Elliptic curves in classical and post-quantum cryptography

Ph.D. candidate: **Jesús Javier Chi Domínguez** [1]
Advisor: **Francisco Rodríguez-Henríquez** [1]

[1]Computer Science Department, Cinvestav - IPN, Mexico City, Mexico

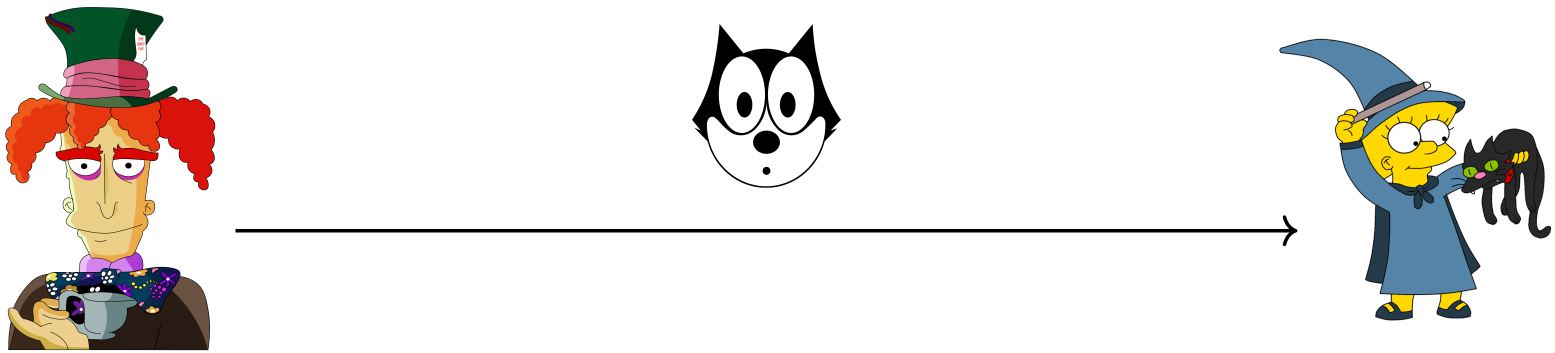
December 9, 2019

Agenda

1. Introduction
2. Extending the GLS endomorphism to speedup the GHS Weil descent using Magma
3. On the Cost of Computing Isogenies Between Supersingular Elliptic Curves
   - VW golden collision search
   - Comments about quantum algorithms
4. Stronger and Faster Side-Channel Protections for CSIDH
   - Constant-time CSIDH algorithm
   - Removing dummy operations
   - Experimental results
5. Concluding remarks
   - Publications
   - Forthcoming research

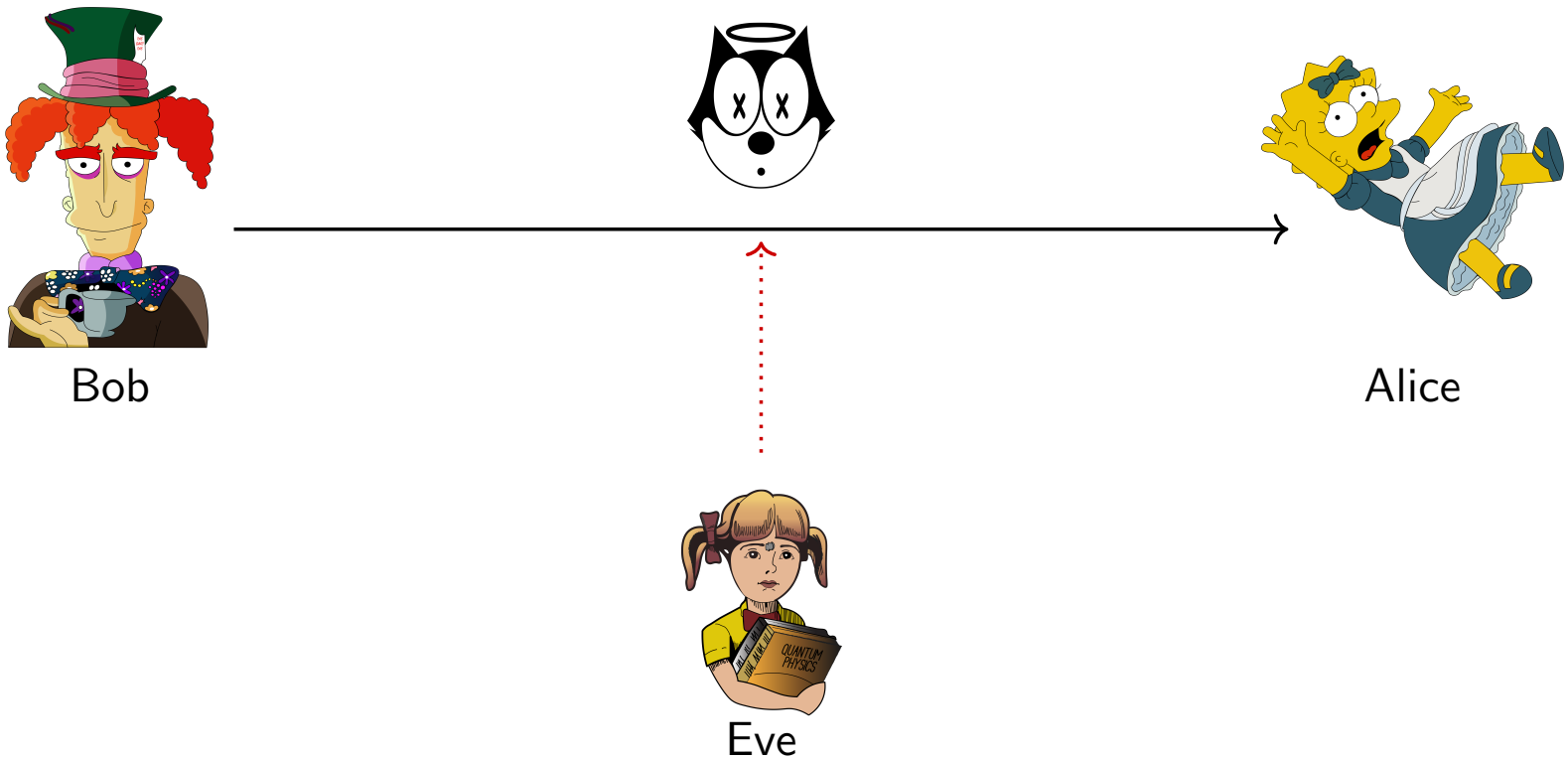
Outline

Basic communication scheme



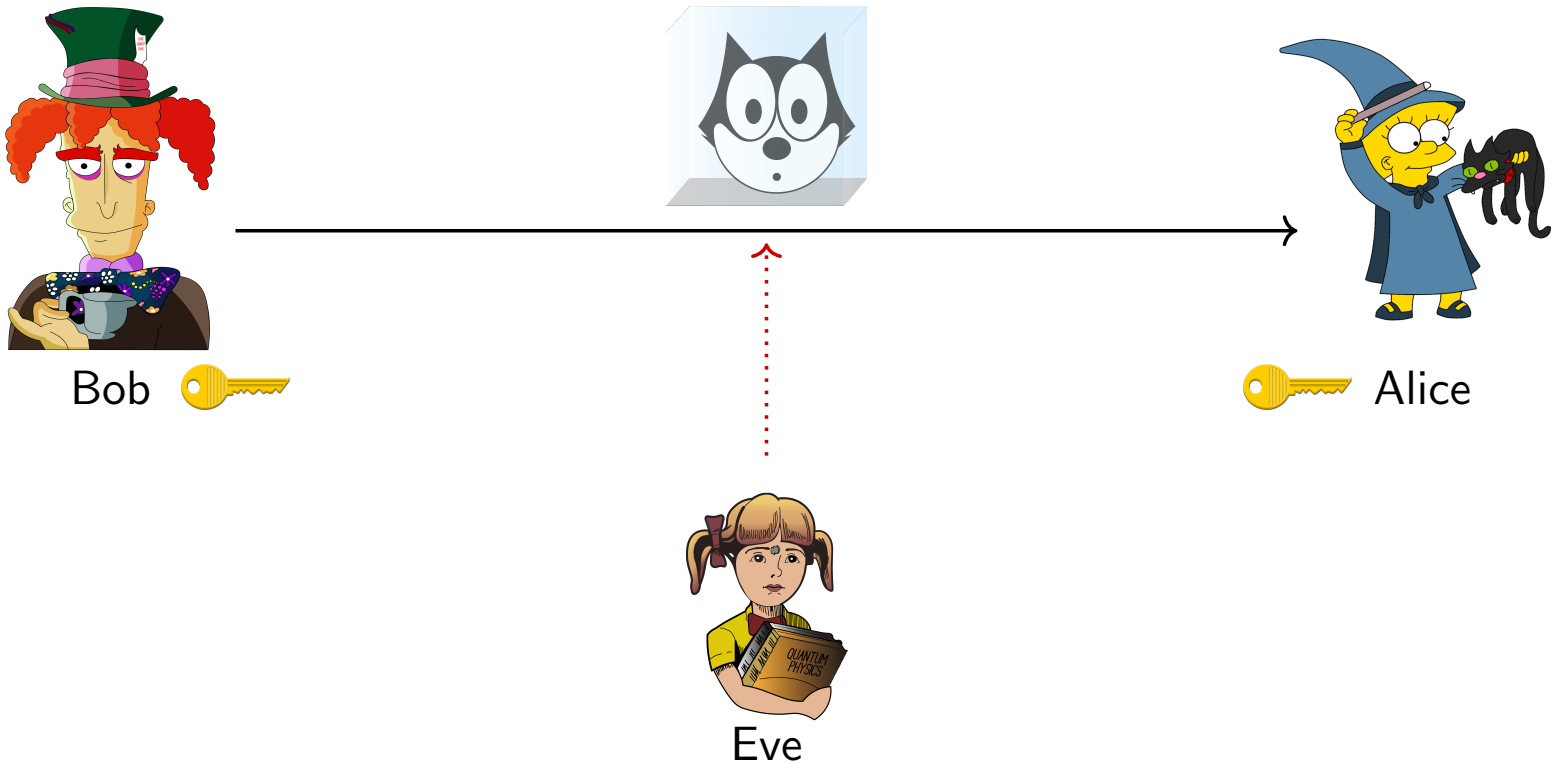
Basic communication scheme

There are not secure channels.



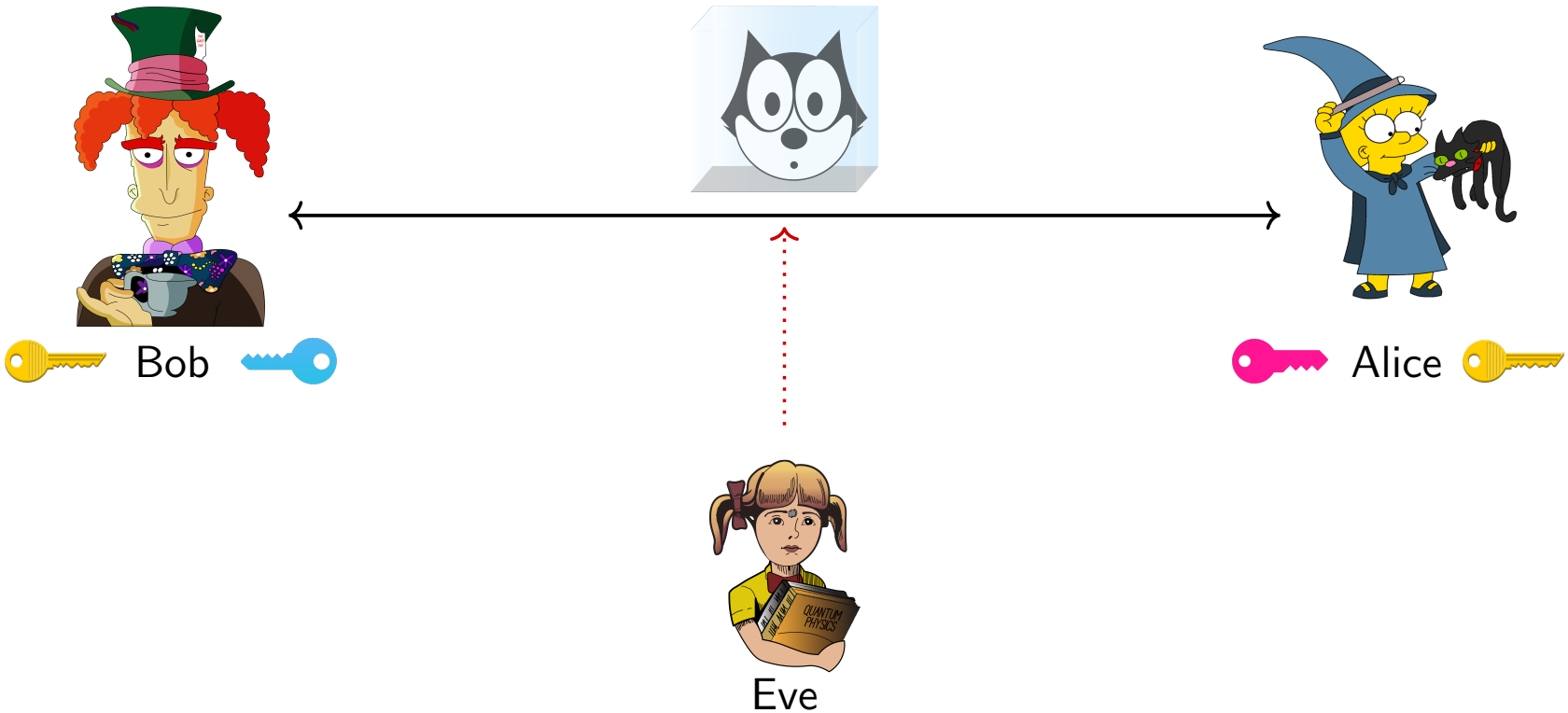
Basic communication scheme

There are not secure channels. Thereby the need of using Cryptography (i.e., to encrypt messages).



Current public-key cryptography

Security based on hard computational problems like *Integer Factorization (IFP)* and *Discrete Logarithm (DLP)*



Why should we use elliptic-curve-based cryptography? It allows small keys compared with other primitives.

Post-quantum cryptography

The Shor's quantum algorithm allows to solve the IFP and DLP with a polynomial running-time complexity [1], and the global giants such as Intel, Google, IBM, Rigetti, and Microsoft are investing heavily in the development of quantum computers.

Figure 1: *... middle of the ghetto, bunch of monsters, this time of night with quantum physics books? ... those books are WAY too advanced for her. If you ask me, I'd say she's up to something...* - James Edwards from MIB film (1997).

Post-quantum cryptography

There are 26 candidates for being considered by the U.S. government's National Institute of Standards and Technology (NIST) for inclusion in a forthcoming standard for quantum-safe cryptography. Those candidates fall in one of the following schemes:

- Code-based,
- Lattice-based ,
- Multivariate-quadratic based,
- Hash-based, and
- Isogeny-based.

Why must we use isogeny-based cryptography? It allows small keys compared with the another primitives.

Contributions of this thesis

1) Classical world (ECDH):

   Improvements on solving DLP on $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}} : y^2+xy = x^3+ax^2+b$

2) Post-quantum world:
   2.a) Security analysis of isogeny-based cryptography (SIDH):

   $$\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$$

   2.b) Efficiently constant-time implementations of isogeny-based cryptography (CSIDH):

   $$\mathcal{E}/\mathbb{F}_{p} : y^2 = x^3 + Ax^2 + x$$

Outline

ECDH overview

ECDH framework [2, 3]:

- n and ℓ two positive integers such that $\gcd(n, \ell)$,
- $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}: y^2 + xy = x^3 + ax^2 + b$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = c \cdot r \approx r \approx 2^{n\times\ell}$, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^\ell}$, and
- an order-r point P.

ECDH overview

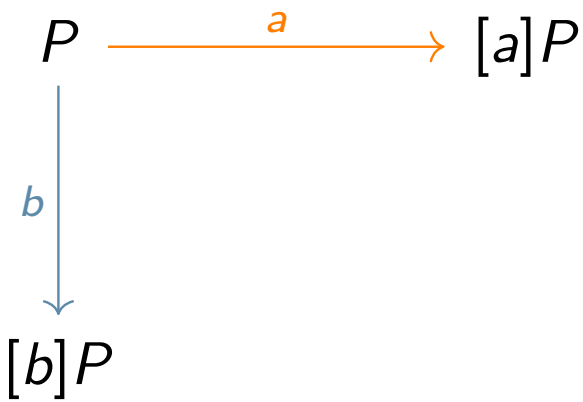
ECDH framework [2, 3]:

- n and ℓ two positive integers such that $\gcd(n, \ell)$,
- $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}: y^2 + xy = x^3 + ax^2 + b$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = c \cdot r \approx r \approx 2^{n\times\ell}$, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^\ell}$, and
- an order-r point P.

General description ECDH:

$P_A \leftarrow [a]P$

$P_B \leftarrow [b]P$



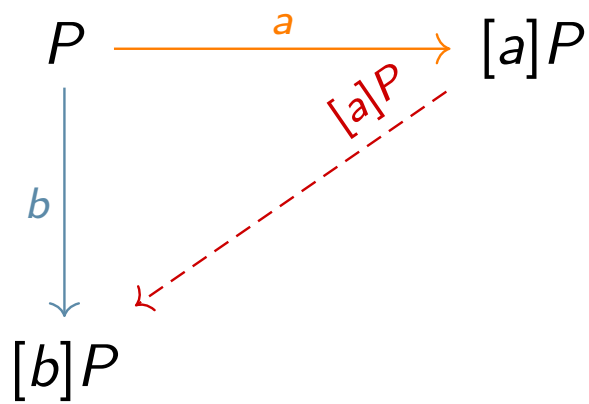
ECDH overview

ECDH framework [2, 3]:

- n and ℓ two positive integers such that $\gcd(n, \ell)$,
- $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}: y^2 + xy = x^3 + ax^2 + b$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = c \cdot r \approx r \approx 2^{n\times\ell}$, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^\ell}$, and
- an order-r point P.

