive number of recent publications.
- However, until today very few works have been published on the parallel opportunities that the SIDH and SIKE protocols can offer. We are aware of just two works,
  - In [Koziel-Azarderakhsh-Kermani Indocrypt'16], it was reported a hardware implementation that concurrently evaluates the isogeny images of an average of four points, as they became available.
  - In [Hutchinson-Karabina Indocrypt'18], two parallel canonical strategies for computing/evaluating isogenies in multi-core environments were proposed

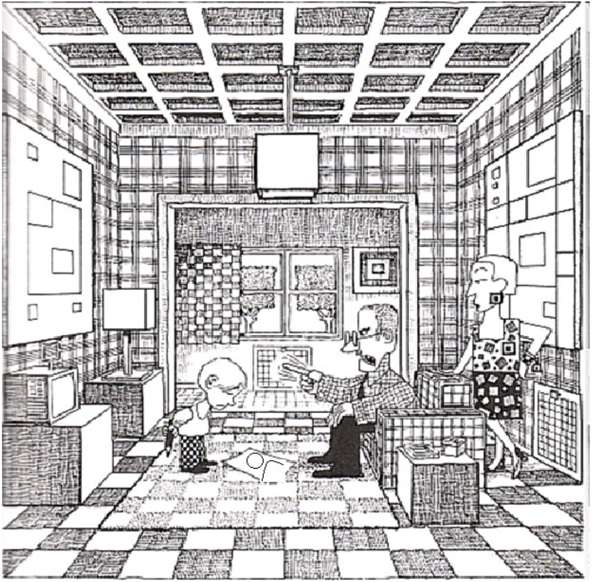
Motivation

- More than 99% of the processors used today have multi-core capabilities.
- Modern multi-core processors come equipped with two or more separate processing units known as cores.
- While most general processors have two, four or eight cores; modern servers are equipped with tens of cores. Moreover, Graphics Processing Unit (GPUs) are many-core architectures equipped with thousands of cores.

Motivation

- In this talk we present two [essentially orthogonal] ideas for exploiting the rich parallelism offered by SIDH
    - We extend the work by [Hutchinson-Karabina Indocrypt'18] to parallelize the whole SIDH protocol considering also the three-point scalar multiplications
    - We propose an extended SIDH (eSIDH) protocol that uses primes allowing more parallelism and faster field arithmetic
- At the end of this talk we present estimates and experimental timings of the combination of these two techniques that for an 8-core processor, achieves an acceleration factor close to two compared against the sequential version of the SIDH and SIKE protocols.

Mathematical Background



Montgomery curves

A Montgomery *elliptic curve* over a finite field $\mathbb{F}_q$ is given by the equation

$$E/\mathbb{F}_q: \quad By^2 = x^3 + Ax^2 + x$$

where $A^2 \neq 4, B \neq 0 \in \mathbb{F}_q$.

$E(\mathbb{F}_q)$ denotes the set of all finite points, i.e. points with coordinates in $\mathbb{F}_q$ that satisfy the equation $x^3 + Ax^2 + x - By^2 = 0$, along with the point at infinity $\mathcal{O}$.

The *j-invariant* $j(E)$ of a curve acts as its fingerprint, and it is given as

$$j(E) = 256\frac{(A^2-3)^3}{A^2-4}.$$

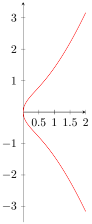
Advantages of Montgomery Curves

Projective Constant Montgomery Curves
[Costello-Longa-Naehrig Crypto'16]:

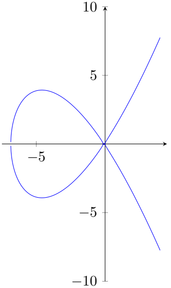
$$E_{(A:C)}/\mathbb{F}_q : Cy^2 = x(Cx^2 + Ax + C).$$

Advantages:

- Allows an x-only arithmetic
- Highly suitable for computing scalar multiplications using Montgomery ladders
- [Costello&Hisil Asiacrypt'17] proposed efficient formulas for computing isogenies between Montgomery curves.



$y^2 = x(x^2 + 1)$



$3y^2 = x(x^2 + 7x + 1)$

Elliptic curve arithmetic

- Scalar multiplication is defined as,

$$[k]P := P + P + \cdots + P, (k-1)\text{(times)}.$$

- The minimum integer m such that $[m]P = \mathcal{O}$ is called the **order** of P.
- The **subgroup generated** by P is the set $\{P, [2]P, [3]P, \ldots, [m-1]P, \mathcal{O}\}$ and is denoted by $\langle P \rangle$.
- The m-**torsion subgroup** is defined as $E[m] = \{P \in E \mid [m]P = \mathcal{O}\}$.

Supersingular elliptic curves

- E is supersingular if

$$\#E(\mathbb{F}_q) = 1 \bmod p.$$

Otherwise E is said to be ordinary.

Basic definitions of isogenies

- An *Isogeny* $\phi : E \to E'$ is a non-trivial homomorphism between elliptic curves given by rational functions. Given P and Q in E then,
  - $\phi(P + Q) = \phi(P) + \phi(Q)$,
  - $\phi(\mathcal{O}) = \mathcal{O}$.
- The *Kernel* of an Isogeny ϕ is the set

$$K = \{P \in E \mid \phi(P) = \mathcal{O}\}.$$

  Note: In this talk the degree of an isogeny is $s := \#K$.
- Let E and E' be two elliptic curves defined over $\mathbb{F}_q$. If there exists an isogeny $\phi : E \to E',$ then we say that E and E' are isogenous.
- If there exists a degree-1 isogeny between E and E' then $j(E) = j(E')$. We say that E and E' are isomorphic. We denote that by $E \cong E'$.

Two curves are isogenous if...

- Tate's theorem states that two elliptic curves E and E' are isogenous over $\mathbb{F}_q$, iff $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$.
- If two elliptic curves E and E' are isogenous over $\mathbb{F}_q$, either both of them are supersingular or both of them are ordinary.

Computing isogenies

- Let E be an elliptic curve and $P \in E$ be an order-m point.
- **Isogeny computation**: there exists an elliptic curve E' and an isogeny $\phi : E \to E'$ such that the *Kernel* of ϕ is $\langle P \rangle$, *i.e.* $\phi(R) = \mathcal{O}$ for each $R \in \langle P \rangle$. We use the notation,

$$E' = E/\langle P \rangle$$

- **Isogeny Evaluation**: Given a point $Q \in E(\mathbb{F}_q)$ such that $Q \notin \mathsf{Ker}(\phi)$, find $\phi(Q)$, *i.e.*, the image of the point Q on E'.

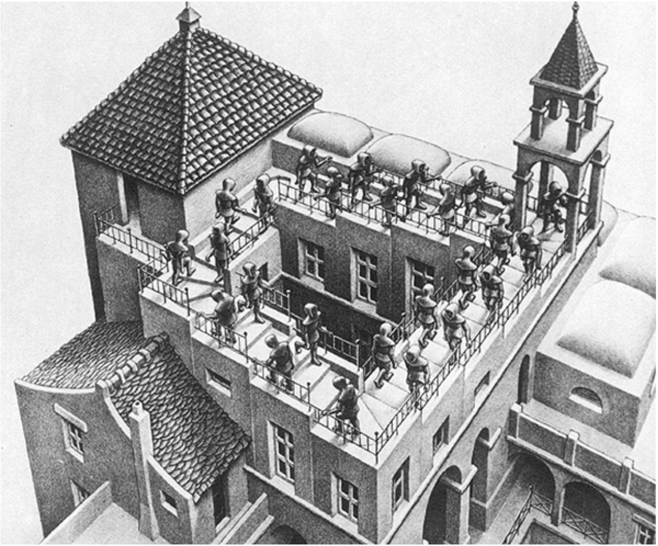
Computing large smooth-degree isogenies

- Given an isogeny $\phi : E \to E'$ of degree ℓ^e then
  - ϕ can be efficiently computed as the composition

$$\phi_{e-1} \circ \phi_{e-2} \circ \cdots \phi_1 \circ \phi_0$$

  where each ϕ_i for $i = 0, \ldots, e-1$ has degree ℓ.

