Santiago 2026 CIMPA School: the isogeny group action

Valerie Gilchrist

Royal Military Academy of Belgium

gilchrist.valerie@gmail.com

July 2026

In what follows we give a brief introduction to the topic of isogeny group actions. Several of the examples and exercises included in these notes require the use of SageMath.

1. Discrete logarithm

Asymmetric cryptography requires the use of *asymmetric functions*. These are mathematical functions that are easy and efficient to compute, but for which their inversion is unreasonable. Among the first examples of asymmetric functions was the *discrete logarithm*.

Problema 1 (Discrete logarithm)**.** *Let $G = \langle g \rangle$ be a cyclic group of order N. Let $a \in \mathbb{Z}_N$.*

Given (g, g^a), compute a.

Take $G = \mathbb{Z}_p$ for p a large prime. In this case, Problem 1 is thought to be hard for classical computers, whereas computing $(g, a) \mapsto g^a$ can be done very quickly. For this reason, Diffie and Hellman in the year 1976 used discrete logarithms to construct a key exchange[1].

Protocolo 1 (Diffie-Hellman)**.** *Select a commutative group $G = \langle g \rangle$.*

Alicia and Beto start by choosing secret integers $a, b \in \mathbb{Z}$. They compute and publish their public keys g^a, g^b respectively. With these public keys, and their secret integers, the two can compute the secret object $(g^a)^b = (g^b)^a$.

The following exercise was written by Lorenz Panny [27].

[1] A similar protocol was developed within the UK's GCHQ in 1969, but they did not publish it until 1977.

Ejercicio 1.1 Write Sage code that implements the Diffie-Hellman protocol where the group used is the group of points from the elliptic curve $E : y = f(x)$, where $f(x) = x^3 + 486662x^2 + x$ over the finite field $\mathbb{F}_p$ with $p = 2^{255} - 19$. Use $G = (9, y)$ as a generator.

You can use the following starter code to begin.

```
p = 2^255-19
E = EllipticCurve(GF(p), [0,486662,0,1,0])

R.<X> = GF(p)[]
f = X^3 + 486662*X^2 + X
y = Mod(f(9), p).sqrt()
G = E(9, y)
```

For the protocol to be secure, it is necessary that the discrete logarithm problem (Problem 1) is hard. In addition to this, however, we also need that given (g, g^a, g^b), nothing can be learned about the secret g^{ab}. This problem is called the *computational Diffie-Hellman problem*.

Problema 2 (Computational Diffie-Hellman)**.** *Let $G = \langle g \rangle$ be a cyclic group of order N. Let $a, b \in \mathbb{Z}_N$.*

Given (g, g^a, g^b), compute g^{ab}.

There also exists the weaker *decisional Diffie-Hellman problem*.

Problema 3 (Decisional Diffie-Hellman)**.** *Let $G = \langle g \rangle$ be a cyclic group of order N. Let $a, b, c \in \mathbb{Z}_N$.*

Given (g, g^a, g^b, g^c), decide whether $c = ab$.

Ejercicio 1.2 What is the relationship between Problem 1 and Problem 2? Are they equivalent?

Ejercicio 1.3 Let E be an elliptic curve, $P, Q \in E[N]$. A *pairing* is a bilinear map that can be defined on E (or other groups) of the form $e : E \times E \to \mu_N$ where $e(P, [r]Q) = e([r]P, Q) = e(P, Q)^r$.

Use a pairing to give an example in which Problem 3 is easy but Problem 2 remains hard.

Is it important for Protocol 1 that Problem 3 is hard?

Diffie-Hellman continues to be used today, especially with elliptic curves. These groups allow for very efficient computations and require little memory. For example, elliptic-curve Diffie-Hellman is used by Bitcoin, TLS, WhatsApp, Signal, and many other applications. The problem that is drawing closer every day comes with the development of quantum computers. With Peter Shor's algorithm [33], an adversary could efficiently compute discrete logarithms. This means that Diffie-Hellman, using any group, would not be secure against quantum adversaries.

Ejercicio 1.4 Consider the Diffie-Hellman protocol using points on an elliptic curve (ECDH), as in the preceding exercise. Shor's algorithm would allow calculating the *period* of a cyclic function. For example, for the function $f(a,b) := [a]P + [b]Q$, Shor's algorithm can compute r_a, r_b where $f(a+r_a, b+r_b) = f(a,b)$.

Using Shor's algorithm, describe an attack against ECDH.

Now we want to explore protocols that could replace Diffie-Hellman, but which have security against quantum adversaries. This type of cryptography is called *post-quantum cryptography.*

2. Group actions

Group actions, previously called *hard homogeneous spaces* by Couveignes [13], will give us a way to generalize the action used in Diffie-Hellman.

Definition 1 (Group action)**.** *Let G be a group, and X a set. A map*

$$\star : G \times X \to X$$

is a group action if:

1. *there exists an element $e \in G$ where $e \star x = x$ for all $x \in X$;*
2. *for any $g, h \in G$, $g \star (h \star x) = gh \star x$.*

In the original Diffie-Hellman, one computes g^a for a group element $g \in G$, and an integer a. This is an example of a group action where X is also a group. Here, the group of multiplicative integers is acting on a group through exponentiation. Therefore Diffie-Hellman can be described more generally using group actions.