General description ECDH:

$$P_A \leftarrow [a]P$$
$$P_B \leftarrow [b]P$$



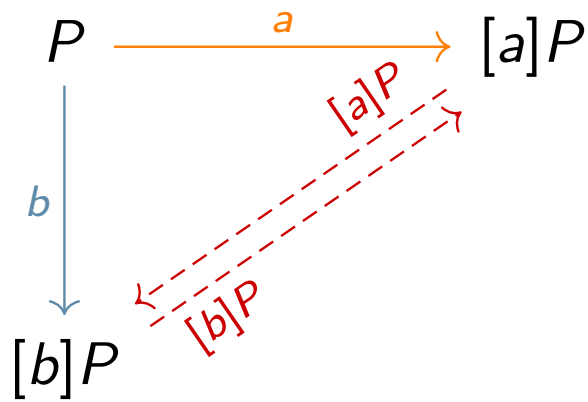
ECDH overview

ECDH framework [2, 3]:

- n and ℓ two positive integers such that $\gcd(n, \ell)$,
- $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}: y^2 + xy = x^3 + ax^2 + b$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = c \cdot r \approx r \approx 2^{n\times\ell}$, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^\ell}$, and
- an order-r point P.

General description ECDH:

$$P_A \leftarrow [a]P$$
$$P_B \leftarrow [b]P$$



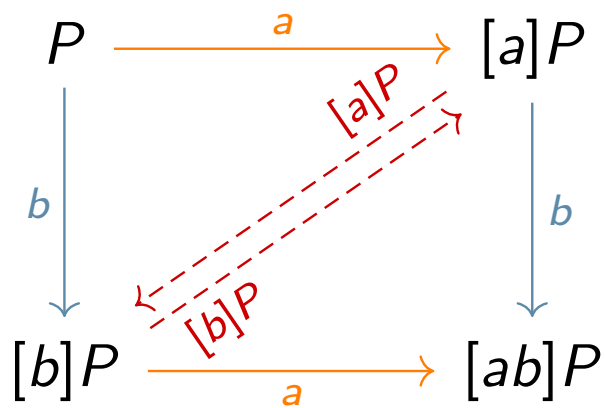
ECDH overview

ECDH framework [2, 3]:

- n and ℓ two positive integers such that $\gcd(n, \ell)$,
- $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}: y^2 + xy = x^3 + ax^2 + b$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = c \cdot r \approx r \approx 2^{n\times\ell}$, $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^\ell}$, and
- an order-r point P.

General description ECDH:

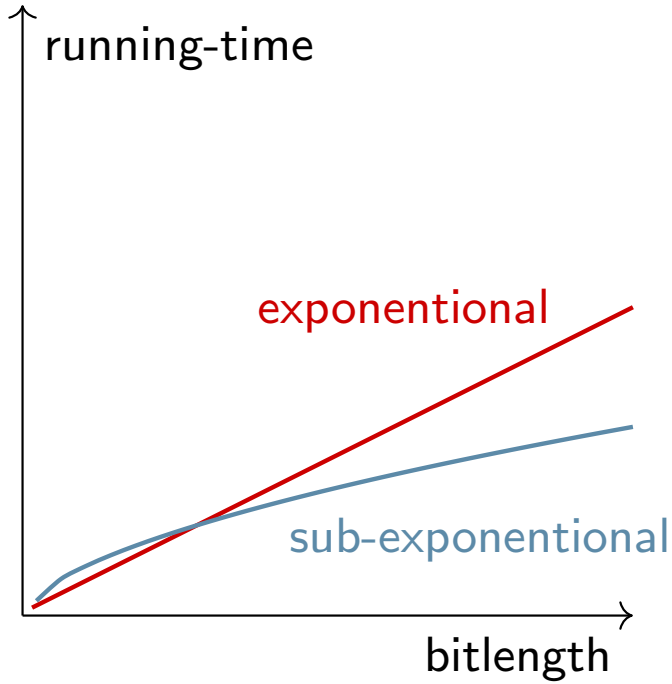
$$P_{A,B} \leftarrow [a]P_B$$
$$P_{B,A} \leftarrow [b]P_A$$



The shared secret key is $[ab]P$, and the security is given by the hardness of computing a (or b) given the data colored in red ink (DLP problem).

GHS Weil descent technique

The Guadry-Hess-Smart (GHS) Weil descent technique allows to map points of an elliptic curve into divisors of the Jacobian of a higher genus hyperelliptic curve.



This is interesting, because there are sub-exponentional index-calculus based algorithms for solving the DLP on higher genus hyperelliptic curves. The two most costly steps are the smooth divisors search and computation of the kernel element.

Index-calculus based algorithm

Smooth divisor search

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda\mathsf{D}$.

Index-calculus based algorithm

Smooth divisor search

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda\mathsf{D}$.

$$m_{1,1}\mathsf{D}_1 + m_{1,2}\mathsf{D}_2 + \cdots + m_{1,t}\mathsf{D} = \alpha_1\mathsf{D} + \beta_1\mathsf{D}'$$

Index-calculus based algorithm

Smooth divisor search

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda\mathsf{D}$.

$$m_{1,1}\mathsf{D}_1 + m_{1,2}\mathsf{D}_2 + \cdots + m_{1,t}\mathsf{D} = \alpha_1\mathsf{D} + \beta_1\mathsf{D}'$$
$$m_{2,1}\mathsf{D}_1 + m_{2,2}\mathsf{D}_2 + \cdots + m_{2,t}\mathsf{D} = \alpha_2\mathsf{D} + \beta_2\mathsf{D}'$$

Index-calculus based algorithm

Smooth divisor search

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda \mathsf{D}$.

$$
\begin{aligned}
m_{1,1}\mathsf{D}_1 + m_{1,2}\mathsf{D}_2 + \cdots + m_{1,t}\mathsf{D} &= \alpha_1 \mathsf{D} + \beta_1 \mathsf{D}' \\
m_{2,1}\mathsf{D}_1 + m_{2,2}\mathsf{D}_2 + \cdots + m_{2,t}\mathsf{D} &= \alpha_2 \mathsf{D} + \beta_2 \mathsf{D}' \\
&\vdots \\
m_{t,1}\mathsf{D}_1 + m_{t,2}\mathsf{D}_2 + \cdots + m_{t,t}\mathsf{D} &= \alpha_2 \mathsf{D} + \beta_2 \mathsf{D}'
\end{aligned}
$$

Index-calculus based algorithm

Smooth divisor search

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda \mathsf{D}$.

$$\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,t} \\ m_{2,1} & m_{2,2} & \cdots & m_{1,t} \\ \vdots & & \ddots & \vdots \\ m_{t,1} & m_{t,2} & \cdots & m_{t,t} \end{bmatrix} \begin{bmatrix} \mathsf{D}_1 \\ \mathsf{D}_2 \\ \vdots \\ \mathsf{D}_t \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_t \end{bmatrix} \mathsf{D} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_t \end{bmatrix} \mathsf{D}'$$

Index-calculus based algorithm

Computation of the kernel element

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda\mathsf{D}$.

$$\begin{bmatrix}\gamma_1 & \gamma_2 & \cdots & \gamma_t\end{bmatrix} \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,t} \\ m_{2,1} & m_{2,2} & \cdots & m_{1,t} \\ \vdots & & \ddots & \vdots \\ m_{t,1} & m_{t,2} & \cdots & m_{t,t} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Index-calculus based algorithm

Computation of the kernel element

Let $\mathcal{F}$ be the set of irreducible s-smooth divisors (pair of polynomials with first entry being an irreducible polynomial of degree at most s). Let's write $t := \#\mathcal{F}$. The task is to find λ such that $\mathsf{D}' = \lambda \mathsf{D}$.

$$\begin{bmatrix}\gamma_1 & \gamma_2 & \cdots & \gamma_t\end{bmatrix} \begin{bmatrix}\beta_1 \\ \beta_2 \\ \vdots \\ \beta_t\end{bmatrix} \mathsf{D}' = -\begin{bmatrix}\gamma_1 & \gamma_2 & \cdots & \gamma_t\end{bmatrix} \begin{bmatrix}\alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_t\end{bmatrix} \mathsf{D}$$

And thus λ can be easily computed in terms of $\overrightarrow{\gamma}$, $\overrightarrow{\alpha}$, and $\overrightarrow{\beta}$.

GLS endomorphism extension

The GLS endomorphism extends an efficient endomorphism on the jacobian, and it yields a factor n and n^2 speedup on the smooth divisors search and kernel element's computation steps, respectively.

$$\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,\frac{t}{n}} \\ m_{2,1} & m_{2,2} & \cdots & m_{1,\frac{t}{n}} \\ \vdots & & \ddots & \vdots \\ m_{\frac{t}{n},1} & m_{\frac{t}{n},2} & \cdots & m_{\frac{t}{n},\frac{t}{n}} \end{bmatrix} \begin{bmatrix} \mathsf{D}_1 \\ \mathsf{D}_2 \\ \vdots \\ \mathsf{D}_{\frac{t}{n}} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\frac{t}{n}} \end{bmatrix} \mathsf{D} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{\frac{t}{n}} \end{bmatrix} \mathsf{D}'$$

GLS endomorphism extension

	This work	Velichka *et al.* work [4]		
		JMS EG Estimate	Opt. EG Estimate	Sieving method
4-smooth divisors search (CPU days)	**1034.572**	8492.67	6338.01	1720.818
Linear algebra step (CPU days)	**0.024**	2.470	2.800	14.244
Total (CPU days)	**1034.597**	8495.650	6340.810	1735.063
Speedup:		**8.212**	**6.129**	**1.677**

Table 1: Index-Calculus based algorithm: DLP computation on a hyperelliptic genus-32 curve $\mathcal{H}/\mathbb{F}_{2^5}$. The 2nd, 3rd, and 4th column show the timing estimations of using the Enge-Gaudry algorithm with i) the strategy and optimal parameters from [5], ii) an optimized version that incorporates large prime variations, and the sieve-based version of Vollmer's algorithm, respectively.

Outline

SIDH overview

SIDH framework [7, 6]:

- $p = \ell_A^{e_A} \ell_B^{e_B} d - 1$ is a prime number,
- $\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = (p+1)^2$ and $j(\mathcal{E}) = 1728\frac{4A^3}{4A^3+27B^2}$.
- $\mathcal{E}[\ell_A^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

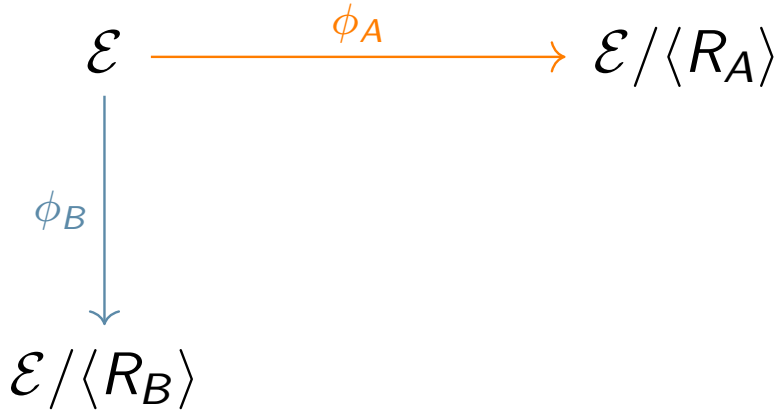
SIDH overview

SIDH framework [7, 6]:

- $p = \ell_A^{e_A} \ell_B^{e_B} d - 1$ is a prime number,
- $\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = (p+1)^2$ and $j(\mathcal{E}) = 1728\frac{4A^3}{4A^3+27B^2}$.
- $\mathcal{E}[\ell_A^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description SIDH:

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$



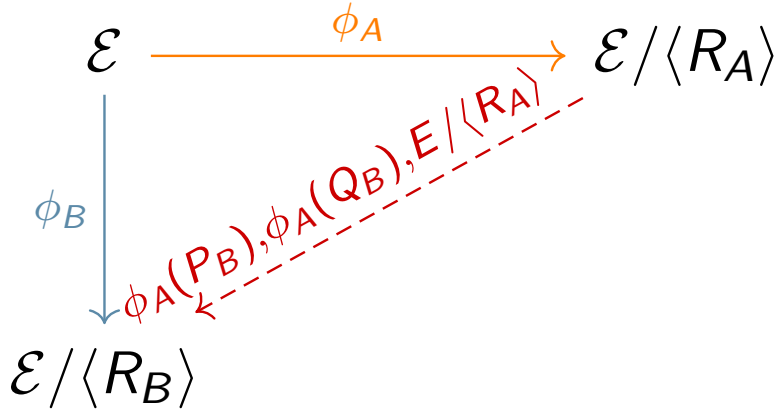
SIDH overview

SIDH framework [7, 6]:

- $p = \ell_A^{e_A} \ell_B^{e_B} d - 1$ is a prime number,
- $\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = (p+1)^2$ and $j(\mathcal{E}) = 1728\frac{4A^3}{4A^3+27B^2}$.
- $\mathcal{E}[\ell_A^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description SIDH:

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$



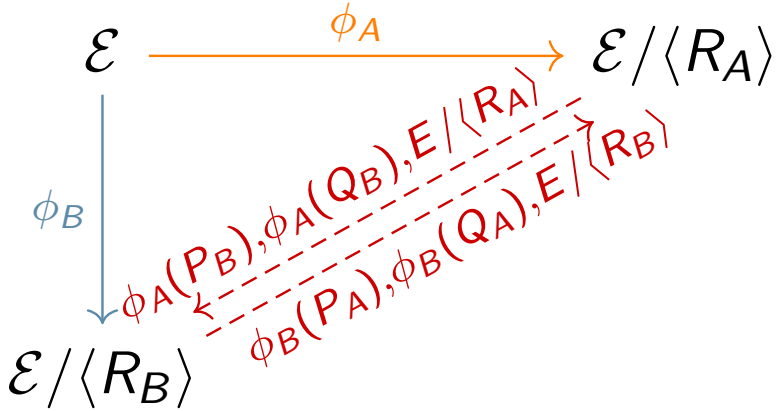
SIDH overview

SIDH framework [7, 6]:

- $p = \ell_A^{e_A} \ell_B^{e_B} d - 1$ is a prime number,
- $\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = (p+1)^2$ and $j(\mathcal{E}) = 1728\frac{4A^3}{4A^3+27B^2}$.
- $\mathcal{E}[\ell_A^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description SIDH:

$$R_A \leftarrow [n_A]P_A + [m_A]Q_A$$
$$R_B \leftarrow [n_B]P_B + [m_B]Q_B$$



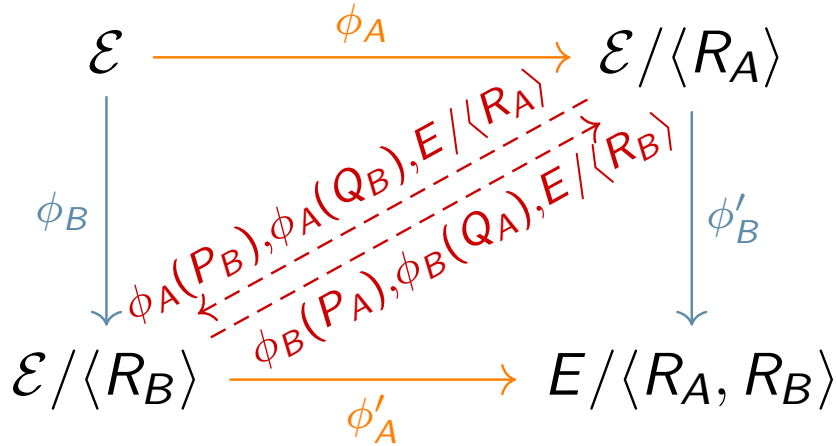
SIDH overview

SIDH framework [7, 6]:

- $p = \ell_A^{e_A} \ell_B^{e_B} d - 1$ is a prime number,
- $\mathcal{E}/\mathbb{F}_{p^2} : y^2 = x^3 + Ax + B$ with $\#\mathcal{E}(\mathbb{F}_{p^2}) = (p+1)^2$ and $j(\mathcal{E}) = 1728\frac{4A^3}{4A^3+27B^2}$.
- $\mathcal{E}[\ell_A^{e_A}](\mathbb{F}_{p^2}) = \langle P_A, Q_A \rangle$ and $E[\ell_B^{e_B}](\mathbb{F}_{p^2}) = \langle P_B, Q_B \rangle$.

General description SIDH:

$$\phi_B(R_A) \leftarrow [n_A]\phi_B(P_A) + [m_A]\phi_B(Q_A)$$
$$\phi_A(R_B) \leftarrow [n_B]\phi_A(P_B) + [m_B]\phi_A(Q_B)$$



The shared secret key is $j(E/\langle R_A, R_B \rangle)$, and the security is given by the hardness of computing ϕ_A (or ϕ_B) given the data colored in red ink (CSSI problem).

Solving CSSI

The isogeny ϕ_A is the composition of e_A degree-ℓ_A isogenies.

Figure 2: $\mathcal{E}/\langle R_A \rangle$ is a degree-$\left(2^6\right)$ isogenous curve to $\mathcal{E}$.

Solving CSSI

The isogeny ϕ_A is the composition of e_A degree-ℓ_A isogenies.

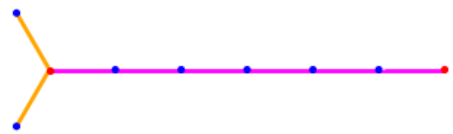


Figure 2: There are 3 degree-(2) isogenous curves to $\mathcal{E}$.

Solving CSSI

The isogeny ϕ_A is the composition of e_A degree-ℓ_A isogenies.

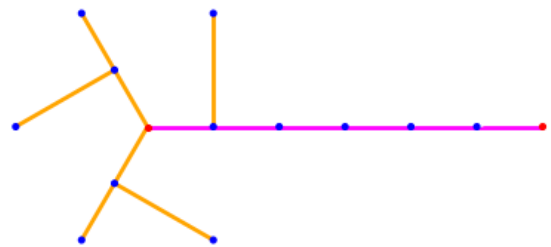


Figure 2: There are $3 \cdot 2$ degree-$\left(2^2\right)$ isogenous curves to $\mathcal{E}$.

Solving CSSI

The isogeny ϕ_A is the composition of e_A degree-ℓ_A isogenies.