Supersingular Isogeny Diffie Hellman

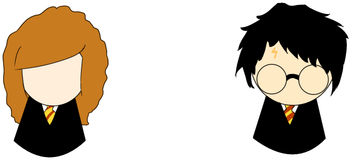


Diffie-Hellman like protocol using isogenies: The SIDH protocol [de Feo-Jao 2011]

SIDH framework:

- Find a prime p of the form $p = 2^{e_{\mathsf{A}}} \cdot 3^{e_{\mathsf{B}}} \cdot f - 1$,
- Let E_0 be a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ with $\#E_0(\mathbb{F}_{p^2}) = (p+1)^2$.

SIDH public parameters



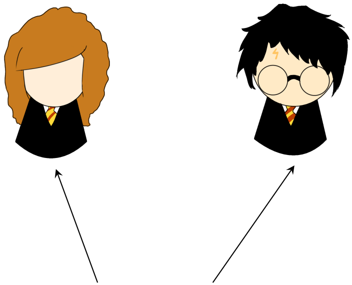
$$\mathsf{p} := 2^{e_{\mathsf{A}}} \cdot 3^{e_{\mathsf{B}}} \cdot f - 1$$

Such that $2^{e_{\mathsf{A}}} \approx 3^{e_{\mathsf{B}}}$

SIDH public parameters

Choose P_{A} and Q_{A}
such that $\langle P_{\mathsf{A}}, Q_{\mathsf{A}} \rangle = E_0[2^{e_{\mathsf{A}}}]$

Choose P_{B} and Q_{B}
such that $\langle P_{\mathsf{B}}, Q_{\mathsf{B}} \rangle = E_0[3^{e_{\mathsf{B}}}]$



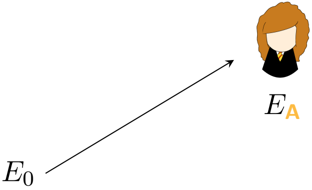
$\mathsf{p} := \quad 2^{e_{\mathsf{A}}} \cdot \quad 3^{e_{\mathsf{B}}} \cdot \quad f - 1$

Such that $2^{e_{\mathsf{A}}} \approx 3^{e_{\mathsf{B}}}$

SIDH protocol

$$K_{\mathsf{A}} := P_{\mathsf{A}} + [m_{\mathsf{A}}]Q_{\mathsf{A}}$$

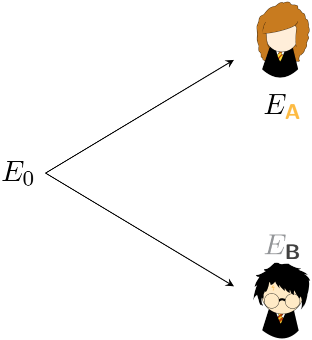
Get ϕ_{A} and $E_{\mathsf{A}} = E_0/\langle K_{\mathsf{A}} \rangle$



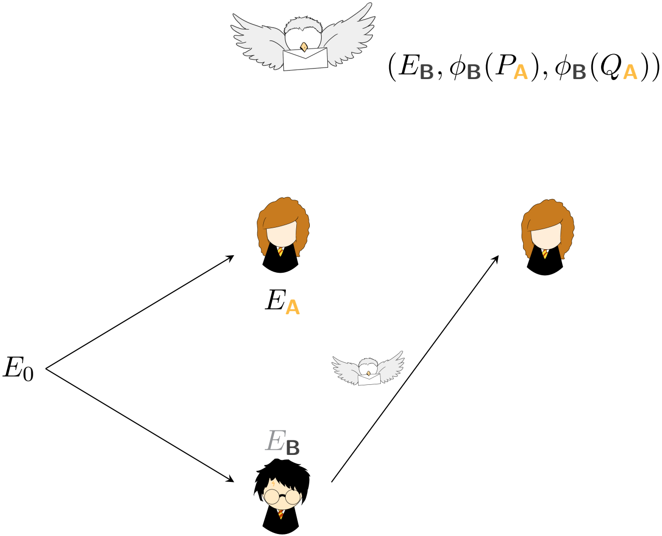
SIDH protocol

$$K_{\mathbf{B}} := P_{\mathbf{B}} + [m_{\mathbf{B}}]Q_{\mathbf{B}}$$

Get $\phi_{\mathbf{B}}$ and $E_{\mathbf{B}} = E_0/K_{\mathbf{B}}$

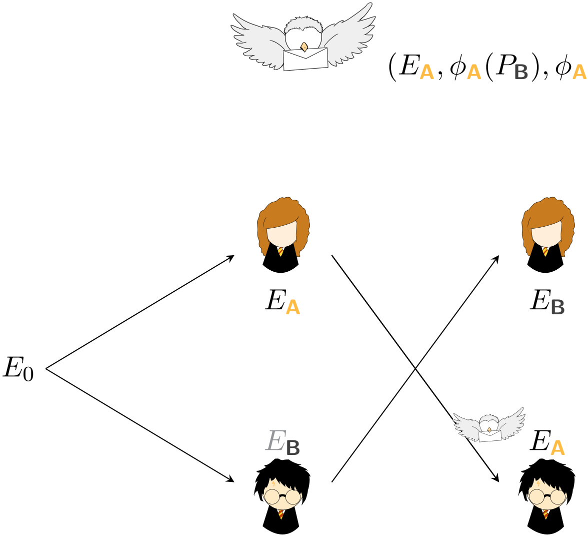


SIDH protocol



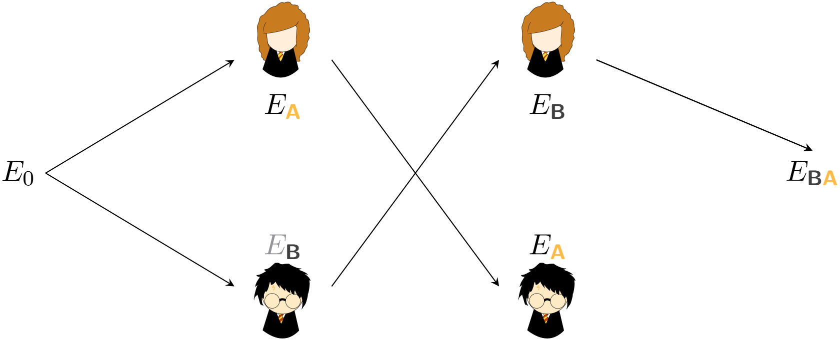
SIDH protocol

$(E_{\mathsf{A}}, \phi_{\mathsf{A}}(P_{\mathsf{B}}), \phi_{\mathsf{A}}(Q_{\mathsf{B}}))$



SIDH protocol

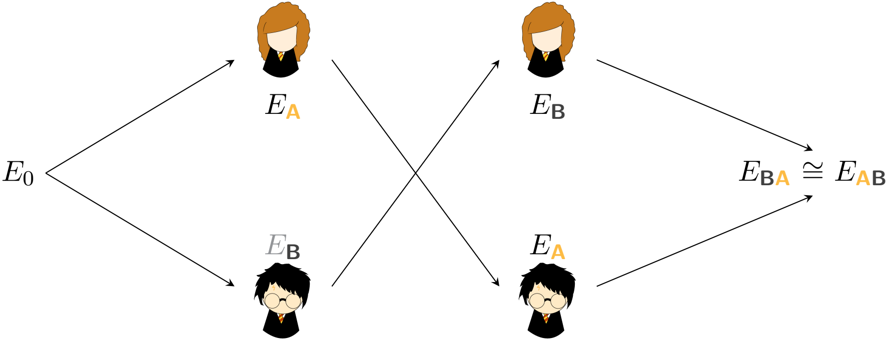
$$K'_{\mathsf{A}} := \phi_{\mathsf{B}}(P_{\mathsf{A}}) + [m_{\mathsf{A}}]\phi_{\mathsf{B}}(Q_{\mathsf{A}})$$
Get $E_{\mathsf{BA}} = E_{\mathsf{B}}/\langle K'_{\mathsf{A}} \rangle$



SIDH protocol

$$K'_{\mathbf{B}} := \phi_{\mathbf{A}}(P_{\mathbf{B}}) + [m_{\mathbf{B}}]\phi_{\mathbf{A}}(Q_{\mathbf{B}})$$

Get $E_{\mathbf{AB}} = E_{\mathbf{A}}/\langle K'_{\mathbf{B}} \rangle$



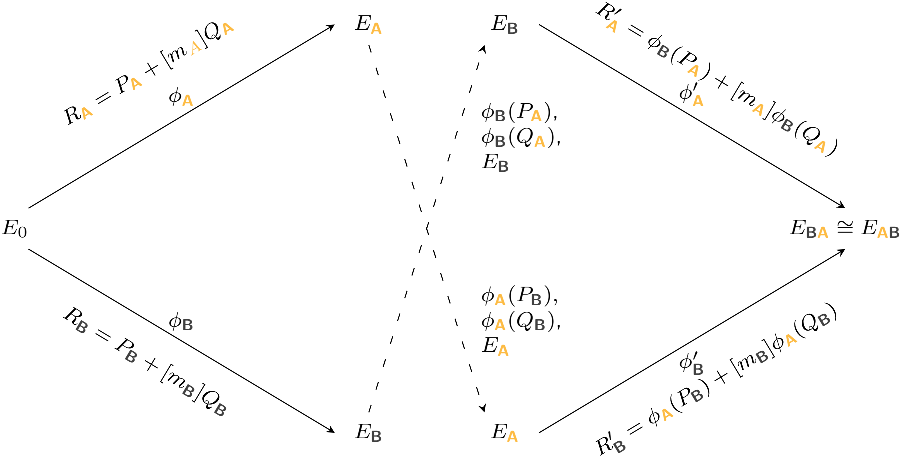


Figure 1: SIDH protocol at a glance

SIDH main building blocks

- The computation of four three-point scalar multiplications of the form

$$R = P + [m]Q$$

  - Can be efficiently computed using the right-to-left Montgomery ladder proposed in [FLOR TC'18] at a per-step cost of

$$\text{xADD} + \text{xDBL} \approx 2\text{xDBL}$$

.

  - Montgomery ladders are not amenable for parallelization
  - They account for about 20-30% of the overall protocol's computational cost

SIDH main building blocks

- The computation of four three-point scalar multiplications of the form

$$R = P + [m]Q$$

  - Can be efficiently computed using the right-to-left Montgomery ladder proposed in [FLOR TC'18] at a per-step cost of

$$\text{xADD} + \text{xDBL} \approx 2\text{xDBL}$$

  .
  - Montgomery ladders are not amenable for parallelization [Really??]
  - They account for about 20-30% of the overall protocol's computational cost

SIDH main building blocks

- The computation of large smooth-degree isogenies and the evaluation of elliptic curve points in those isogenies,
  - Compute the degree-ℓ^e isogeny, $\phi : E \to E'$. Recall that ϕ can be efficiently computed as the composition

$$\phi_{e-1} \circ \phi_{e-2} \circ \cdots \phi_1 \circ \phi_0$$

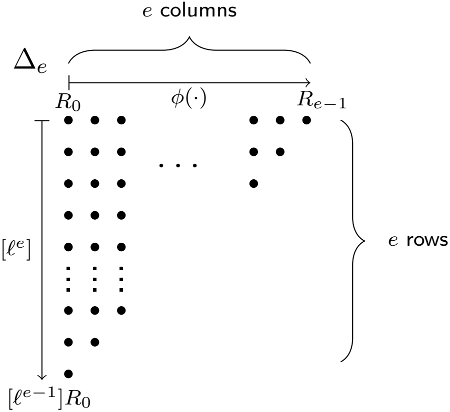
where each ϕ_i for $i = 0, \ldots, e-1$ has degree ℓ.

- These tasks can be efficiently computed using optimal strategies as proposed by [deFeo-Jao-Plût, JMC'14]
- Optimal strategies are highly amenable for parallelization
- They amount for about 70-80% of the overall protocol's computational cost

computing a degree-ℓ^e isogeny $\phi : E \to E'$

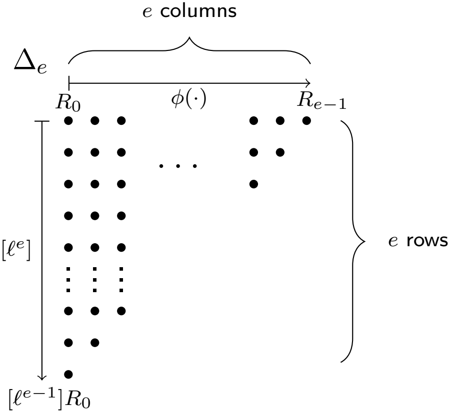
Setting:

- One can compute a degree-ℓ^e isogeny by traversing a weighted directed graph represented as a right triangular lattice Δ_e having $\frac{e(e+1)}{2}$ points distributed in e columns and rows.
- A leaf is defined as the most bottom point in a given column of the lattice.
- The vertices of the graph represent elliptic curve points and its vertical and horizontal edges have a p_ℓ and q_ℓ weight: the costs of performing one scalar multiplication by ℓ and one degree-ℓ isogeny, respectively.



Game objective:

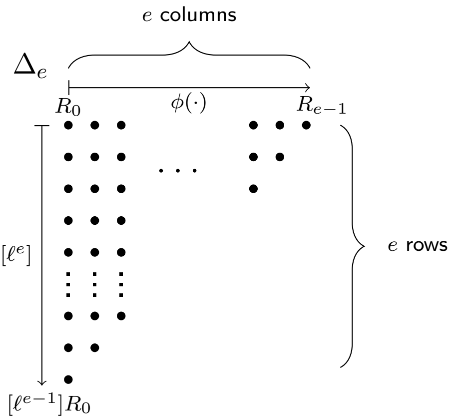
- At the beginning of the isogeny computation, only the point R_0 of order ℓ^e is known.
- The goal of the isogeny computation/evaluation computation is to obtain one by one, all the leaves in Δ_e until the farthest right one, R_{e-1}, has been calculated.
- Then, $\phi : E \to E'$ can be obtained by simply computing a degree-ℓ isogeny with kernel R_{e-1}.



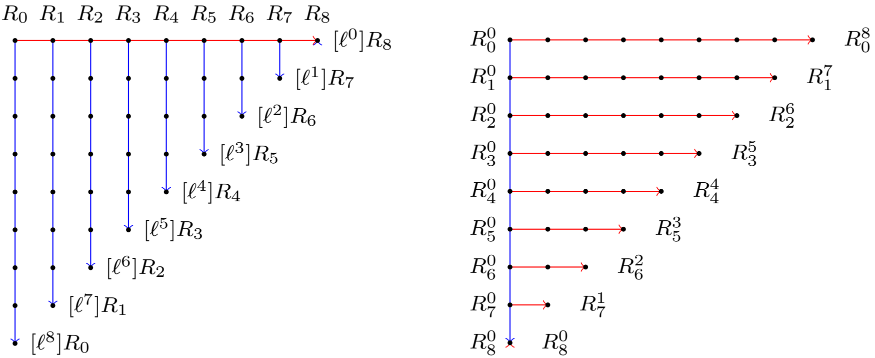
computing a degree-ℓ^e isogeny $\phi : E \to E'$

Rules of the game:

0. Once that you go down you can't go up
1. A Vertical edge corresponds to a scalar multiplication by ℓ
2. A Horizontal edge corresponds to a degree-ℓ isogeny evaluation
3. One cannot compute any horizontal edge unless one has previously reached the leave of the column where you are at
4. All horizontal edges are independent of each other and therefore can be computed in parallel



Example: Two naive strategies

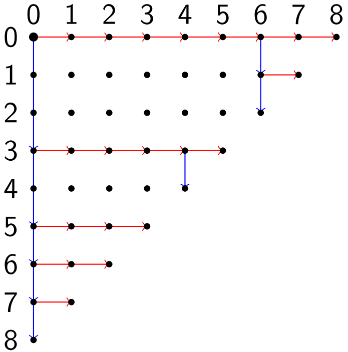


(a) Cost: 8 + 36

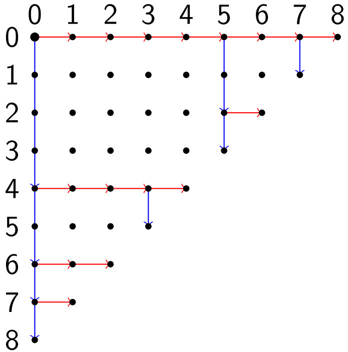
(b) Cost: 36 + 8

Figure 2: Two basic strategies for computing a degree-ℓ^9 isogeny. Both strategies have quadratic complexity in terms of mults. or isogeny evaluations.