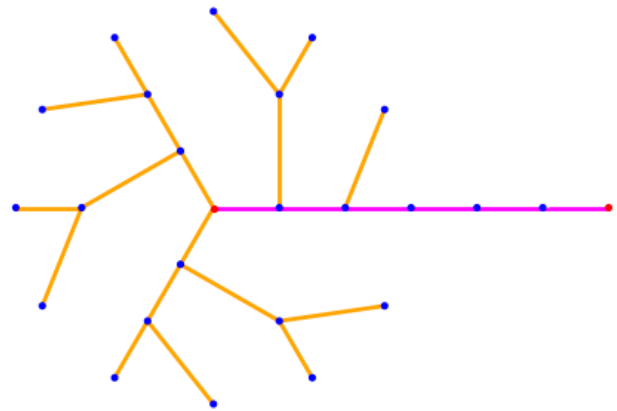


Figure 2: There are $3 \cdot 2^2$ degree-$\left(2^3\right)$ isogenous curves to $\mathcal{E}$.

Solving CSSI

The isogeny ϕ_A is the composition of e_A degree-ℓ_A isogenies.

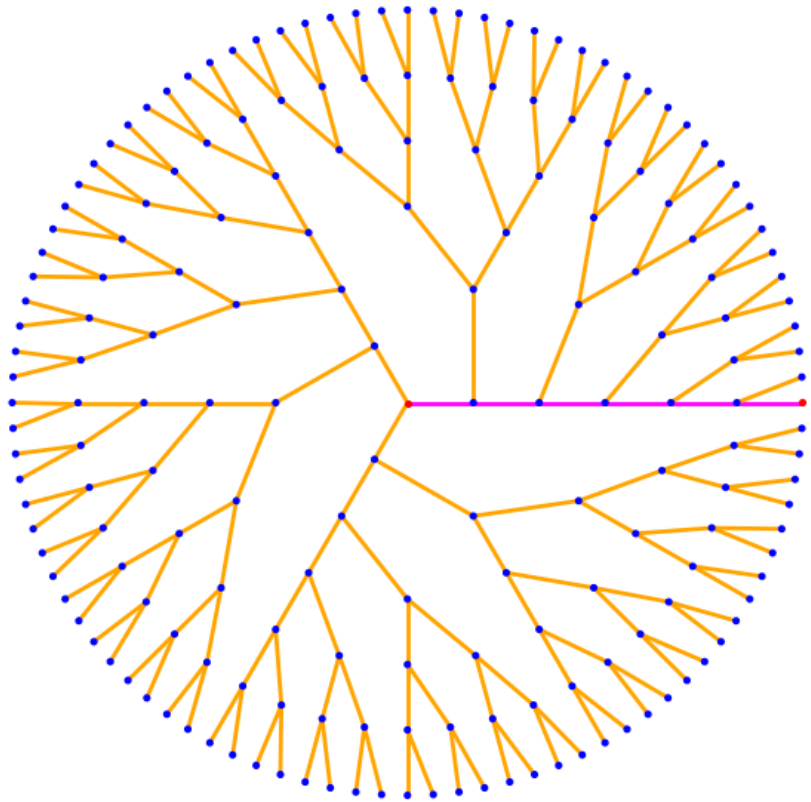


Figure 2: There are $3 \cdot 2^5$ degree-$\left(2^6\right)$ isogenous curves to $\mathcal{E}$.

Solving CSSI

Compute all the degree-$\ell_A^{e_A/2}$ isogenous curves to $\mathcal{E}$ and $\mathcal{E}/\langle R_A \rangle$.

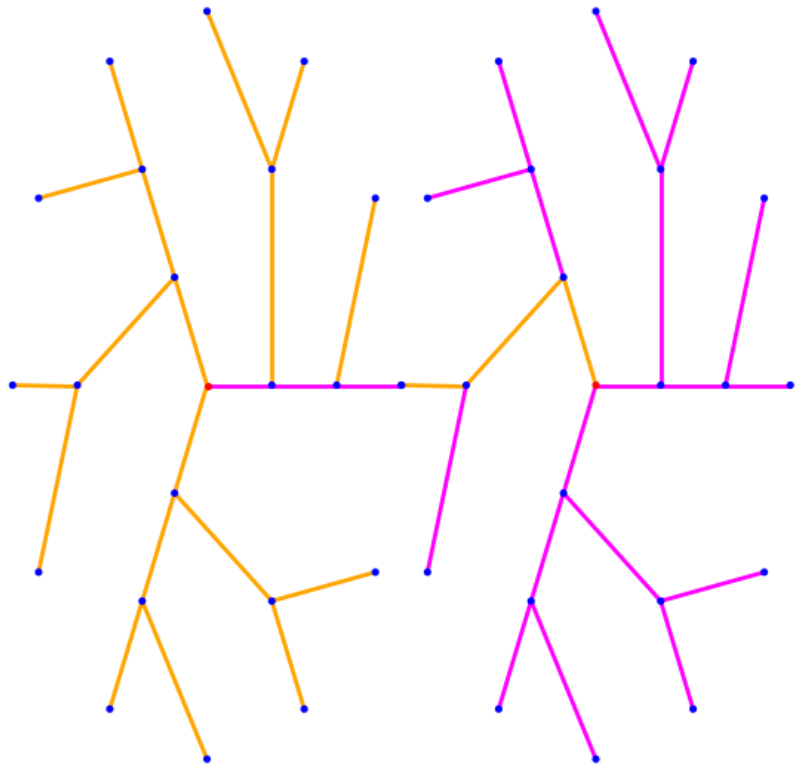


Figure 3: Degree-$\left(2^3\right)$ isogenous curves to $\mathcal{E}$ and $\mathcal{E}/\langle R_A \rangle$.

Solving CSSI (before this work)

The best known classical algorithm has a running-time of $p^{1/4}$.

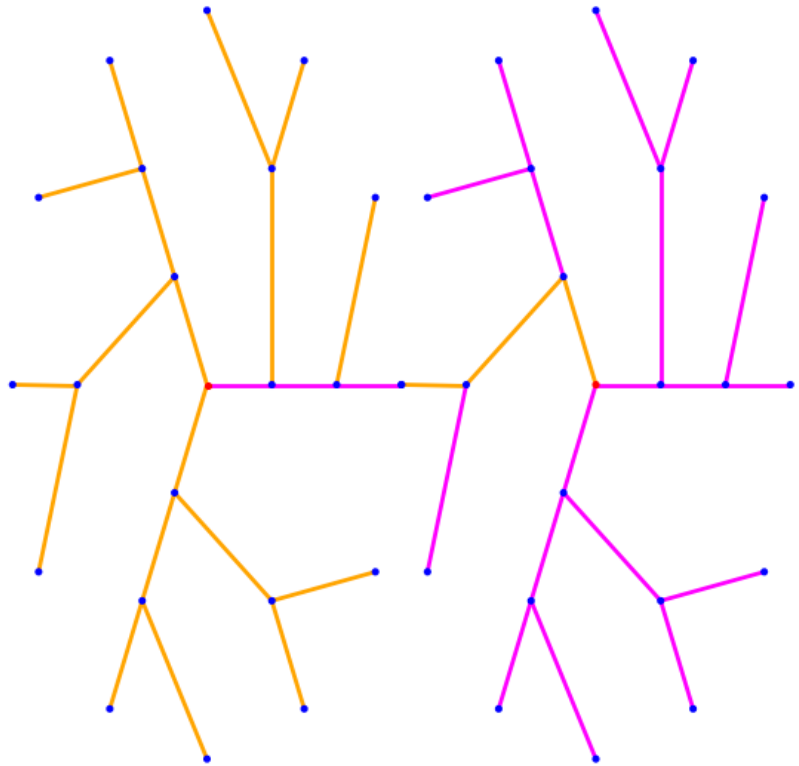


Figure 3: Degree-(2^3) isogenous curves to $\mathcal{E}$ and $\mathcal{E}/\langle R_A \rangle$.

Solving CSSI (before this work)

The best known quantum algorithm has a running-time of $p^{1/6}$.

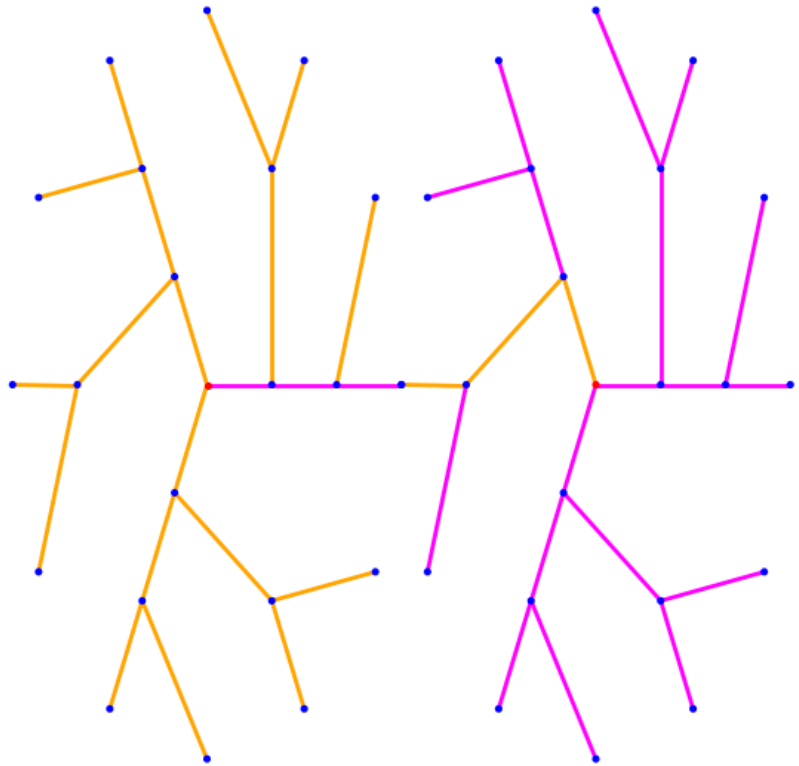


Figure 3: Degree-$\left(2^3\right)$ isogenous curves to $\mathcal{E}$ and $\mathcal{E}/\langle R_A \rangle$.

Outline

Collision search

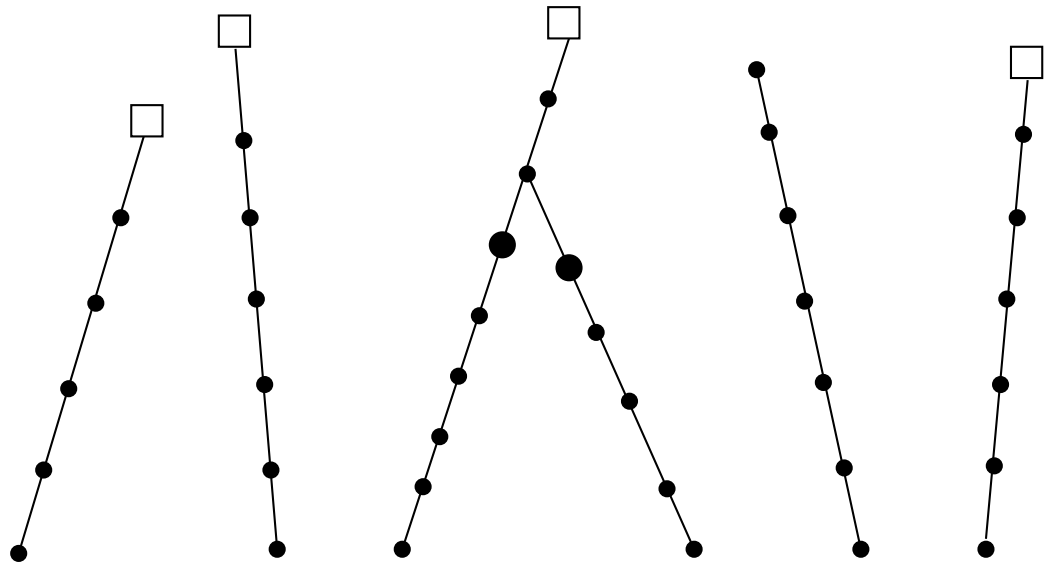
Firstly, The average-case time complexity of the Meet-in-the-middle attack is $1.5N$ and it has space complexity N, where $N \approx (\ell_A + 1)\ell^{e_A/2-1} \approx p^{1/4}$ (Infeasible for $N \geq 2^{80}$).
Consequently, using m processors and w cells of memory, the running time of MITM is approximately

$$(w/m + N/m)\frac{N}{w} \approx N^2/(w \cdot m) \approx p^{1/2}/(w \cdot m).$$

Really? $p^{1/2}$ is the square of $p^{1/4}$. But don't worry!, we can do it better than $p^{1/2}$.

VW golden collision search

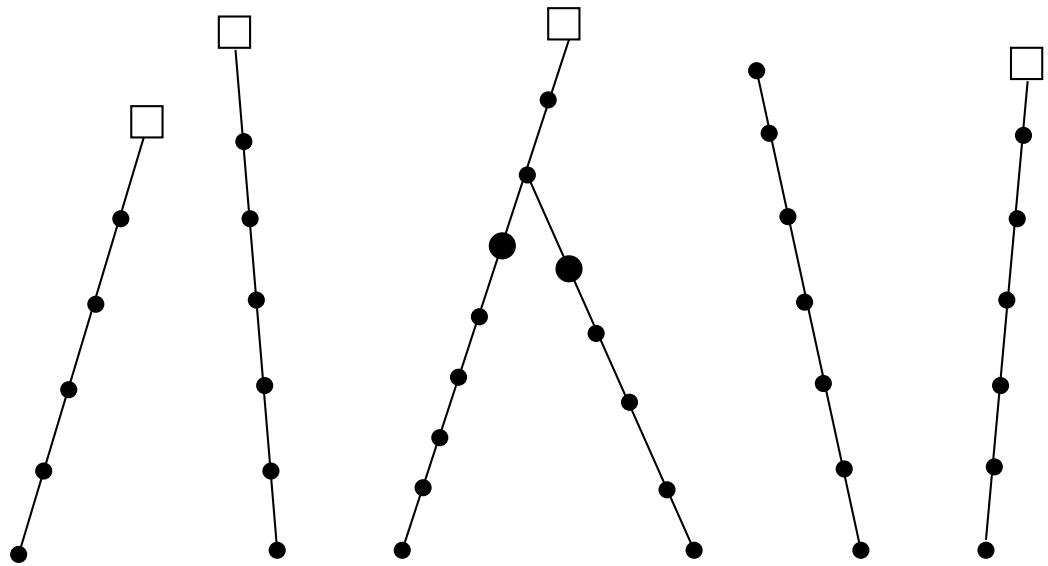
Let S be a finite set of size M. The goal is to find a collision for a random function $f : S \to S$.



Let's define an element x of S to be *distinguished* if it has some easily-testable distinguishing property, and let θ be the proportion of distinguished elements of S. However, a random function $f : S \to S$ is expected to have $(M-1)/2$ unordered collisions.

VW golden collision search

Let S be a finite set of size M. The goal is to find a collision for a random function $f : S \to S$.



Suppose we seek a particular one of these collisions, called the *golden collision*, which can be efficiently recognized. Thus, one continues generating distinguished points and collisions until the golden collision is encountered.

VW golden collision search

The golden collision might occur with very small probability compared to other collision. Consequently, it is necessary to change the version of f periodically.

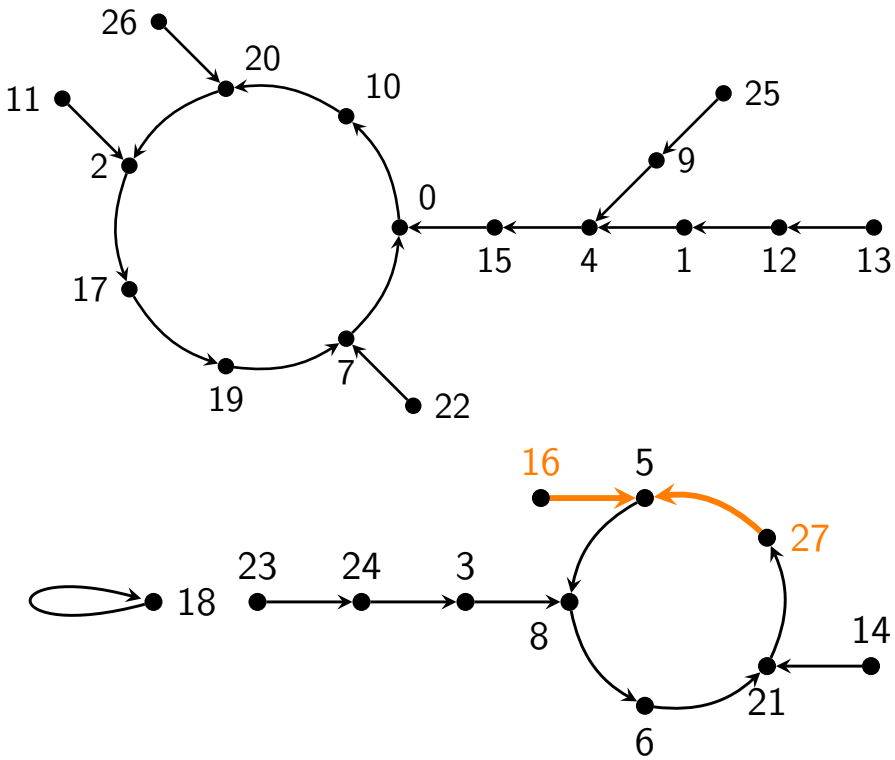


Figure 4: Functional graph of a random function $f : [\![0, 27]\!] \to [\![0, 27]\!]$. The desire golden collision is marked with Orange.

VW golden collision search

Let

- w be the number of elements we can store in memory,
- $\theta = 2.25\sqrt{w/M}$,
- $10w$ be the number of distinguished elements that each version of f produces,
- $2^{10} \leq w \leq M/2^{10}$.

Heuristically, van Oorschot and Wiener saw that each version of f generates approximately $1.3w$ collisions, of which approximately $1.1w$ are distinct [8, 9]. In addition, the expected running time to find the golden collisions when m processors are employed is

$$\frac{1}{m}\left(2.5\sqrt{M^3/w}\right). \tag{1}$$

VW golden collision search

Consequently, the expected running time for solving CSSI problem VW golden collision search is

$$\frac{1}{m}\left(2.5\sqrt{8N^3/w}\right) \approx 7.1p^{3/8}/(w^{1/2}m) \ll p^{1/2}/(w \cdot m)$$

e	p	w	2^8	2^{10}	2^{12}	2^{14}	2^{16}
50	$2^{50}3^{31}179-1$	c_1	1.37	1.36	1.37	1.41	1.49
		c_2	1.14	1.12	1.12	1.11	1.09
60	$2^{60}3^{37}31-1$	c_1	1.37	1.34	1.34	1.35	1.36
		c_2	1.15	1.13	1.13	1.12	1.12
70	$2^{70}3^{32}127-1$	c_1	1.33	1.34	1.34	1.34	1.34
		c_2	1.13	1.14	1.13	1.13	1.13
80	$2^{80}3^{25}71-1$	c_1	1.35	1.32	1.33	1.34	1.33
		c_2	1.14	1.12	1.13	1.13	1.13

Table 2: Observed number $c_1 w$ of collisions and number $c_2 w$ of distinct collisions per CSSI-based random function f_n.