Example: Two competitive strategies



(a) Cost: 20 + 11



(b) Cost: 16 + 13

Figure 3: Subfigures 3a and 3b correspond to two more efficient different strategies to traverse Δ_9.

Optimization rules

Definition (Optimal Strategy Problem)

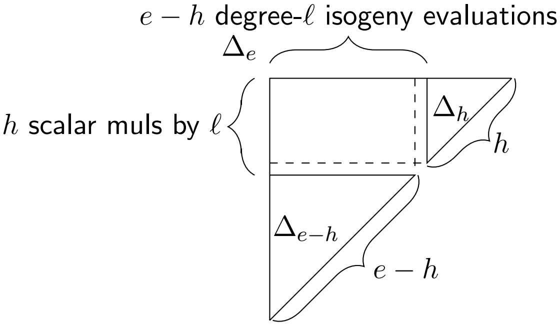
Let Δ_e be the upper-left triangle of the grid of $(e) \times (e)$ vertices, G_e. The Optimal Strategy Problem consists of finding a **legal** directed-rooted-weighted subtree S_{Δ_e} such that:

$$\sum_{E \in \textsf{Edges}(S\Delta_e)} w(E),$$

is minimum, where $w(E)$ is the weight of the edge E.

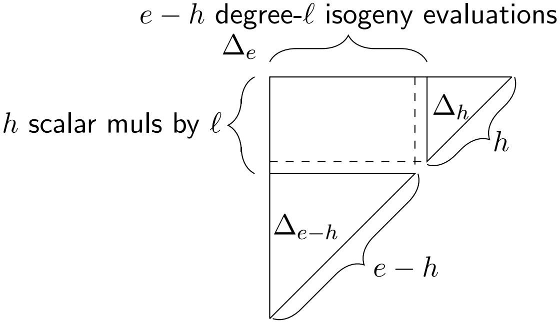
In this case we say that S_{Δ_e} is an optimal strategy to traverse Δ_e.

Optimal strategies [deFeo-Jao-Plût, JMC'14]



- Optimal strategies exploit the fact that a triangle Δ_e can be optimally and recursively decomposed into two sub-triangles Δ_h and Δ_{e-h}

Optimal strategies [deFeo-Jao-Plût, JMC'14]

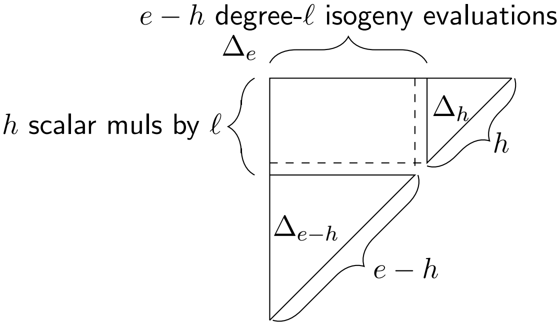


- Let us denote as Δ_e^h the design decision of splitting a triangle Δ_e at row h. Then, the sequential cost of walking through the triangle Δ_e using the cut Δ_e^h is given as,

$$C(\Delta_e^h) = C(\Delta_h) + C(\Delta_{e-h}) + (e-h)\cdot q_\ell + h\cdot p_\ell.$$

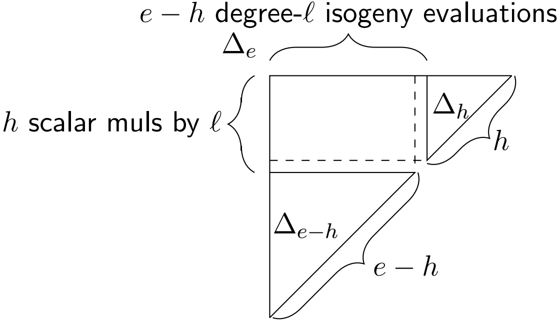
- We say that $\Delta_e^{\hat{h}}$ is optimal if $C(\Delta_e^{\hat{h}})$ is minimal among all Δ_e^h for $h \in [1, e-1]$.

Optimal strategies [deFeo-Jao-Plût, JMC'14]



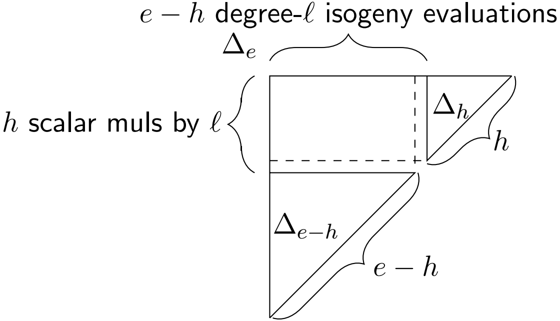
- Applying this strategy recursively leads to a procedure that computes a degree-ℓ^e isogeny at a cost of approximately $\frac{e}{2}\log_2 e$ scalar multiplications by ℓ, $\frac{e}{2}\log_2 e$ degree-ℓ isogeny evaluations, and e computations of degree-ℓ isogenous curves.

Optimal strategies [deFeo-Jao-Plût, JMC'14]



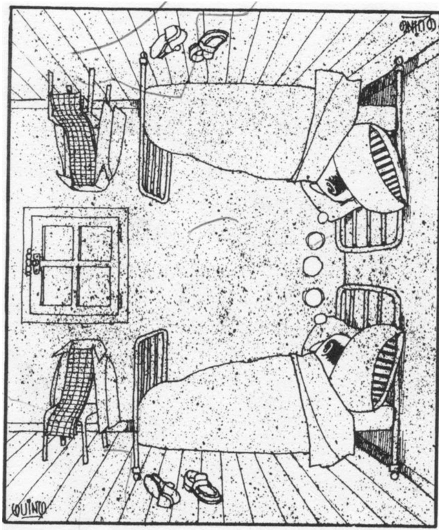
- The optimal strategies presented in [deFeo-Jao-Plût, JMC'14] is one of the major contributions to the SIDH protocol. The authors proved the optimality of their result and virtually all the implementations of the SIDH and SIKE protocols have adopted them.

Optimal strategies [deFeo-Jao-Plût, JMC'14]



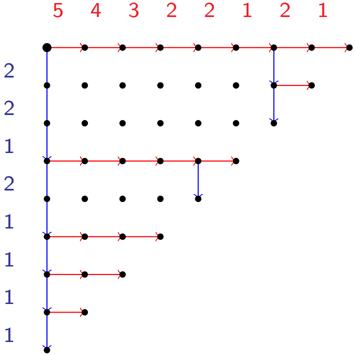
- The optimal strategies presented in [deFeo-Jao-Plût, JMC'14] stand as one of the major contributions to the SIDH protocol. The authors proved the optimality of their result and virtually all the implementations of the SI(DH/KE) protocol have adopted them. Nevertheless... these strategies must be revisited for parallel multi-core environments!

Parallel computations of SIDH

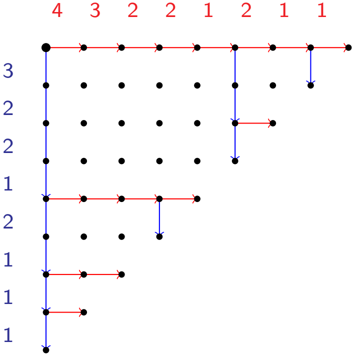


Computing large smooth degree isogenies in parallel environments

Single Core



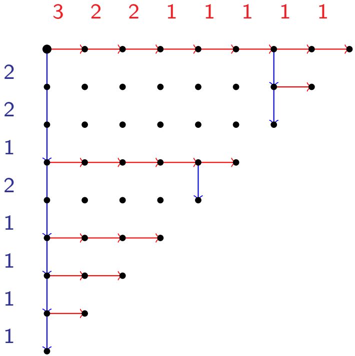
(a) Cost: 20 + 11



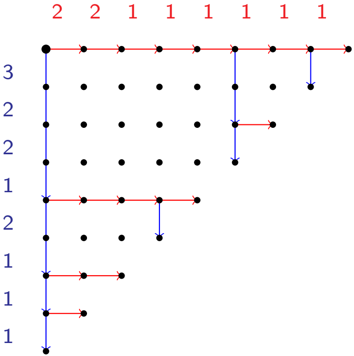
(b) Cost: 16 + 13

Computing large smooth degree isogenies in parallel environments

Two Cores



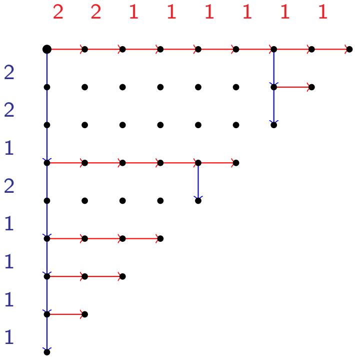
(c) Cost: 12 + 11



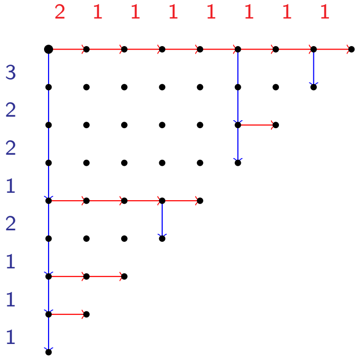
(d) Cost: 10 + 13

Computing large smooth degree isogenies in parallel environments

Three Cores



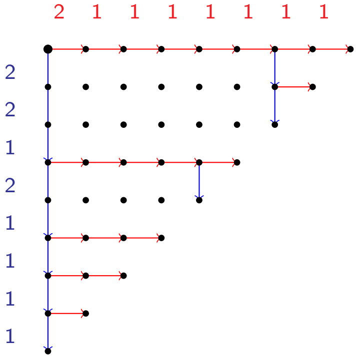
(e) Cost: 11 + 11



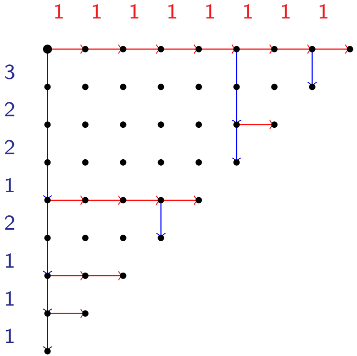
(f) Cost: 10 + 13

Computing large smooth degree isogenies in parallel environments

Four Cores



(g) Cost: 10 + 11



(h) Cost: 9 + 13

Parallel Optimal Strategy Problem

Proposition (Parallel Optimal Strategy Problem)

Let q_ℓ be the timing cost associated to the computation of a degree-ℓ isogeny. Let us define a set of horizontal edges for a fixed index $j \in \{0, 1, \ldots, e-2\}$ by $Col_j(S_{\Delta_e}) = \{[(i, j), (i, j+1)] \in S_{\Delta_e} \mid i \in [0, e-j-2]\}$. The timing cost of computing all horizontal edges in $Col_j(S_{\Delta_e})$ using k cores is of

$$\left\lceil \frac{\#Col_j(S_{\Delta_e})}{k} \right\rceil \cdot q_\ell$$

Parallel Optimal Strategy Problem

Proposition (Parallel Optimal Strategy Problem)

Let q_ℓ *be the timing cost associated to the computation of a degree-*ℓ *isogeny. Let us define a set of horizontal edges for a fixed index* $j \in \{0, 1, \ldots, e-2\}$ *by* $Col_j(S_{\Delta_e}) = \{[(i, j), (i, j+1)] \in S_{\Delta_e} \mid i \in [0, e-j-2]\}$*. The timing cost of computing all horizontal edges in* S_{Δ_e} *using* k *cores is given by*

$$\sum_{j=0}^{e-2} \left\lceil \frac{\#Col_j(S_{\Delta_e})}{k} \right\rceil \cdot q_\ell$$

Parallel Optimal Strategy Problem

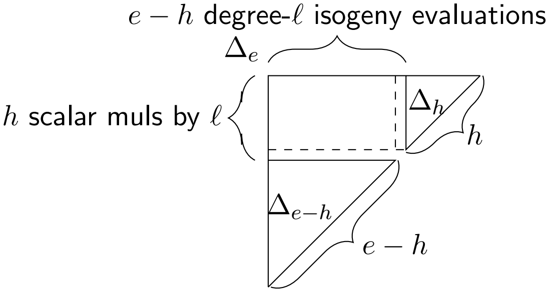
Proposition (Parallel Optimal Strategy Problem)

Let q_ℓ *be the timing cost associated to the computation of a degree-*ℓ *isogeny. Let us define a set of horizontal edges for a fixed index* $j \in \{0, 1, \ldots, e-2\}$ *by* $Col_j(S_{\Delta_e}) = \{[(i,j),(i,j+1)] \in S_{\Delta_e} \mid i \in [0, e-j-2]\}$*. Now the cost of evaluating* S_{Δ_e} *using* k *cores is given as*

$$C^k(S_{\Delta_e}) = \sum_{j=0}^{e-1} \left\lceil \frac{\#Col_j(S_{\Delta_e})}{k} \right\rceil \cdot q_\ell + \#V(S_{\Delta_e}) \cdot p_\ell,$$

where $V(S_{\Delta_e})$ *is the set of all vertical edges in* S_{Δ_e}

Optimal strategies for parallel environments



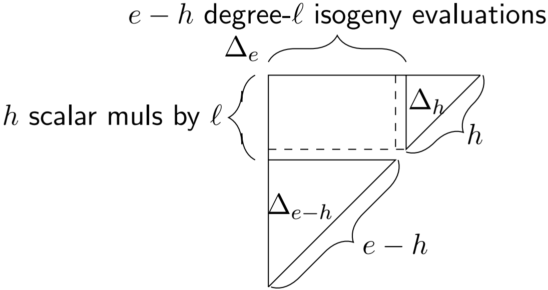
Lemma (Single core cost [deFeo-Jao-Plût JMC'14])

Given a triangle Δ_e and its decomposition into Δ_h and Δ_{e-h}, the sequential cost of traversing S_{Δ_e} is given as,

$$C^1(S^h_{\Delta_e}) = C^1(S_{\Delta_h}) + C^1(S_{\Delta_{e-h}}) + (e-h)\cdot q_\ell + h\cdot p_\ell.$$

We say that S_{Δ_e} is an optimal strategy if $C^1(S^h_{\Delta_e})$ is minimal among all $S^h_{\Delta_e}$ for $h \in [1, e-1]$.