Solving CSSI problem: 128-, 160-, 192-bit security

# processors m	space w	$p \approx 2^{448}$ calendar time	$p \approx 2^{448}$ total time	$p \approx 2^{512}$ calendar time	$p \approx 2^{512}$ total time	$p \approx 2^{536}$ calendar time	$p \approx 2^{536}$ total time	$p \approx 2^{614}$ calendar time	$p \approx 2^{614}$ total time
Meet-in-the-middle using Depth-first search									
48	64	106	154	138	186	150	198	188	236
48	80	90	138	122	170	134	182	172	220
64	80	74	138	106	170	118	182	156	220
van Oorschot and Wiener golden collision search									
48	64	88	136	112	160	121	169	149	197
48	80	80	128	104	152	113	161	141	189
64	80	64	128	88	152	97	161	125	189

Table 3: Time complexity estimates of CSSI attacks for $p \approx 2^{448}$, $p \approx 2^{512}$, $p \approx 2^{536}$ and $p \approx 2^{614}$. All numbers are expressed in their base-2 logarithms. The unit of time is a $2^{e/2}$-isogeny computation [1], and we are ignoring communication costs.

Conclusion: MITM is more costly than VW golden collision search.

[1] *Calendar time* is the elapsed time taken for a computation, whereas *total time* is the sum of the time expended by all m processors.

Outline

Comments about quantum attacks

Tani's algorithm

The fastest known quantum attack on CSSI is Tani's algorithm [13], which has an running time equal to $O(p^{1/6})$ and requires $O(p^{1/6})$ space.

Grover's algorithm

Clearly, CSSI can also be solved by an application of Grover's quantum search [10], which has a running time equal to $O(p^{1/4})$. However, using m quantum circuits only yields a speedup by a factor of $\sqrt{m}$ [11].

Tani vs Grover: the recent work of Jaques and Schanck argue that Tani's algorithm cost the same (up to poly-log factors) as Grover' s search in realistic models of quantum computation [14].

Comments about quantum attacks

Assuming that the maximum circuit depth is 2^k, the number of quantum circuits needed to perform Grover's search in one year for $p \approx 2^r$ is approximately $\left(\frac{2^{\frac{r}{4}}}{2^k}\right)^2$.

Maximum depth of a quantum circuit	$p \approx 2^{448}$	$p \approx 2^{512}$	$p \approx 2^{536}$	$p \approx 2^{614}$
	m	m	m	m
40	144	176	188	227
64	96	128	140	179

Table 4: Number of quantum circuits needed to perform Grover's search in one year for $p \approx 2^{448}$, $p \approx 2^{512}$, $p \approx 2^{536}$, and $p \approx 2^{614}$. All numbers are expressed in their base-2 logarithms. In particular, the NIST suggests that 2^{40} is the maximum depth of a quantum circuit that can be executed in one year using presently envisioned quantum computing architectures [12].

Conclusion: Gorver is more costly than VW golden collision search.

Outline

CSIDH overview

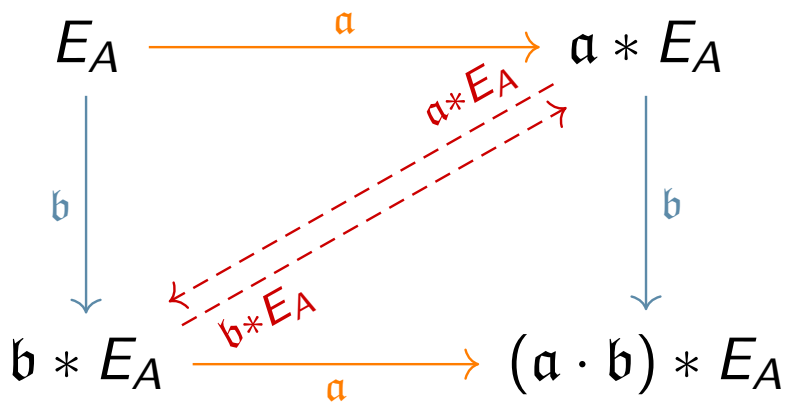
CSIDH framework [15]:

- Small odd primes numbers ℓ_i such that $p = 4\prod_{i=1}^{n}\ell_i - 1$ is prime number;
- Supersingular elliptic curves in Montgomery form $E_A/\mathbb{F}_p\colon y^2 = x^3 + Ax^2 + x$ with $\#E(\mathbb{F}_p) = p + 1$; and
- Positive integer m.

General description CSIDH:

The shared secret key is $(\mathfrak{a}\cdot\mathfrak{b})*E_A$.

The security is given by the hardness of computing $\mathfrak{a}$ (or $\mathfrak{b}$) given the data colored in red ink.



CSIDH overview

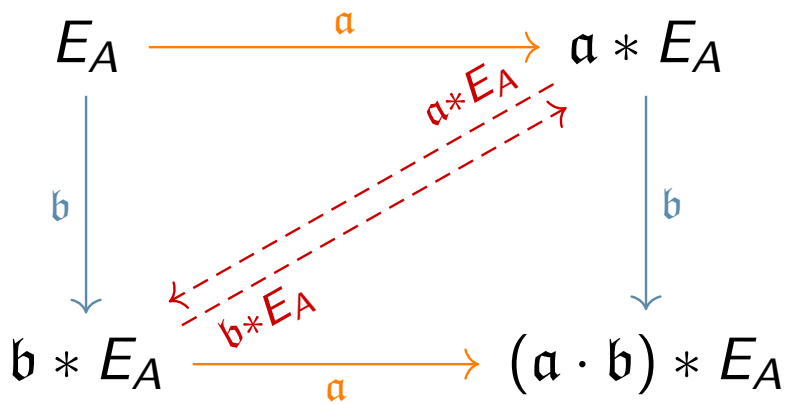
CSIDH framework [15]:

- Small odd primes numbers ℓ_i such that $p = 4\prod_{i=1}^{n} \ell_i - 1$ is prime number;
- Supersingular elliptic curves in Montgomery form $E_A/\mathbb{F}_p\colon y^2 = x^3 + Ax^2 + x$ with $\#E(\mathbb{F}_p) = p + 1$; and
- Positive integer m.

General description CSIDH:

The shared secret key is $(\mathfrak{a} \cdot \mathfrak{b}) * E_A$.

The security is given by the hardness of computing $\mathfrak{a}$ (or $\mathfrak{b}$) given the data colored in red ink.



Each ℓ_i is required e_i times for evaluating the *action* $\mathfrak{a} * E_A$ (similarly for $\mathfrak{b} * E_A$). Formally, this is written as $\mathfrak{a} = \mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}$.

CSIDH overview

The action $\mathfrak{a} * E_A$ defines a path on the isogeny graph over $\mathbb{F}_p$, and is determined by an integer vector $(e_1, \ldots, e_n) \in [\![-m, m]\!]^n$:

1) Nodes are supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form;
2) Edges are degree-ℓ_i isogenies.

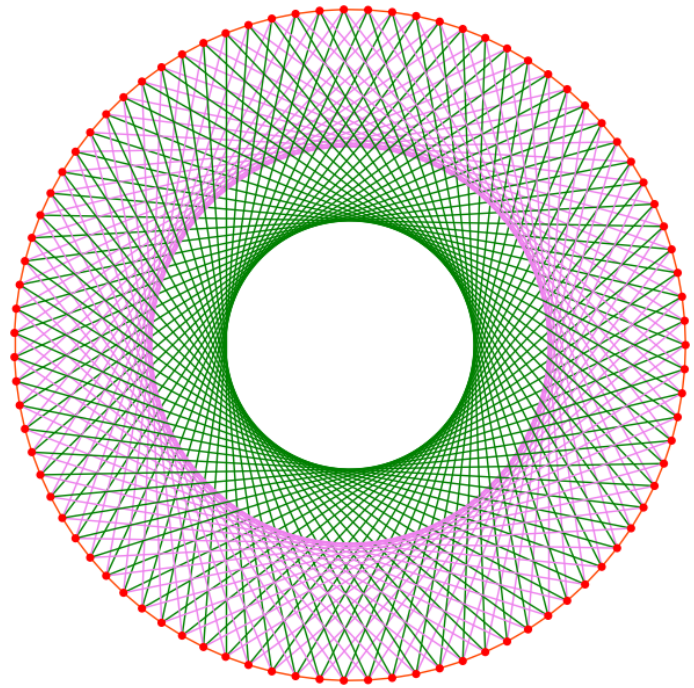


Figure 5: Isogeny graph over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Nodes are supersingular curves and edges marked with orange, green , and violet inks denote isogenies of degree 5, 13 and 61, respectively.

CSIDH overview

The action $\mathfrak{a} * E_A$ defines a path on the isogeny graph over $\mathbb{F}_p$, and is determined by an integer vector $(e_1, \ldots, e_n) \in [\![-m, m]\!]^n$:

1) Nodes are supersingular elliptic curves over $\mathbb{F}_p$ in Montgomery form;
2) Edges are degree-ℓ_i isogenies. Two types of edges: isogeny with kernel generated by
   2.a) $(x, y) \in E_A[\ell_i, \pi - 1]$, or
   2.b) $(x, iy) \in E_A[\ell_i, \pi + 1]$.

Here, $x, y \in \mathbb{F}_p$, $\pi : (X, Y) \mapsto (X^p, Y^p)$ is the Frobenius map, $i = \sqrt{-1}$ and thus $i^p = -i$.

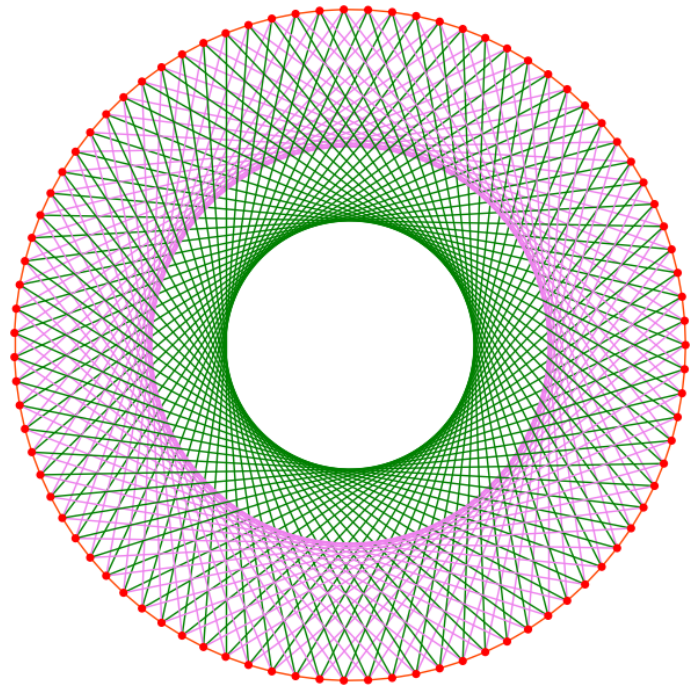


Figure 5: Isogeny graph over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Nodes are supersingular curves and edges marked with orange, green , and violet inks denote isogenies of degree 5, 13 and 61, respectively.

CSIDH overview

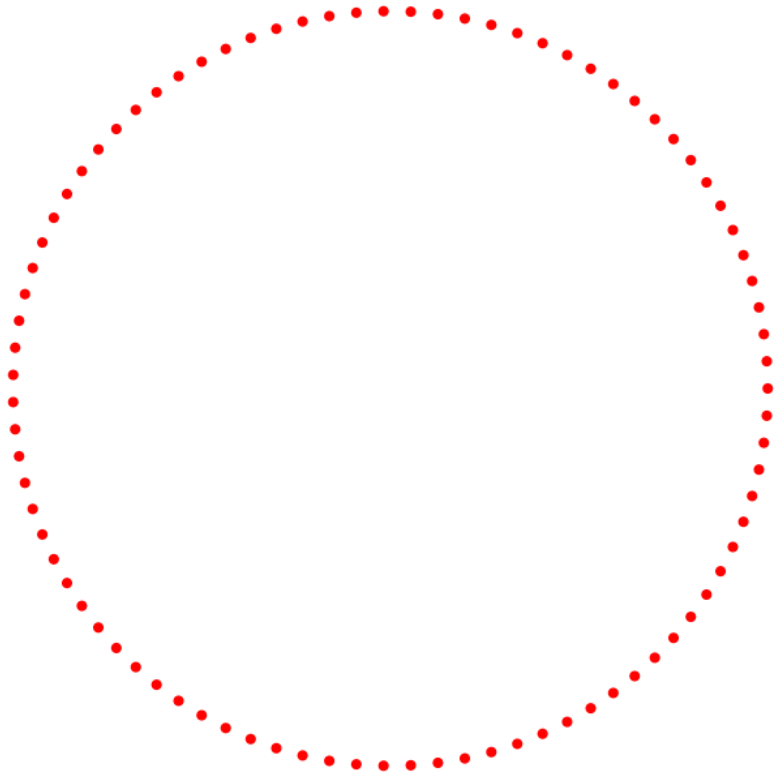


Figure 6: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

E_0

CSIDH overview

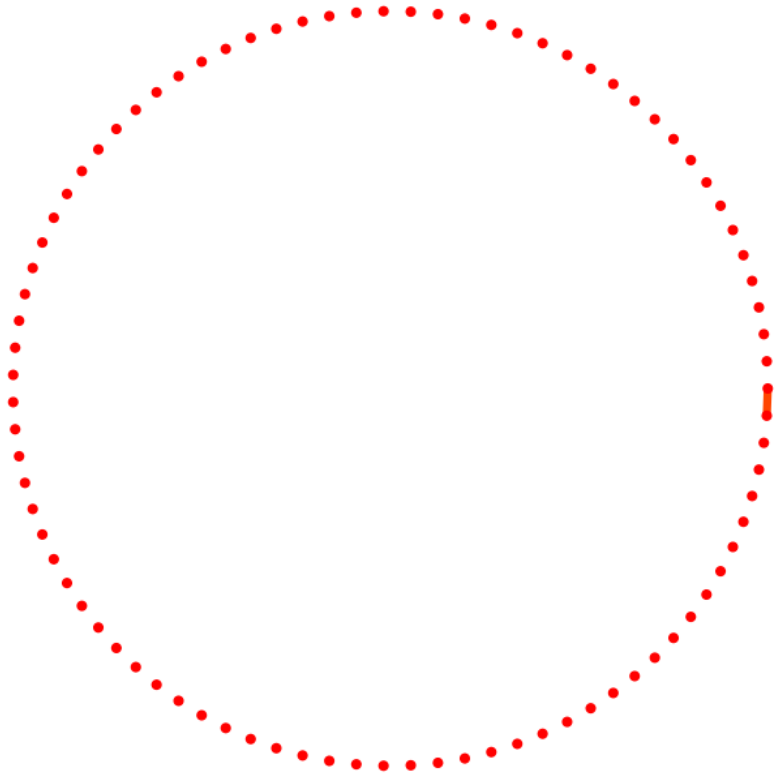


Figure 6: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \longrightarrow E_{0\text{x}3\text{A}7\text{D}}$$

CSIDH overview

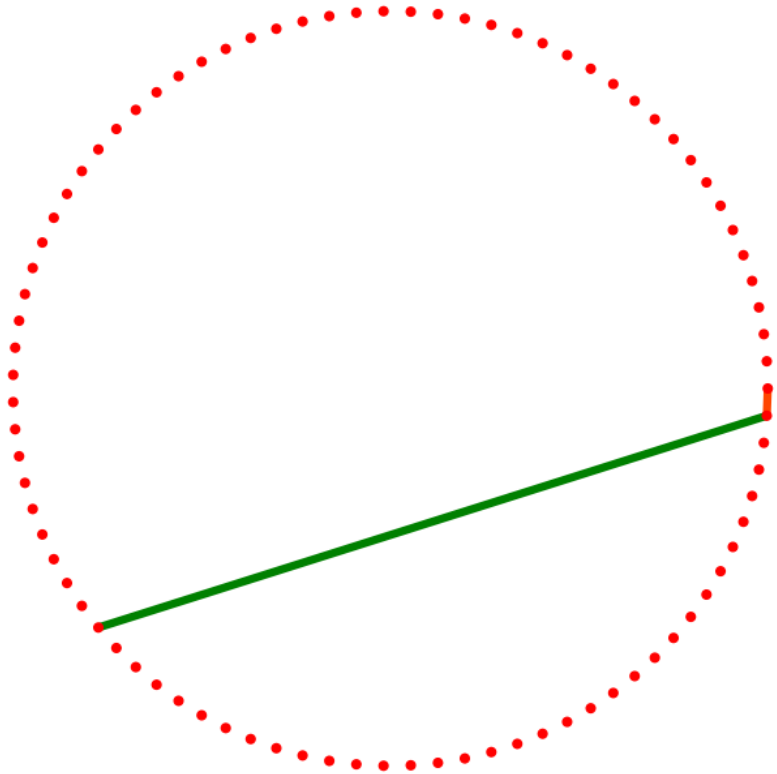


Figure 6: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \to E_{0\text{x}3\text{A}7\text{D}} \to E_{0\text{x}2\text{BF}7}$$

CSIDH overview

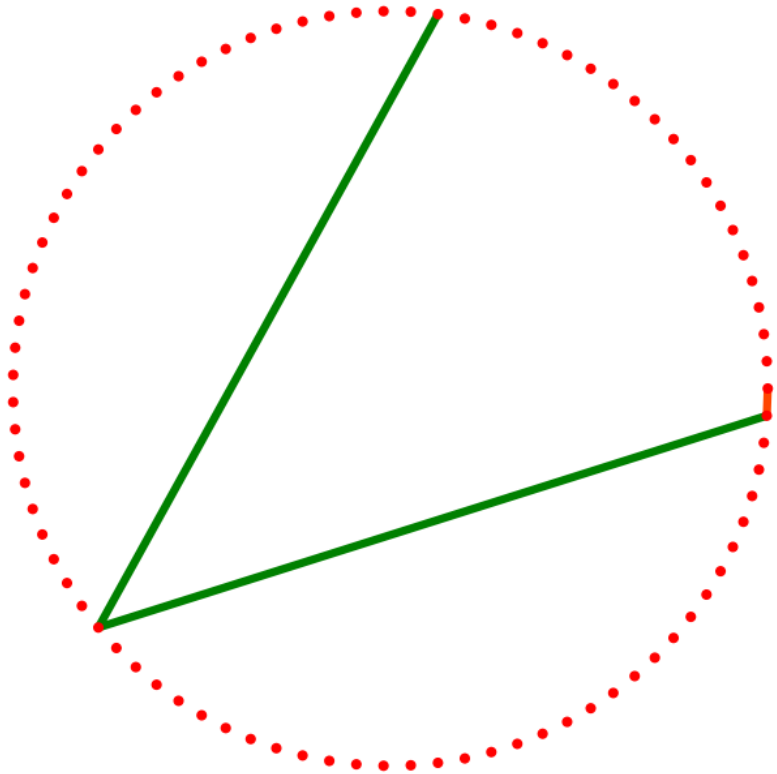


Figure 6: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(-1, 2, 1) \in [\![-2, 2]\!]^3$:

$$E_0 \to E_{0\text{x}3\text{A}7\text{D}} \to E_{0\text{x}2\text{BF}7} \to E_{0\text{x}1404}$$

Outline

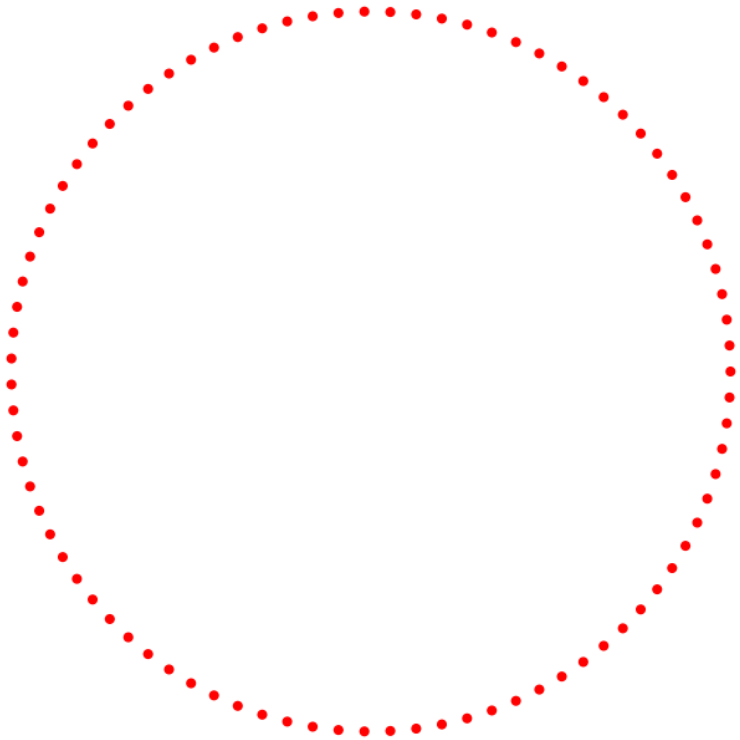
Constant-time CSIDH algorithm [16, 25]

In both the original CSIDH and the Onuki *et al.* variants $e_i \in [\![-m_i, m_i]\!]$, while in Meyer-Campos-Reith variant $e_i \in [\![0, m_i]\!]$. However, in constant-time implementations of CSIDH, the exponents e_i are implicitly interpreted as

$$|e_i| = \underbrace{1 + 1 + \cdots + 1}_{e_i \text{ times}} + \underbrace{0 + 0 + \cdots}_{m_i - e_i \text{ times}},$$

and then it starts by constructing isogenies with kernel generated by $P \in E_A[\ell_i, \pi - \mathtt{sign}(e_i)]$ for e_i iterations, then performs dummy isogeny computations for $(m_i - e_i) = 2k_i$ iterations.

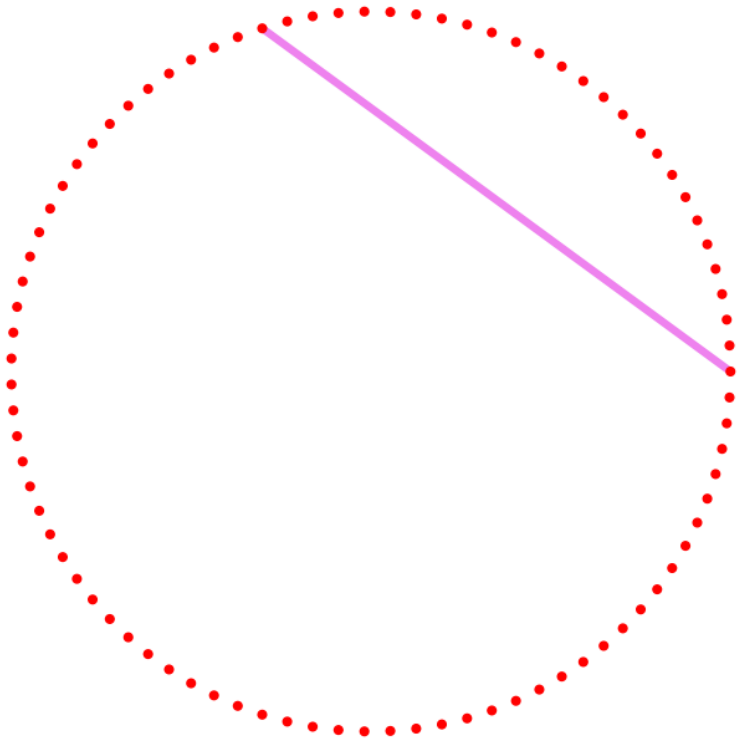
Constant-time CSIDH algorithm [16, 25]



E_0

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]

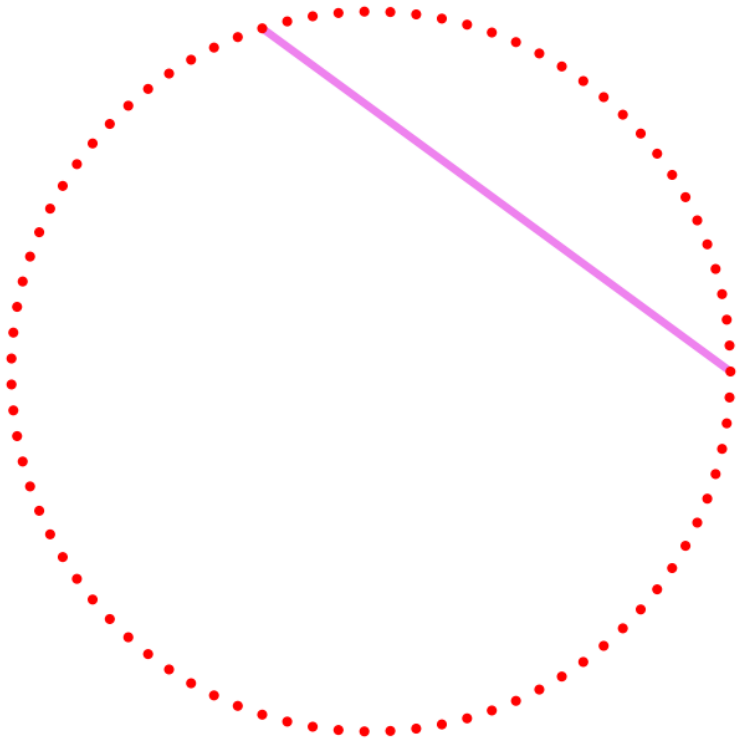


$E_0 \longrightarrow$

$E_{0\times 3653}$

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]

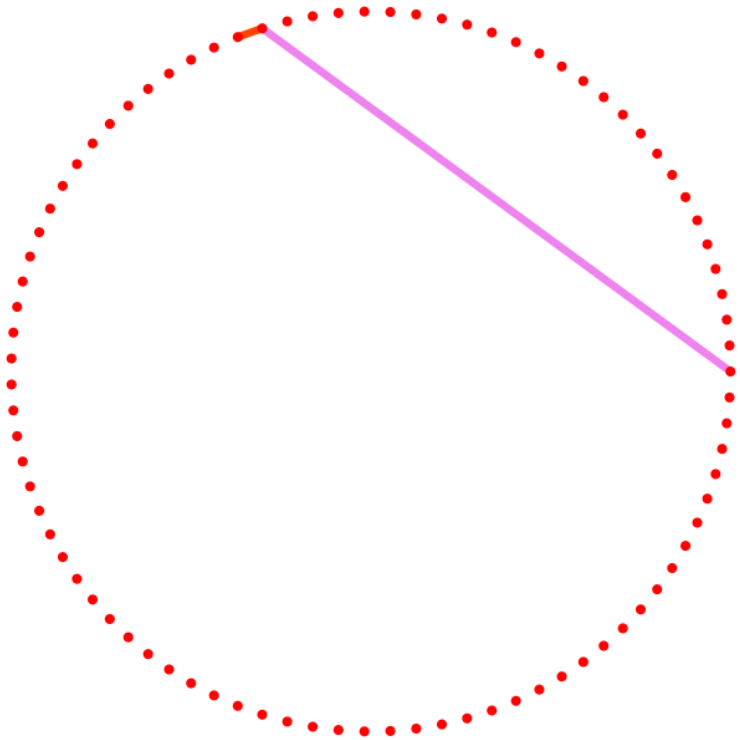


$E_0 \longrightarrow$

$E_{0\times3653}$, dummy

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



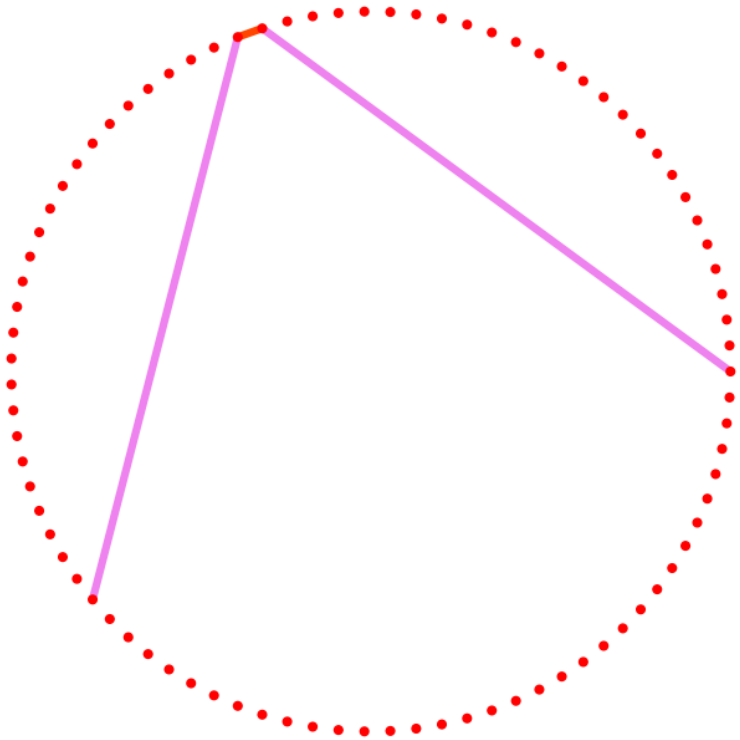
$E_0 \rightarrow$

$E_{0\text{x}3653}$, dummy$\rightarrow$

$E_{0\text{x}25\text{B}3}$

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

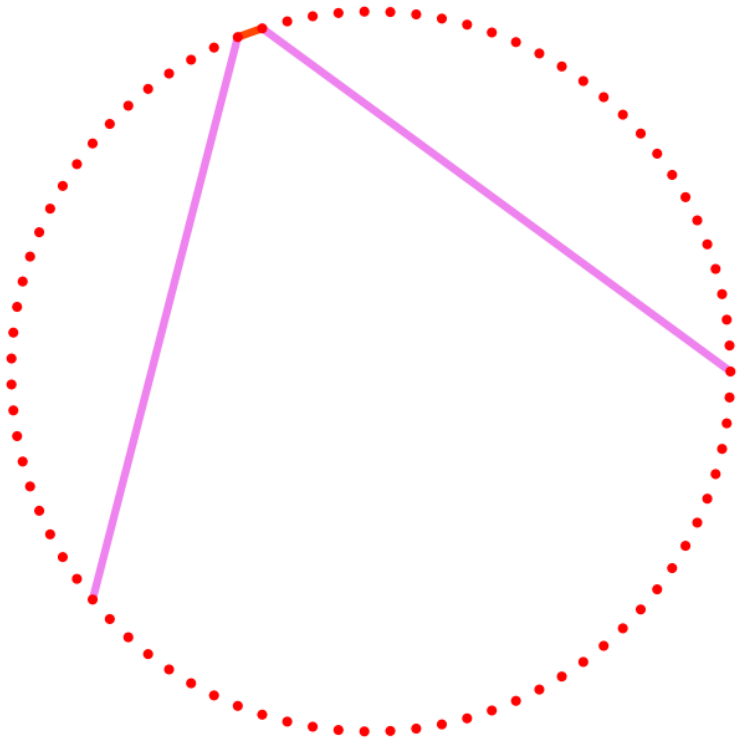
$E_{0\text{x}3653}$, dummy$\rightarrow$

$E_{0\text{x}25\text{B}3} \rightarrow$

$E_{0\text{x}2\text{BF}7}$

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

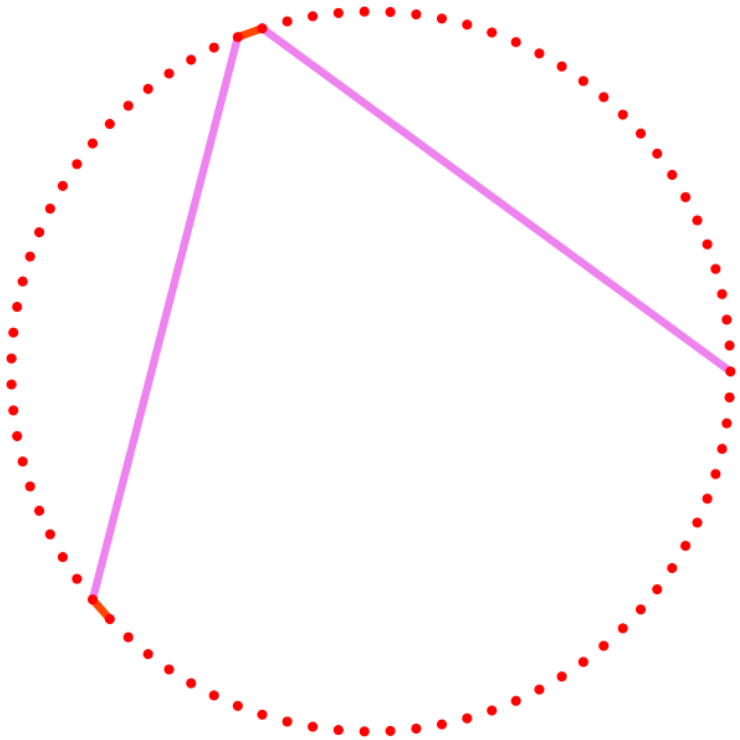
$E_{0\text{x}3653}$, dummy$\rightarrow$

$E_{0\text{x}25\text{B}3} \rightarrow$

$E_{0\text{x}2\text{BF}7}$, dummy

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

E_{0x3653}, dummy $\rightarrow$

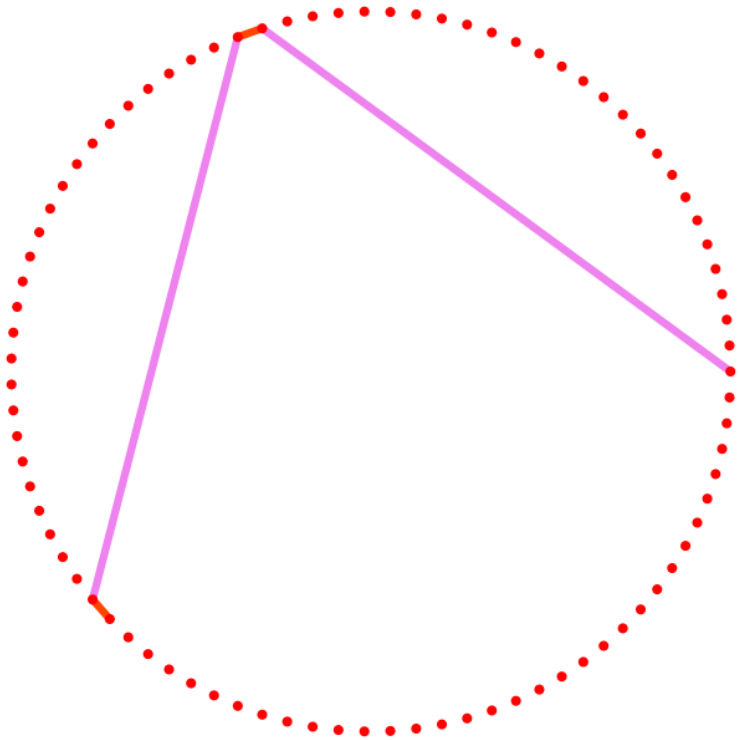
$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy $\rightarrow$