Optimal strategies for parallel environments



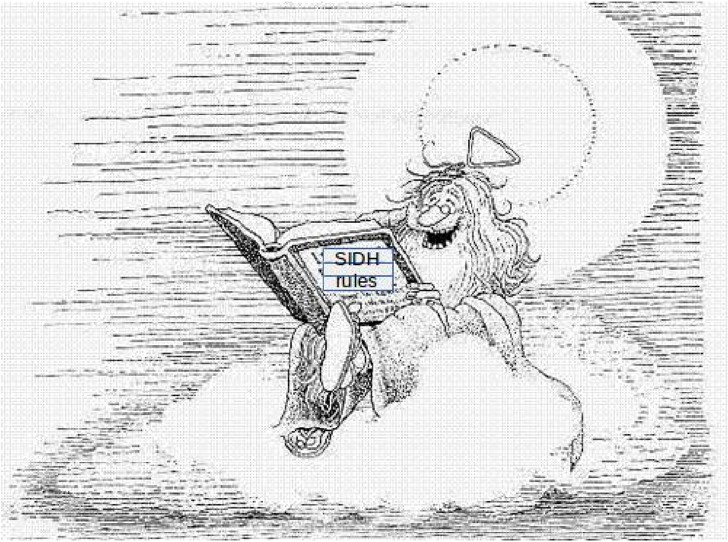
Lemma (Multi-core cost)

Given a triangle Δ_e and its decomposition into Δ_h and Δ_{e-h}, the cost of traversing S_{Δ_e} using k cores is given as,

$$C^k(S^h_{\Delta_e}) = C^k(S_{\Delta_{e-h}}) + C^k(St_h) + \frac{(e-h)\cdot q_\ell}{k} + h\cdot p_\ell,$$

We say that $S^h_{\Delta_e}$ is an optimal parallel strategy if $C^k(S^h_{\Delta_e})$ is minimum among all $S^h_{\Delta_e}$ for $h \in [1, e-1]$.

Breaking the rules



Breaking the rules

Proposition

- *Computing* $[2^i]R_{\mathsf{A}}$ *costs* $(e_{\mathsf{A}} - i)$ ***xDBL*** .
- *Computing* $R_{\mathsf{A}} = P_{\mathsf{A}} + [m_{\mathsf{A}}]Q_{\mathsf{A}}$ *costs* e_{A} ***xDBL*** .
- *Hence, computing* $[2^i]R_{\mathsf{A}}$ *costs less than computing* R_{A}.

Breaking the rules

Proposition

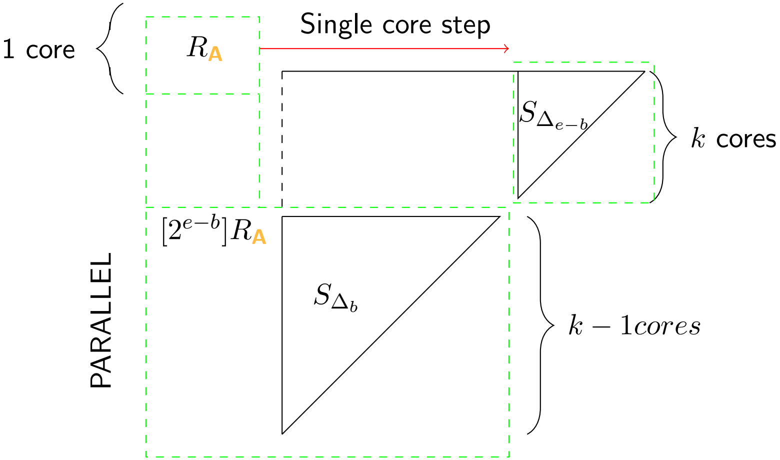
- *Computing* $[2^i]R_A$ *costs* $(e_A - i)$ *xDBL* .
- *Computing* $R_A = P_A + [m_A]Q_A$ *costs* e_A *xDBL* .
- *Hence, computing* $[2^i]R_A$ *costs less than computing* R_A.

Proof.

- As P_A and Q_A are public parameters, then we can pre-compute all points $[2^i]P_A$ and $[2^i]Q_A$ for $i = 1$ to $e_A - 1$.
- Note that $[2^i]R_A = [2^i]P_A + [m_A]([2^i]Q_A)$.
- As $[2^i]Q_A$ has order $2^{e_A - i}$, then we can replace m_A by $\bar{m_A}$ where $\bar{m_A} = m_A \mod 2^{e_A - i}$. This has a size of $2(e_A - i)$ bits
- Computing $[2^i]R_A = [2^i](P_A + [m_A \mod 2^{e_A - i}]Q_A)$ costs $(e_A - i)$ xDBL

$\square$

New parallel strategy: computing Alice's secret point and isogenies concurrently



Experiments and efficiency

Experimental setting

- **SIDH protocol instantiation**
  - NIST quantum security level 5 level using the prime,

$$p_{751} = 4^{186} \cdot 3^{239} - 1$$

  - $e_4 = 185$, $p_4 = 11,902$, $q_4 = 8,108$, $r_4 = 3,492$,
    $\mathbb{F}_{p_{751}^2}$ inversion $=$ 310,512.
    Where r_4 is the cost of computing a degree-4 isogeny curve.
    All the costs above are given in clock cycles.

Experimental setting

- **Software tools and platform**
  - We benchmarked our software on an Intel Core i9-9980XE processor supporting the Skylake micro-architecture.
  - The Intel Hyper-Threading and Intel Turbo Boost technologies were disabled.
  - The OpenMP v4.5 API was used for parallelization.
  - The source code was compiled using Clang v9.0 with the `-O3` optimization flag and using the options `-mbmi2 -madx -fwrapv -fomit-frame-pointer -fopenmp`.

Experimental results

# of cores k	Estimated Cost (including R in parallel)	Experimental timings / Strategy type: Parallel (including R in parallel)	Single core
1	19.60 (19.60)	19.00 (19.00)	19.00
2	16.44 (14.73)	16.57 (15.17)	17.06
3	15.04 (13.21)	15.95 (13.82)	16.35
4	14.19 (12.25)	14.64 (13.51)	16.11
6	13.30 (11.36)	14.17 (13.30)	15.20
8	12.71 (10.80)	13.50 (12.97)	15.20

Table 1: A comparison of estimated versus experimental costs of computing the key agreement phase of the SIDH protocol for its p_{751} instantiation. All estimates and experimental results are given in 10^6 clock cycles. The last column reports the timing costs of the SIDH protocol using the [sequential] optimal strategies of [deFeo-Jao-Plût JMC'14].

Experimental results

Cores	Isogeny Evaluations Serial	Isogeny Evaluations Parallel	Muls	Cost	AF
1	784	784	636	20.32	1
2	849	540	508	15.35	1.32
3	1,006	436	445	13.76	1.48
4	1,125	370	413	12.77	1.59
8	1,723	303	331	11.26	1.80
22	3,083	233	281	10.15	2.00
60	9,099	245	187	9.26	2.20
122	9,456	184	184	8.73	2.33
184	9,456	184	184	8.73	2.33
185	9,456	184	184	8.73	2.33

Table 1: Estimate costs of Alice's SIDH key agreement phase instantiated for p_{751}. All estimates are given in 10^6 clock cycles. The Acceleration Factor (AF) column is the quotient of the Single core cost and the parallel cost using k cores.

Achieving the limits of SIDH parallelization

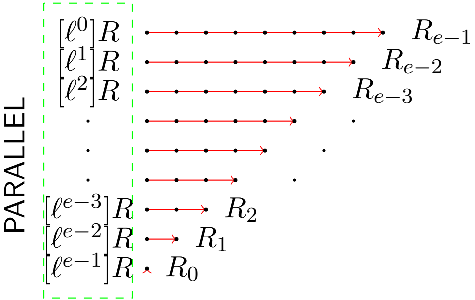


Figure 4: If the hardware resources are plentiful enough, all multiples of R can be computed in parallel. Also, if there are e available cores, all isogeny evaluations can be computed in parallel.

Extended SIDH

Parameters

$$\mathsf{p} := 2^{e_{\mathsf{A}}} \cdot 3^{e_{\mathsf{B}}} \cdot 5^{e_{\mathsf{C}}} f - 1$$

Such that $3^{e_{\mathsf{B}}} 5^{e_{\mathsf{C}}} \approx 2^{e_{\mathsf{A}}}$

and $3^{e_{\mathsf{B}}} \approx 5^{e_{\mathsf{C}}}$

Parameters

Choose P_{B} and Q_{B}
such that $\langle P_{\mathsf{B}}, Q_{\mathsf{B}} \rangle = E[3^{e_{\mathsf{B}}}]$

Choose P_{A} and Q_{A}
such that $\langle P_{\mathsf{A}}, Q_{\mathsf{A}} \rangle = E[2^{e_{\mathsf{A}}}]$

Choose P_{C} and Q_{C}
such that $\langle P_{\mathsf{C}}, Q_{\mathsf{C}} \rangle = E[5^{e_{\mathsf{C}}}]$

$$\mathsf{p} := 2^{e_{\mathsf{A}}} \cdot 3^{e_{\mathsf{B}}} \cdot 5^{e_{\mathsf{C}}} f - 1$$

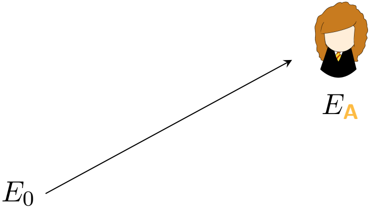
Such that $3^{e_{\mathsf{B}}} 5^{e_{\mathsf{C}}} \approx 2^{e_{\mathsf{A}}}$
and $3^{e_{\mathsf{B}}} \approx 5^{e_{\mathsf{C}}}$

Define **S** $:= P_{\mathsf{B}} + P_{\mathsf{C}}$ and **T** $:= Q_{\mathsf{B}} + Q_{\mathsf{C}}$
to be the public parameters of **B** and **C**

eSIDH

$K_{\mathsf{A}} := P_{\mathsf{A}} + [m_{\mathsf{A}}]Q_{\mathsf{A}}$

Get ϕ_{A} and E_{A}

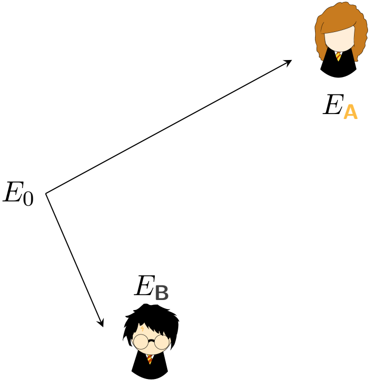


eSIDH

Parallel

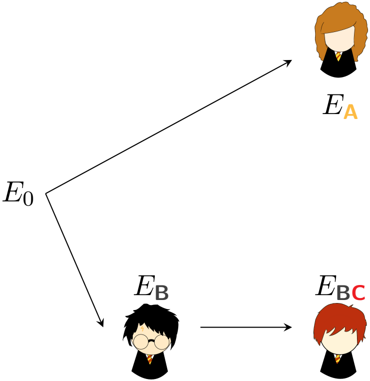
$K_B := P_B + [m_B]Q_B$ $K_{\mathbf{C}} := P_{\mathbf{C}} + [m_{\mathbf{C}}]Q_{\mathbf{C}}$

Get ϕ_B and E_B. "*Send*" $\phi_B(K_{\mathbf{C}})$ to **C**.



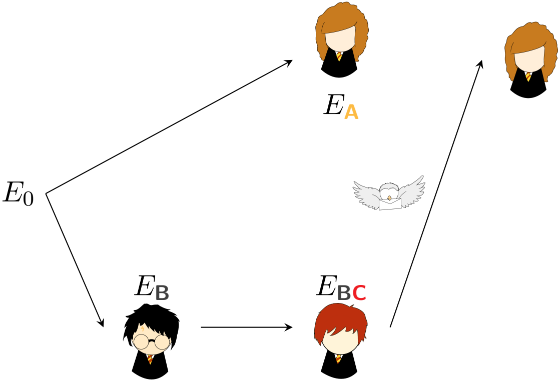
eSIDH

Use $\phi_B(K_{\mathbf{C}})$ to get $E_{B\mathbf{C}}$ and $\phi_{B\mathbf{C}}$

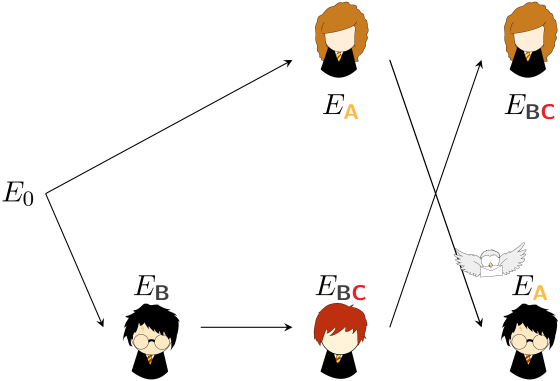


eSIDH

$$(E_{B\mathbf{C}}, \phi_{B\mathbf{C}}(P_{\mathbf{A}}), \phi_{B\mathbf{C}}(Q_{\mathbf{A}}))$$

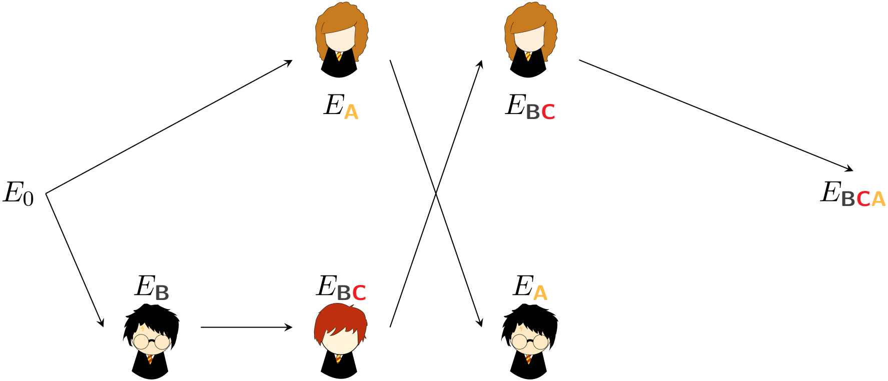


$$(E_{\mathsf{A}}, \phi_{\mathsf{A}}(\mathsf{S}), \phi_{\mathsf{A}}(\mathsf{T}))$$



$$K'_{A} := \phi_{BC}(P_{A}) + [m_{A}]\phi_{BC}(Q_{A})$$

Get E_{BCA}

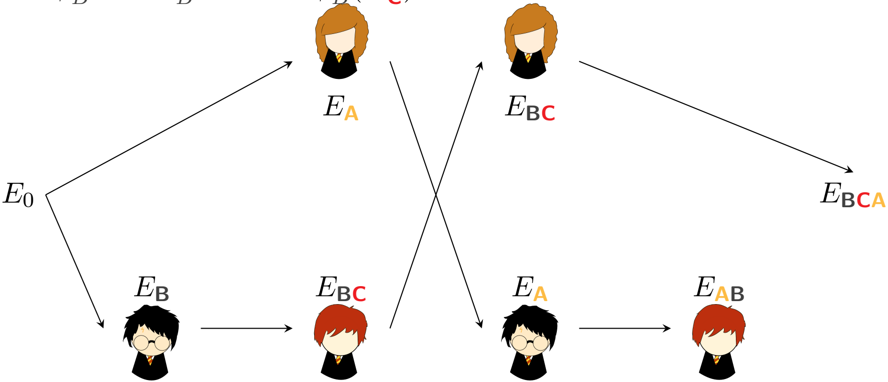


eSIDH

Parallel

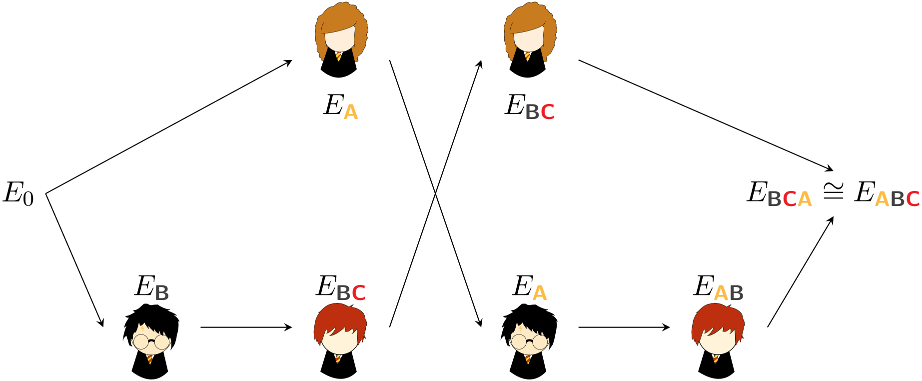
$K'_B := [5^{e_C}](\phi_A(S) + [m_B]\phi_A(T))$ $K'_C := [3^{e_B}](\phi_A(S) + [m_C]\phi_A(T))$

Get ϕ'_B and E'_B. "*Send*" $\phi'_B(K'_C)$ to **C**.