E_{0x56D}

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

E_{0x3653}, dummy$\rightarrow$

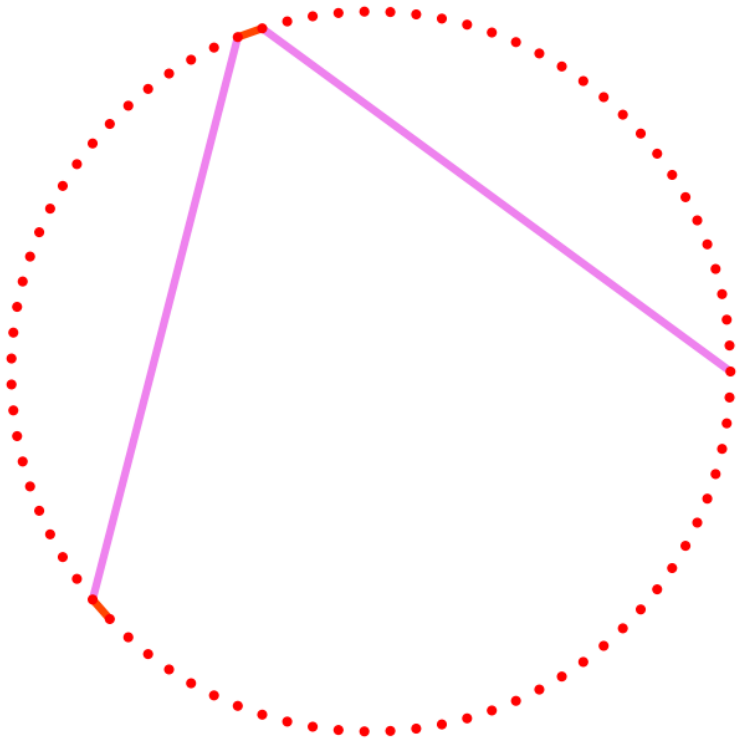
$E_{0x25B3} \rightarrow$

E_{0x2BF7}, dummy$\rightarrow$

E_{0x56D}, dummy

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

$E_{0\text{x}3653}$, dummy$\rightarrow$

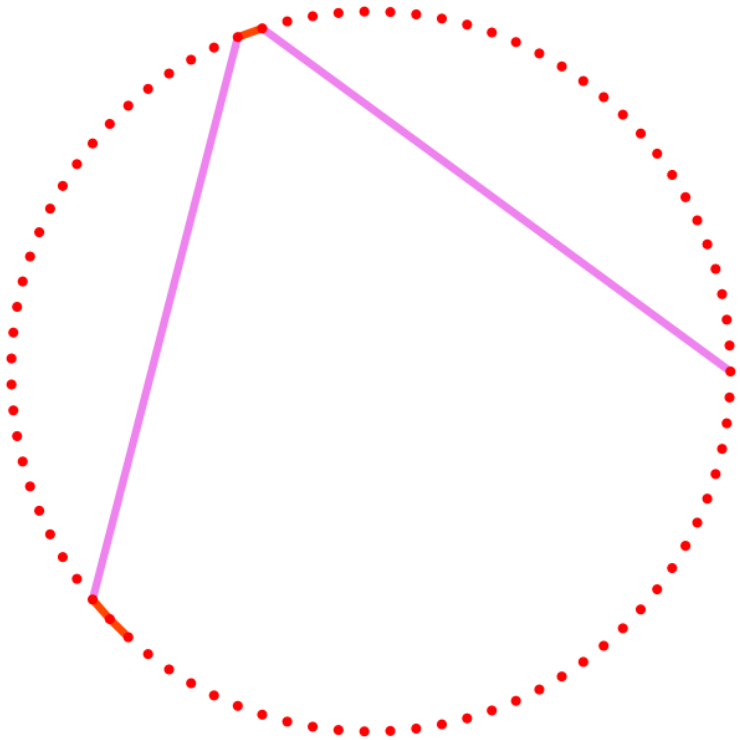
$E_{0\text{x}25\text{B}3} \rightarrow$

$E_{0\text{x}2\text{BF}7}$, dummy$\rightarrow$

$E_{0\text{x}56\text{D}}$, dummy, dummy

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

$E_{0x3653}, \text{dummy} \rightarrow$

$E_{0x25B3} \rightarrow$

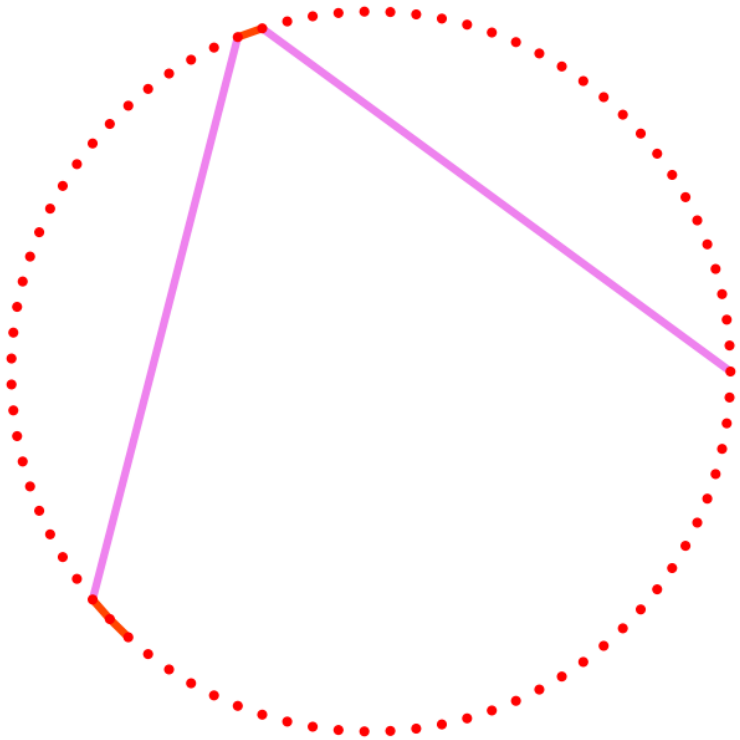
$E_{0x2BF7}, \text{dummy} \rightarrow$

$E_{0x56D}, \text{dummy}, \text{dummy} \rightarrow$

E_{0x24D5}

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

$E_{0\text{x}3653}$, dummy$\rightarrow$

$E_{0\text{x}25\text{B}3} \rightarrow$

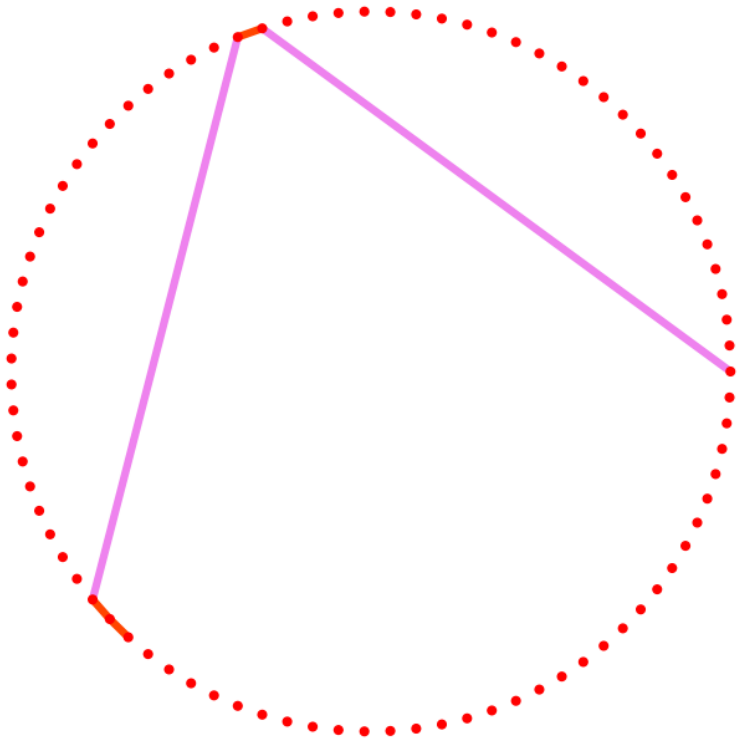
$E_{0\text{x}2\text{BF}7}$, dummy$\rightarrow$

$E_{0\text{x}56\text{D}}$, dummy, dummy$\rightarrow$

$E_{0\text{x}24\text{D}5}$, dummy

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]



$E_0 \rightarrow$

$E_{0\text{x}3653}, \text{dummy} \rightarrow$

$E_{0\text{x}25\text{B}3} \rightarrow$

$E_{0\text{x}2\text{BF}7}, \text{dummy} \rightarrow$

$E_{0\text{x}56\text{D}}, \text{dummy}, \text{dummy} \rightarrow$

$E_{0\text{x}24\text{D}5}, \text{dummy}, \text{dummy}$

Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Constant-time CSIDH algorithm [16, 25]

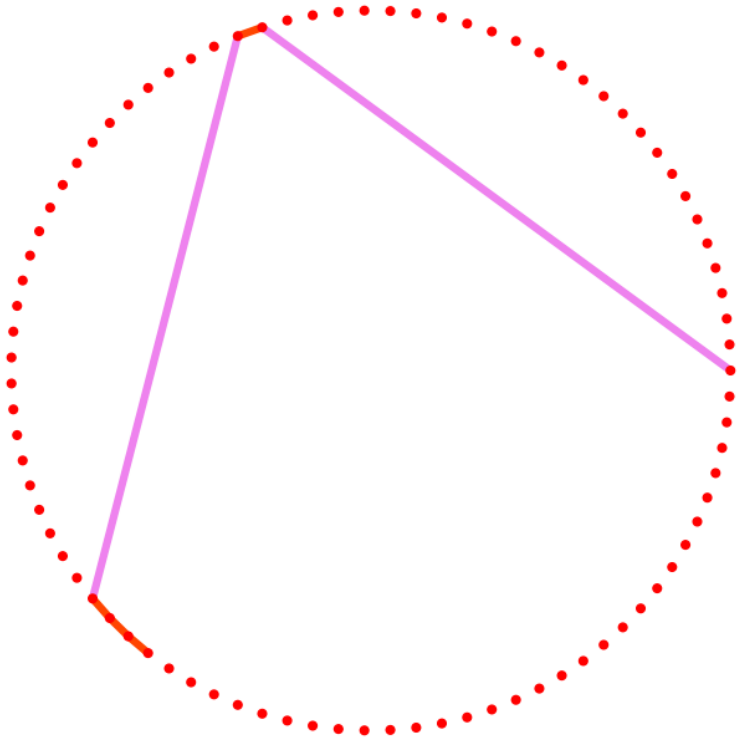


Figure 7: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

$E_0 \rightarrow$

$E_{0\text{x}3653}$, dummy$\rightarrow$

$E_{0\text{x}25\text{B}3} \rightarrow$

$E_{0\text{x}2\text{BF}7}$, dummy$\rightarrow$

$E_{0\text{x}56\text{D}}$, dummy, dummy$\rightarrow$

$E_{0\text{x}24\text{D}5}$, dummy, dummy$\rightarrow$

$E_{0\text{x}280\text{E}}$

Outline

CSIDH with dummy operations

To mitigate power consumption analysis attacks, the constant-time algorithms proposed in [16] and [25] always compute the maximal amount of isogenies allowed by the exponent, using dummy isogeny computations if needed.

This implies that an attacker can obtain information on the secret key by injecting faults into variables during the computation. If the final result is correct, then she knows that the fault was injected in a dummy operation; if it is incorrect, then the operation was real.

Removing dummy operations

For our new approach, the exponents e_i are uniformly sampled from sets

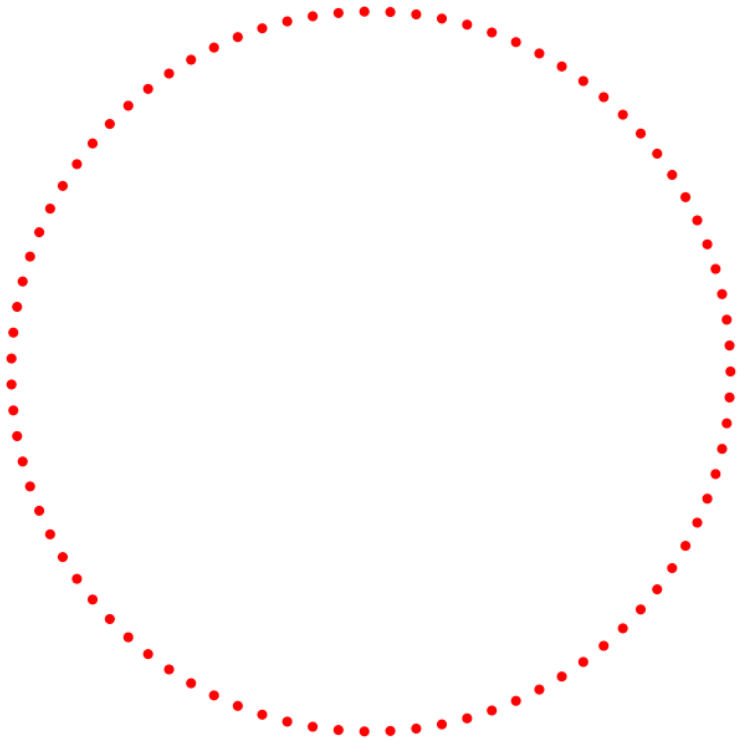
$$\mathcal{S}(m_i) = \{e \mid e = m_i \bmod 2 \text{ and } |e| \leq m_i\},$$

i.e., centered intervals containing only even or only odd integers. Consequently, the exponents e_i can implicitly interpreted as

$$|e_i| = \underbrace{1 + 1 + \cdots + 1}_{e_i \text{ times}} + \underbrace{(1 - 1) - (1 - 1) + (1 - 1) - \cdots}_{m_i - e_i \text{ times}},$$

and then our approach starts by constructing isogenies with kernel generated by $P \in E_A[\ell_i, \pi - \mathtt{sign}(e_i)]$ for e_i iterations, then alternates between isogenies with kernel generated by $P \in E_A[\ell_i, \pi - 1]$ and $P \in E_A[\ell_i, \pi + 1]$ for $(m_i - e_i) = 2k_i$ iterations.

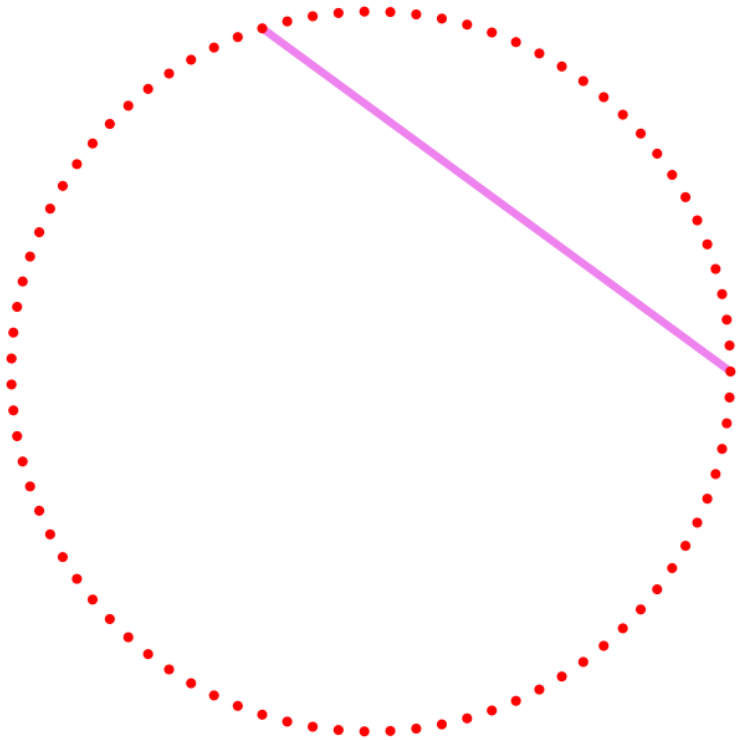
Removing dummy operations



E_0

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

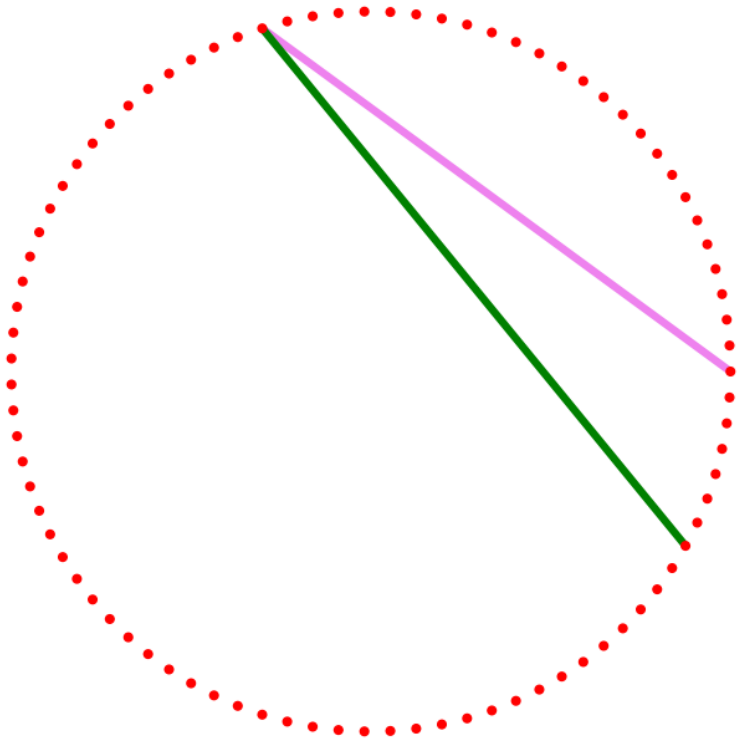
Removing dummy operations



$$E_0 \longrightarrow E_{0\text{x}3653}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

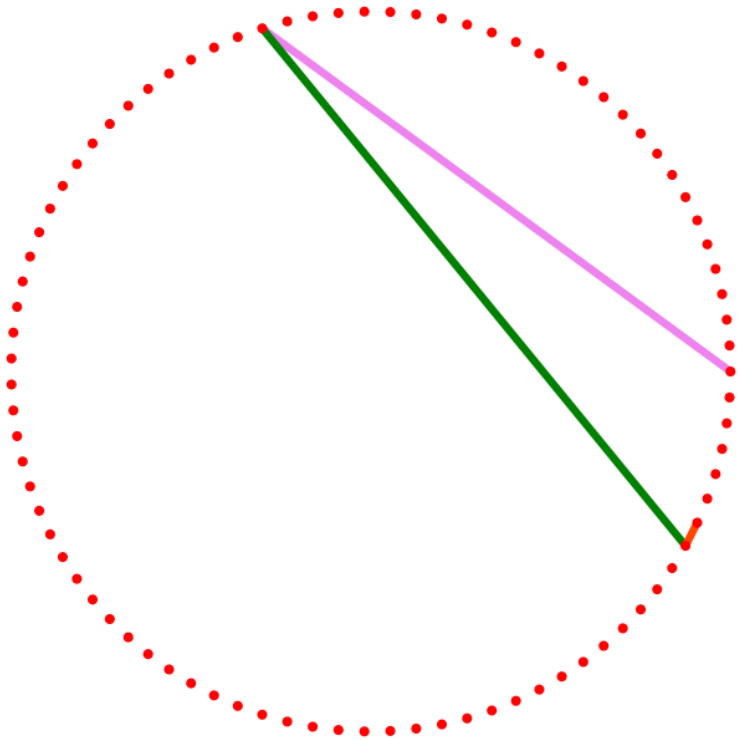
Removing dummy operations



$$E_0 \rightarrow E_{0\times3653} \rightarrow E_{0\times3C4A}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

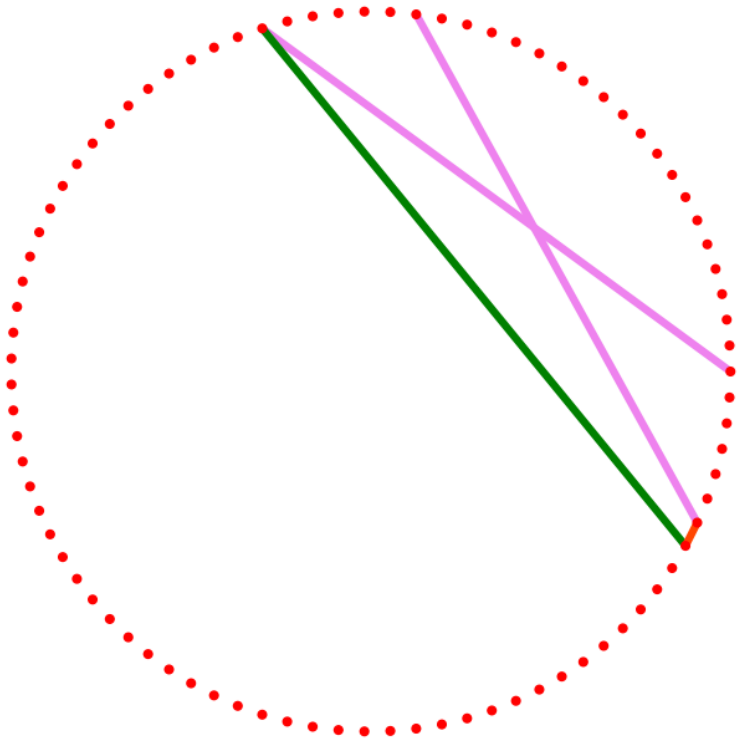
Removing dummy operations



$$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

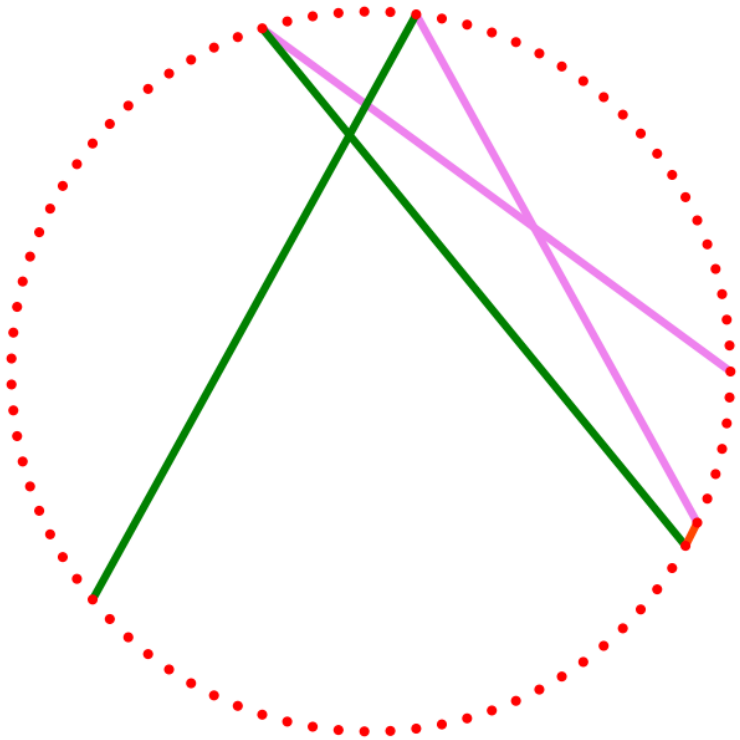
Removing dummy operations



$$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB} \to E_{0\text{x}1404}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

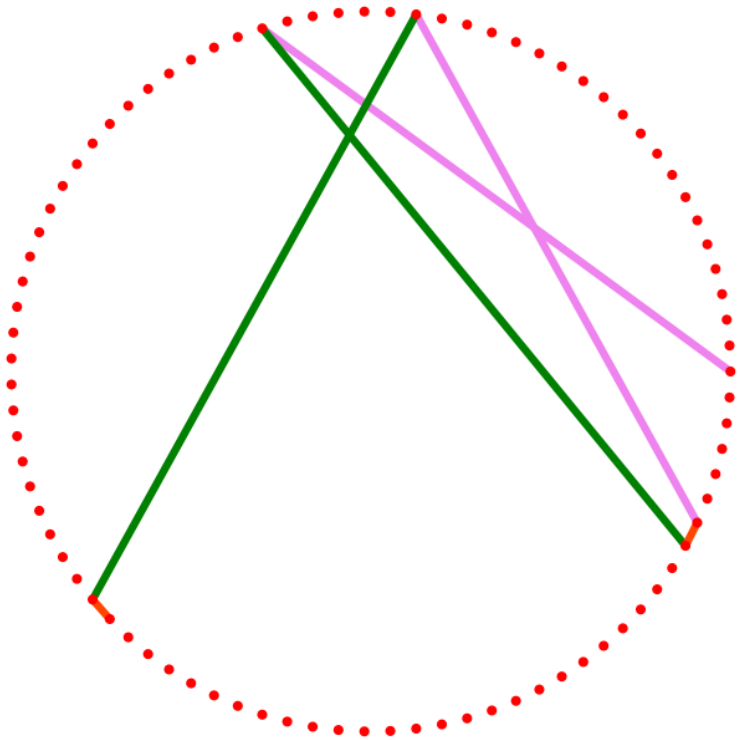
Removing dummy operations



$$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB} \to E_{0\text{x}1404} \to E_{0\text{x}2BF7}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

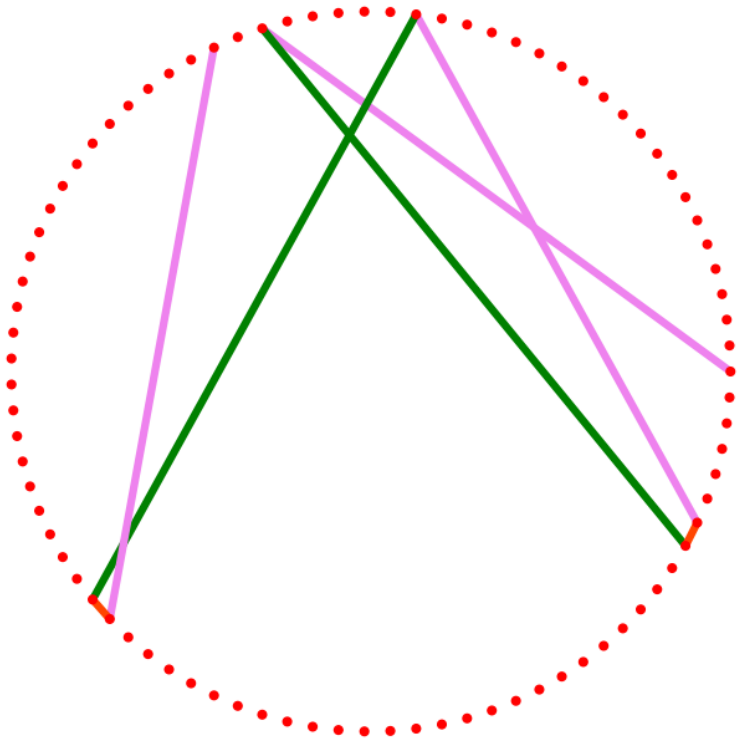
Removing dummy operations



$$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB}$$
$$\to E_{0\text{x}1404} \to E_{0\text{x}2BF7} \to E_{0\text{x}56D}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

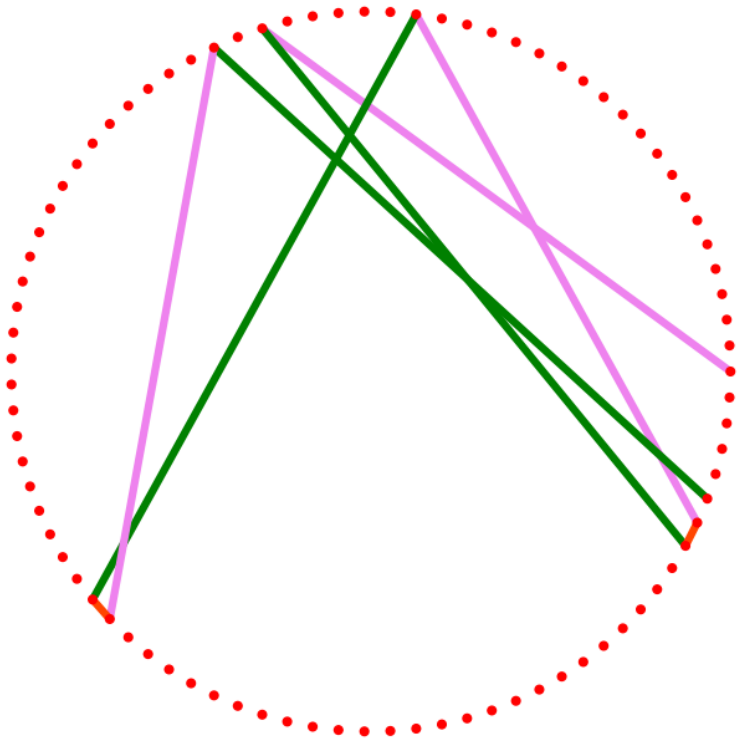
Removing dummy operations



$$E_0 \to E_{0\text{x}3653} \to E_{0\text{x}3C4A} \to E_{0\text{x}5EB} \to E_{0\text{x}1404} \to E_{0\text{x}2BF7} \to E_{0\text{x}56D} \to E_{0\text{x}8EC}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

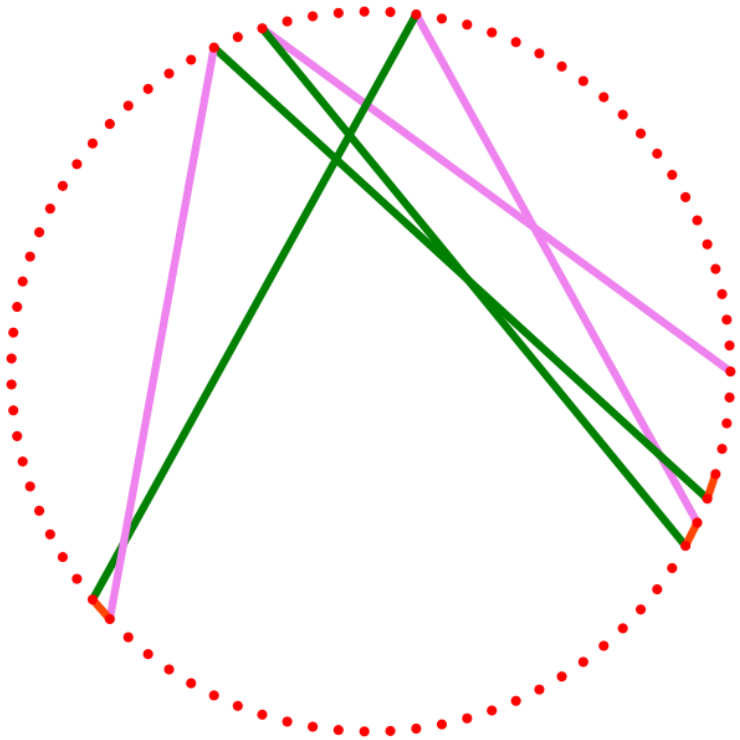
Removing dummy operations



$$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB} \to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D} \to E_{0x8EC} \to E_{0x1D50}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

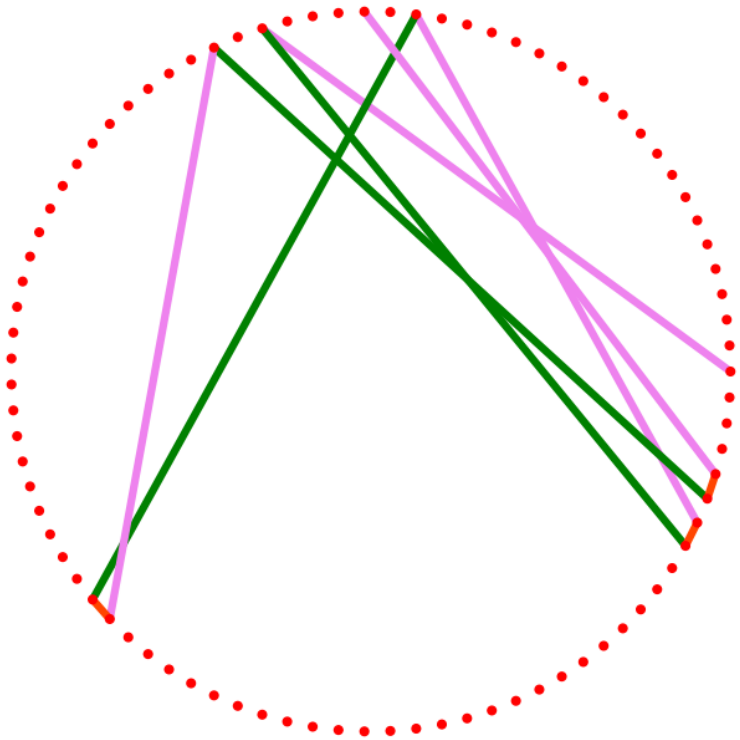
Removing dummy operations



$$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB} \to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D} \to E_{0x8EC} \to E_{0x1D50} \to E_{0x13F5}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

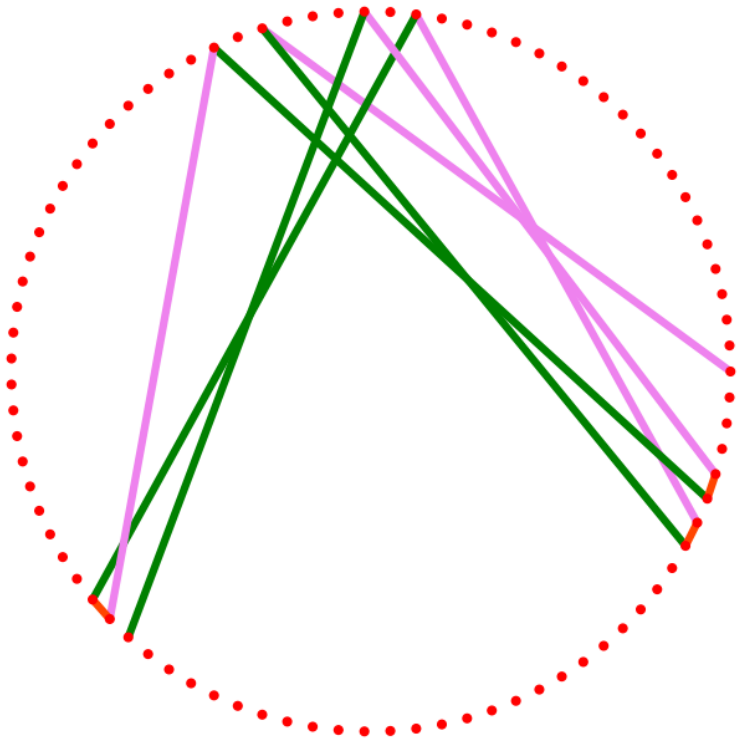
Removing dummy operations



$$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB} \to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D} \to E_{0x8EC} \to E_{0x1D50} \to E_{0x13F5} \to E_{0x1CDD}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

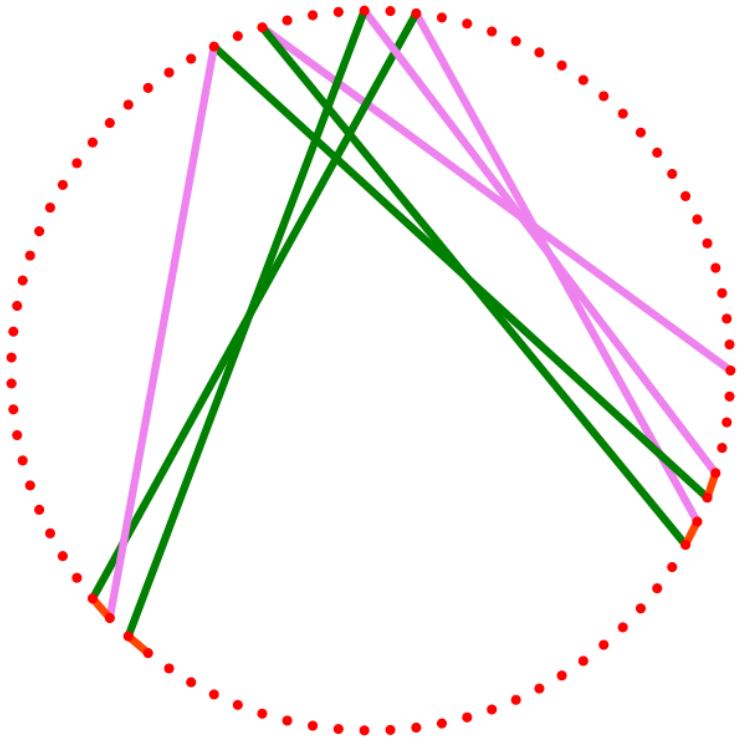
Removing dummy operations



$$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB} \to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D} \to E_{0x8EC} \to E_{0x1D50} \to E_{0x13F5} \to E_{0x1CDD} \to E_{0x24D5}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Removing dummy operations



$$E_0 \to E_{0x3653} \to E_{0x3C4A} \to E_{0x5EB}$$
$$\to E_{0x1404} \to E_{0x2BF7} \to E_{0x56D}$$
$$\to E_{0x8EC} \to E_{0x1D50} \to E_{0x13F5}$$
$$\to E_{0x1CDD} \to E_{0x24D5} \to E_{0x280E}$$

Figure 8: Action evaluation over $\mathbb{F}_p$ with $p = 4 \cdot (5 \cdot 13 \cdot 61) - 1$. Secret integer vector $(4, 0, -2) \in \{-4, -2, 0, 2, 4\}^3$.

Outline

Running-time: field operations

Table 5: Field operation counts for constant-time CSIDH. Counts are given in millions of operations, averaged over 1024 random experiments. The performance ratio uses [16] as a baseline, considers only multiplication and squaring operations, and assumes $M = S$.