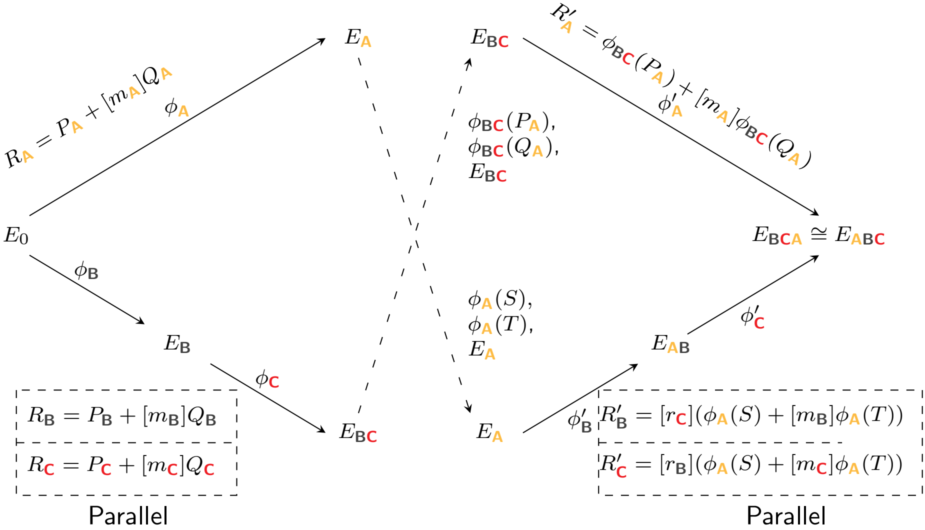


eSIDH

Use $\phi'_B(K'_{\mathbf{C}})$ to get $E_{\mathbf{A}B\mathbf{C}}$



eSIDH parallel instantiation at a glance



Computing a degree-$3^{e_B}5^{e_C}$ isogeny $\phi_{BC} = \phi_C \circ \phi_B$

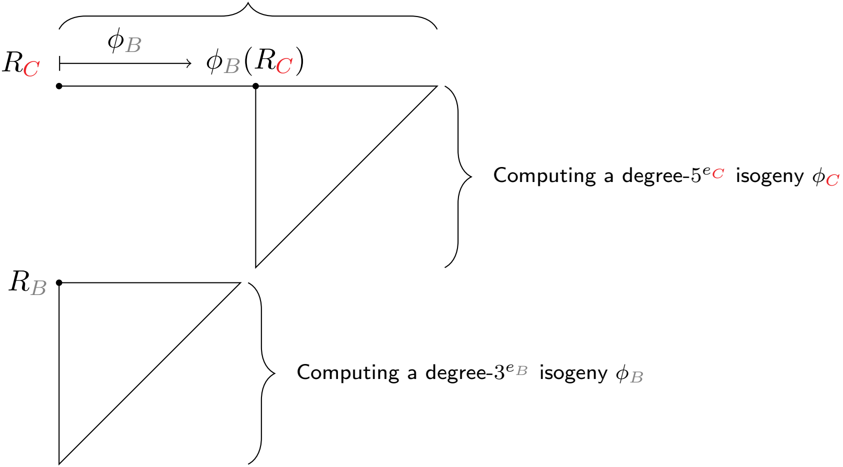


Figure 5: Overview of an strategy to compute a degree-$3^{e_B}5^{e_C}$ isogeny, which exploits parallelism by defining two secret points R_B and R_C for Bob. The kernel of ϕ_B is the subgroup $\langle R_B \rangle$, and the kernel of ϕ_C is the subgroup $\langle \phi_B(R_C) \rangle$.

Protocol	Single Core processor required # of xDBL	Two-Core processor required # of xDBL
SIDH [Jao-deFeo-Plût]	$\frac{16\lambda}{4}$	$\frac{16\lambda}{4}$
CRT-based*	$\frac{15\lambda}{4}$	$\frac{13\lambda}{4}$
eSIDH Parallel	$\frac{16\lambda}{4}$	$\frac{11\lambda}{4}$

Table 2: Let $\lambda = \lceil \log_2(p) \rceil$ be the bit-length of the eSIDH prime p. This table shows the approximate number of xDBL operations processed by the SIDH protocol of Jao-deFeo-Plût compared against the parallel eSIDH variant discussed here. *The description of the eSIDH CRT-based version was omitted in this talk.

Experiments and efficiency

Experiments and efficiency

eSIDH primes proposed here	N	γ	SIKE primes	N	γ
$p_{443} = 2^{222}3^{73}5^{45} - 1$	7	3	$p_{434} = 2^{216}3^{137} - 1$	7	3
$p_{508} = 2^{258}3^{74}5^{57} - 1$	8	4	$p_{503} = 2^{250}3^{159} - 1$	8	3
$p_{628} = 2^{320}3^{94}5^{67} - 1$	10	5	$p_{610} = 2^{305}3^{192} - 1$	10	4
$p_{765} = 2^{391}3^{119}5^{81} - 1$	12	6	$p_{751} = 2^{372}3^{239} - 1$	12	5

Table 3: Selection of eSIDH primes matching the four security levels offered by the SIKE primes. $N = \lceil \lceil \log_2(p) \rceil / 64 \rceil$, and γ is the largest integer for that N such that $p \equiv -1 \mod 2^{\gamma \cdot 64}$ holds.

Experiments and efficiency

Phase	p_{751} Number of cores 1	p_{751} Number of cores 2	p_{751} Number of cores 3	p_{765} Number of cores 1	p_{765} Number of cores 2	p_{765} Number of cores 3
Key generation	26.74	23.69	22.26	24.26	17.80	15.81
Encapsulation	43.19	38.57	35.59	40.38	36.12	33.98
Decapsulation	46.51	40.82	38.48	45.03	37.26	35.10
Total	116.44	103.08	96.33	109.67	91.18	84.89

Table 3: SIKE Performance comparison of the SIKE prime p_{751} against the eSIDH prime p_{765}. All timings are reported in 10^6 clock cycles measured on an Intel Skylake proccessor at 4.0 GHz.

Combining all the tricks

Experiments and Estimates Work in progress

Phase	p_{751}	p_{765}		
	Number of cores	Number of cores		
	1	1	2	3
Alice Key Generation	23.59	22.27	15.93	14.80
Bob Key Generation	26.74	24.34	17.76	15.79
Alice Key Agreement	19.37	18.21	14.30	13.07
Bob Key Agreement	22.76	23.24	17.16	15.94
Total	92.46	88.05	65.15	59.06

Table 4: SIKE protocol experimental timing costs for its p_{751} instantiation. All timings are given in 10^6 clock cycles measured on an Intel Skylake proccessor at 4.0 GHz. An acceleration factor of 1.57 was measured for a SIKE three-core implementation.

Experiments and Estimates Work in progress

k	Estimate (including R)	Parallel Strategy (including R)
1	19.08 (19.08)	18.22 (18.22)
2	15.96 (14.32)	16.14 (14.30)
3	15.04 (12.83)	14.91 (13.07)
4	14.60 (11.90)	
6	12.90 (10.98)	
8	12.32 (10.48)	

Table 4: A comparison of estimated versus experimental costs of the key agreement phase of the SIKE protocol for its p_{751} instantiation. All estimates and experimental results are given in 10^6 clock cycles measured on an Intel Skylake proccessor at 4.0 GHz. An acceleration factor of 1.82 is expected for a SIKE eight-core implementation.

Gracias-Thanks
for your attention

Questions?

Credits: Many of the drawings in this presentation were designed by Daniel Cervantes-Vázquez. Others were borrowed from Quino, Escher and Banksy. The pictures of Botero's and Miro's paintings were taken by the speaker.