Implementation	**CSIDH Algorithm**	**M**	**S**	**A**	**Ratio**
Castryck et al. [15]	unprotected, unmodified	0.252	0.130	0.348	0.26
Meyer–Campos–Reith [16]	unmodified	1.054	0.410	1.053	1.00
Onuki et al. [25]	unmodified	0.733	0.244	0.681	0.67
This work	MCR-style	0.901	0.309	0.965	0.83
	OAYT-style	0.657	0.210	0.691	0.59
	No-dummy	1.319	0.423	1.389	1.19

Running-time: measured clock cycles

Table 6: Clock cycle counts for constant-time CSIDH implementations, averaged over 1024 experiments. The ratio is computed using [16] as baseline implementation.

Implementation	**CSIDH algorithm**	**Mcycles**	**Ratio**
Castryck et al. [15]	unprotected, unmodified	155	0.39
Meyer–Campos–Reith [16]	unmodified	395	1.00
This work	MCR-style	337	0.85
	OAYT-style	239	0.61
	No-dummy	481	1.22

Outline

GHS Weil descent and GLS endomorphism

- The GLS endomorphism extends an efficient endomorphism on the jacobian, and it yields a factor n and n^2 speedup on the smooth divisors search and kernel element's computation steps, respectively.
- Our analysis is backed up by the explicit computation of a DLP defined on a prime order subgroup of a GLS elliptic curve over the field $\mathbb{F}_{2^{5\cdot 31}}$. A **Magma-code** implementation of a standard index-calculus procedure boosted with the GLS endomorphism is able to find this discrete logarithm in about $1,035$ CPU days.
- This is the first work showing that one endomorphism on the elliptic curve is "preserved" by the GHS weil descent technique.

CSSI

- We showed that VW Golden Collision search can be used to solve CSSI.
- First implementations of MITM and Golden collision search CSSI attacks reported.
- The implementations confirm that the performance of these attacks is accurately predicted by their heuristic analysis.
- Our concrete cost analysis of the attacks leads to the conclusion that golden collision search is more cost effective than the meet-in-the-middle attack.

CSSI

- As a consequence, our security analysis has strongly impacted on the post-quantum cryptography community, to the point of being endorsed by the SIKE protocol designers and pushing them to use our proposed smaller prime integer numbers.

Protocol phase		CLN library [19]			CLN + enhancements		
		p_{751}	p_{434}	p_{546}	p_{751}	p_{434}	p_{546}
Key Gen.	Alice	35.7	7.51	13.20	26.9	5.3	10.5
	Bob	39.9	8.32	14.84	30.5	6.0	11.7
Shared Secret	Alice	33.6	7.01	12.56	24.9	5.0	10.0
	Bob	38.4	7.94	14.35	28.6	5.8	11.5

Table 7: Performance of the SIDH protocol. All timings are reported in 10^6 clock cycles, measured on an Intel Core i7-6700 supporting a Skylake micro-architecture. The "CLN + enhancements" columns are for our implementation that incorporates improved formulas for degree-4 and degree-3 isogenies from [17] and Montgomery ladders from [18] into the CLN library.

CSIDH

1) Previous implementations failed at being constant time because of a subtle mistake (Elligator was being used in an insecure way).
2) We fixed the problem, and proposed new improvements, to achieve the most efficient version of CSIDH protected against timing and simple power analysis attacks to date.
3) We proposed a protection against some fault-injection and timing attacks that only comes at a cost of a twofold slowdown.

Outline

Publications

[26] **Jesús-Javier Chi** and Thomaz Oliveira, "Attacking a Binary GLS Elliptic Curve with Magma", *Progress in Cryptology - LATINCRYPT 2015*, LNCS 9230 (2015), 308–326.

[27] Gora Adj, Daniel Cervantes-Vázquez, **Jesús-Javier Chi-Domínguez**, Alfred Menezes, Francisco Rodríguez-Henríquez, "On the Cost of Compu-ting Isogenies Between Supersingular Elliptic Curves", *Selected Areas in Cryptography — SAC 2019*. LNCS 11349 (2018), 322–343.

[28] Daniel Cervantes-Vázquez, Mathilde Chenu, **Jesús-Javier Chi-Domín-guez**, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith, "Stronger and Faster Side-Channel Protections for CSIDH", *Progress in Cryptology - LATINCRYPT 2019*. LNCS 11774 (2019), 173-193

Additionally, the following work has been submitted to the indexed journal *Finite Fields and Their Applications*, which is still under revision.

- **Jesús-Javier Chi-Domínguez**, Francisco Rodríguez-Henríquez, and Be-njamin Smith, "Extending the GLS endomorphism to speedup the GHS Weil descent using Magma".

Outline

Forthcoming research

- Constructive aspects of genus-2 curve based cryptography: GLS endomorphism of a GLS curve $\mathcal{E}/\mathbb{F}_{2^{2n}}$ induces an efficient endomorphism Ψ^* on the jacobian of the image of the GHS Weil descent applied on $\mathcal{E}/\mathbb{F}_{2^{2n}}$.
- How does VW golden collision search CSSI attack behave in a cuda-code implementation on GPU's?
- Analysis of the VW (golden?) collision search CSSI attack but applied to CSIDH protocol must be done.
- Study of practical implications of using radix-tree in VW golden collision search CSSI attack.
- Design of a conservative CSIDH protocol (with respect to quantum attacks).

Thank you for your attention

I look forward to your comments and questions.

e-mail: jjchi@computacion.cs.cinvestav.mx

Elliptic-curve based crypto: moving from cities to cities.

Isogeny-based crypto: moving from planets to planets

References I

- P. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *Journal on Computing*, SIAM 26 (1997), 1484–1509.
- S. Miller, "Use of Elliptic Curves in Cryptography", *Advances in Cryptology - CRYPTO '85*, LNCS 218 (1986), 417–426.
- N. Koblitz, "Elliptic curve cryptosystems', *Mathematics of Computation*, AMS 48 (1987), 203–209.
- M. Velichka, M. Jacobson Jr, and A. Stein, "Computing discrete logarithms in the jacobian of high-genus hyperelliptic curves over even characteristic finite fields", *Mathematics of Computation*, AMS 83 (2014), 935-963.

References II

- M. Jacobson, A. Menezes, and A. Stein, "Solving elliptic curve discrete logarithm problems using Weil descent", *Journal of the Ramanujan Mathematical Socciety*, 16 (2001) 231–260.
- D. Jao et al., "Supersingular isogeny key encapsulation", Round 1 submission, NIST Post-Quantum Cryptography Standardization, November 30, 2017.
- L. De Feo, D. Jao and J. Plût, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies", *Journal of Mathematical Cryptology*, 8 (2014), 209–247.
- P. van Oorschot and M. Wiener, "Improving implementable meet-in-the-middle attacks by orders of magnitude", *Advances in Cryptology — CRYPTO '96*, LNCS 1109 (1996), 229–236.

References III

- P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", *Journal of Cryptology*, 12 (1999), 1–28.
- L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", *Symposium on the Theory of Computing*, ACM (1996), 212–219.
- C. Zalka, "Grover's quantum searching algorithm is optimal", *Physical Review A*, 60 (1999), 2746–2751.
- National Institute of Standards and Technology, "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process", December 2016. Available from `https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf`.

References IV

- S. Tani, "Claw finding algorithms using quantum walk", *Theoretical Computer Science*, 410 (2009), 5285–5297.
- S. Jaques and J. Schanck: "Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE", Cryptology ePrint Archive: Report 2019/103. Available `http://eprint.iacr.org/2019/103`.
- W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes, "CSIDH: An Efficient Post-Quantum Commutative Group Action", *Advances in Cryptology — ASIACRYPT 2018*, LNCS 11274 (2018), 395–427.
- M. Meyer, F. Campos, and S. Reith, "On Lions and Elligators: An Efficient Constant-Time Implementation of CSIDH", *Post-Quantum Cryptography — PQCrypto 2019*, LNCS 11505 (2019), 307–325.

References V

- C. Costello and H. Hisil, "A simple and compact algorithm for SIDH with arbitrary degree isogenies", *Advances in Cryptology — ASIACRYPT 2017*, LNCS 10624 (2017), 303–329.
- A. Faz-Hernández, J. López, E. Ochoa-Jiménez and F. Rodríguez-Henríquez, "A faster software implementation of the supersingular isogeny Diffie-Hellman key exchange protocol", *IEEE Transactions on Computers*, 67 (2018), 1622–1636.
- C. Costello, P. Longa and M. Naehrig, "Efficient algorithms for supersingular isogeny Diffie-Hellman", *Advances in Cryptology — CRYPTO 2016*, LNCS 9814 (2016), 572–601.
- M. Trimoska, Sorina Ionica, and G. Dequen, "Time-Memory Trade-offs for Parallel CollisionSearch Algorithms", Cryptology ePrint Archive: Report 2017/581. Available `http://eprint.iacr.org/2017/581`.

References VI

- P. Gaudry, F. Hess, and N. Smart, "Constructive and destructive facets of weil descent on elliptic curves", *Journal of Cryptology*, 15 (2002), 19–46.
- S. Galbraith, F. Hess, and N. Smart, "Extending the GHS weil descent attack", *Advances in Cryptology - EUROCRYPT 2002*, LNCS 2332 (2002), 29-44.
- F. Hess, "The GHS Attack Revisited", *Advances in Cryptology - EUROCRYPT 2003*, LNCS 2656 (2003), 374–387.
- F. Hess, "Generalising the GHS Attack on the Elliptic Curve Discrete Logarithm Problem", *LMS Journal of Computation and Mathematics*, 7 (2004), 167–192.

References VII

- H. Onuki, Y. Aikawa, T. Yamazaki, and T. Takagi, "A Faster Constant-time Algorithm of CSIDH keeping Two Torsion Points", Cryptology ePrint Archive: Report 2019/353. Available `http://eprint.iacr.org/2019/353`.
- J. Chi and T. Oliveira, "Attacking a binary GLS elliptic curve with magma", *Progress in Cryptology - LATINCRYPT, 2015*, LNCS 9230 (2015), 308–326.
- G. Adj, D. Cervantes-Vázquez, J. Chi-Domínguez, A. Menezes and F. Rodríguez-Henríquez, "On the cost or computing isogenies between supersingular elliptic curves", *Selected Areas in Cryptography — SAC 2018*. LNCS 11349 (2019), 322–343.

References VIII

- D. Cervantes-Vázquez, M. Chenu, J.-J. Chi-Domín-guez, L. De Feo, F. Rodríguez-Henríquez, and Benjamin Smith, "Stronger and Faster Side-Channel Protections for CSIDH", *Progress in Cryptology - LATINCRYPT 2019*. LNCS 11774 (2019), 173-193

Index-calculus based algorithms I

Let $A_{s'}$ be the number of irreducible divisors $\mathrm{div}(u, v) \in \mathrm{Jac}_H(\mathbb{F}_q)$ with $\deg u = s'$, then

$$A_{s'} \approx \frac{1}{2}\left(\frac{1}{s'}\sum_{d|s'}\mu\left(\frac{s'}{d}\right)q^d\right), \tag{2}$$

where μ denotes the Möbius function, i.e., for every positive integer n we have

$$\mu(n) = \begin{cases} (-1)^k & \text{if } n \text{ is square free and has} \\ & k \text{ different prime factors,} \\ 0 & \text{if } n \text{ is not square free.} \end{cases} \tag{3}$$

Index-calculus based algorithms II

Consequently, $\#\mathcal{F}_s \approx \sum_{i=1}^{s} A_i$. On the other hand, the number of s-smooth divisors $\mathrm{div}(u, v) \in \mathrm{Jac}_{\mathcal{H}}(\mathbb{F}_q)$ with $\deg u \leq g$ is given as

$$M(g, s) = \sum_{i=1}^{g} \left([x^i] \prod_{s'=1}^{s} \left(\frac{1 + x^{s'}}{1 - x^{s'}} \right)^{A_{s'}} \right), \tag{4}$$

where [.] denotes the coefficient operator. However, when $A_{s'}$ is known, $M(g, s)$ can be computed by finding the first $(g+1)$ terms of the Taylor expansion of $\prod_{s'=1}^{s} \left(\frac{1+x^{s'}}{1-x^{s'}} \right)^{A_{s'}}$ around $x = 0$, and adding the coefficients of x, x^2, ..., x^g. Thus, the expected number of random-walk iterations before a t-smooth divisor is encountered is

$$E(s) = \frac{\#\mathrm{Jac}_{\mathcal{H}}(\mathbb{F}_q)}{M(g, s)} \approx \frac{q^g}{M(g, s)}. \tag{5}$$

Index-calculus based algorithms III

In addition, the expected number of random-walk iterations before $\left(\#\mathcal{F}_s + \epsilon\right)$ relations are generated is

$$T(s) = \left(\#\mathcal{F}_s + \epsilon\right) E(s). \tag{6}$$

With respect to the linear algebra task, the running time of the Lanczo's algorithm employed by magma can be approximated by, $L(s) \approx d \cdot \left(\#\mathcal{F}_s + \epsilon\right)^2$, where d denotes the per-row density of the matrix M. In fact, it can be shown that $d \leq g$.

GHS weil descent technique

The genus-g algebraic curve $\mathcal{C}/\mathbb{F}_{2^n}$ (not necessary hyperelliptic) that the generalized GHS (gGHS) Weil descent technique computes can be obtained by constructing the Weil restriction $\mathcal{A}/\mathbb{F}_{2^n}$ of $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}$, intersecting $\mathcal{A}/\mathbb{F}_{2^n}$ with $(\ell-1)$-dimensional hyperplanes to obtain a subvariety $\mathcal{A}'/\mathbb{F}_q$ of $\mathcal{A}/\mathbb{F}_{2^n}$, and finding an irreducible component $\mathcal{C}/\mathbb{F}_{2^n}$ of $\mathcal{A}'/\mathbb{F}_{2^n}$ (for more details see [21, 22, 23, 24]).

The Weil restriction is just writing the equation of $\mathcal{E}/\mathbb{F}_{2^{n\times\ell}}$ in terms of a $\mathbb{F}_{2^\ell}$-basis of $\mathbb{F}_{2^{n\times\ell}}$.

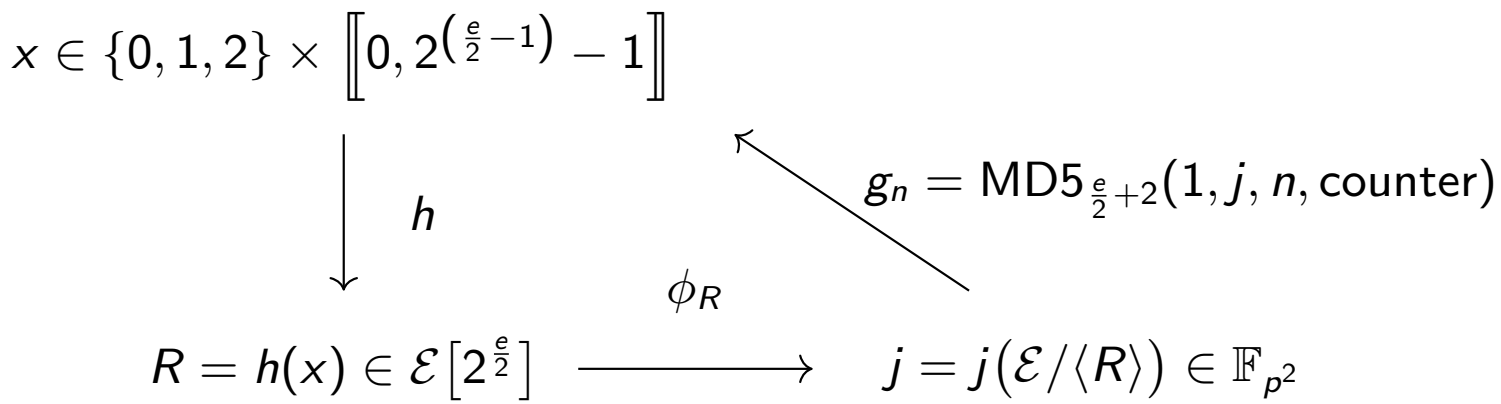
The explict description of the extending endomorphism is given as follows:

$$\Psi^\star\big(\mathrm{div}(u,v)\big) = \mathrm{div}\left(\delta_1^{\deg u}\cdot({}^\sigma u)\Big(\frac{x}{\delta_1}\Big), \delta_3({}^\sigma v)\Big(\frac{x}{\delta_1}\Big) + \delta_4\big({}^\sigma(h \mod u)\big)\left(\frac{x}{\delta_1}\right)\right).$$

for some $\delta_1, \delta_3, \delta_4 \in \mathbb{F}_{2^n}$.

CSSI-based random function f_n I

f_n is defined as the composition $g_n \circ \phi_R \circ h$.



Random points

Algorithm 1: Constant-time projective Elligator

Input: A supersingular curve $\mathcal{E}_{(A':C')} : C'y^2 = C'x^3 + A'x^2 + C'x$ over $\mathbb{F}_p$, and a random element $u \in \{2, \ldots, \frac{p-1}{2}\}$.

Output: A pair of points $\mathsf{T}_+ \in \mathcal{E}_{(A':C')}[\pi - 1]$ and $\mathsf{T}_- \in \mathcal{E}_{(A':C')}[\pi + 1]$.

$t \leftarrow A'\big((u^2-1)u^2A'^2C' + ((u^2-1)C')^3\big)$;

$a \leftarrow \texttt{isequal}(t, 0)$; // $t = 0$ iff $A' = 0$

$\alpha, \beta \leftarrow 0, u$;

$\texttt{cswap}(\alpha, \beta, a)$; // $\alpha = 0$ iff $A' \neq 0$

$t' \leftarrow t + \alpha(u^2 + 1)$; // $t' \neq 0$

$\mathsf{T}_+ \leftarrow (A' + \alpha C'(u^2-1) : C'(u^2-1))$;

$\mathsf{T}_- \leftarrow (-A'u^2 - \alpha C'(u^2-1) : C'(u^2-1))$;

$b \leftarrow \texttt{Legendre\_symbol}(t', p)$; // $b = \pm 1$

$c \leftarrow \texttt{isequal}(b, -1)$;

$\texttt{cswap}(\mathsf{T}_+, \mathsf{T}_-, c)$;

return $(\mathsf{T}_+, \mathsf{T}_-)$;