Protocolo 2 (Diffie-Hellman with group action)**.** *Select a group action $(\star, G, X)$. Select an element $x \in X$. Let $(\star, G, X, x)$ be the public parameters.*

Alicia and Beto select their secret group elements $g_A, g_B \in G$. Each computes and publishes $g_A \star x, g_B \star x$ respectively. Now Beto can compute $g_B \star (g_A \star x)$ and Alicia can compute $g_A \star (g_B \star x)$. If G is commutative, then Alicia and Beto can compute the same object $(g_B \cdot g_A) \star x = (g_A \cdot g_B) \star x$.

Unfortunately, group actions where X is also a group are not secure against Shor's attack. Because of this, it is necessary to study group actions in which X is not a group. In addition to this, we are still missing some properties for the action to be useful in cryptography. These properties are described in the following definition taken from [3].

Definition 2 (Restricted effective group action (REGA))**.** *A restricted effective group action is a group action $\star : G \times X \to X$, together with some generating set of $G = \langle g_1, \dots g_n \rangle$, where G is finite and $n = poly(\log(|G|))$, and where there exist efficient polynomial-time algorithms to compute the following tasks:*

(a) *Membership test: decide whether a given input represents an element of* X*;*

(b) *Unique representation of sets: for any* $x \in X$*, compute a canonical representation* $\hat{x}$ *of* x*;*

(c) *Origin: compute the distinguished point* $x_0 \in X$ *known as the origin;*

(d) *Group action: for any* $i \in [n], x \in X$*, compute* $(g_i \star x)$ *and* $(g_i^{-1} \star x)$*.*

The notion of restricting the group action is seen in property (d). An effective group action (EGA) that is not restricted can evaluate $g \star x$ for any g, not just the generators g_i.

All these properties ensure that Protocol 2 can be performed efficiently, but a description of security is missing. Briefly, as in Diffie-Hellman, we can see that we would want it to be hard for an adversary to compute g_A given access to $(x, g_A \star x)$. Due to Couveignes [13], this problem is called *vectorization*. In the literature, this problem can also be referred to as the *Group Action Inverse Problem* (GAIP).

Problema 4 (Vectorization)**.** *Given a group action* $(G, X, \star)$*, and set elements* $x, g \star x$ *where* $x \in X, g \in G$*, recover* g*.*

However, an adversary could still attack the system without needing to recover the exact secret keys. We call this notion *parallelization*, which is reminiscent of computational Diffie-Hellman.

Problema 5 (Parallelization)**.** *Given a group action* $(G, X, \star)$*, and set elements* $x, g_1 \star x, g_2 \star x$ *where* $x \in X, g_1, g_2 \in G$*, compute* $(g_1 \cdot g_2) \star x$*.*

We define a *cryptographic group action* as a REGA such that the problems of *vectorization* and *parallelization* are hard. From here on, unless specifically indicated otherwise, the group action is assumed to be cryptographic.

The most studied post-quantum cryptographic group actions come from lattices, codes, and isogenies. Of these, only the actions from isogenies are commutative.

3. Isogenies

We will explore how to construct group actions using isogenies following the example from De Feo's presentation [17].

3.1. Recall of isogenies

An introduction to isogenies was offered previously in the school, but here we recall some of the fundamental definitions. For more details, see the notes by Amalia Pizarro Madariaga.

Isogenies are geometric objects that describe relationships between elliptic curves.

Definition 3. *Let E_0, E_1 be two elliptic curves. An* isogeny *from E_0 to E_1 is a rational (and non-constant) map*

$$\phi : E_0 \to E_1$$

where $\phi(O_{E_0}) = O_{E_1}$.

Two curves E_0, E_1 are said to be isogenous *if there exists an isogeny from E_0 to E_1.*

Ejemplo 1. *Take for example multiplication by an integer n.*

$$\phi : P \mapsto [n]P$$

The points on an elliptic curve form a group and thus are closed under multiplication by an integer. This means that ϕ sends points from a curve to the same curve,

$$\phi : E \to E.$$

This type of isogeny is called an endomorphism.

Ejemplo 2. *An example of an isogeny that is not an endomorphism could be*

$$\varphi : E_0 \to E_1,$$

$$(x, y) \mapsto (\frac{x^2 + 1}{x}, \frac{x^2 y - y}{x^2}).$$

In general, it is difficult to construct isogenies using rational expressions. For this reason, it is very common to construct them using their *kernel*, denoted $\ker(\varphi)$. The kernel of $\varphi : E_0 \to E_1$ is $\ker \varphi = \{P \in E_0 : \varphi(P) = O_{E_1}\}$. The kernel is sufficient to define the isogeny using Vélu's formulas [34]. In most cases, the size of the kernel is called the *degree* of the isogeny, denoted by $\deg(\varphi)$.

In the preceding example, the isogeny φ can be constructed using a point P of order 2. This means that $[2]P = O_{E_1}$. Then $\ker(\varphi) = \{O_{E_0}, P\}$ and $\deg(\varphi) = |\ker(\varphi)| = 2$.

Here, we were able to construct an isogeny very easily, by selecting a point P, and defining $\ker(\varphi) := \langle P \rangle$. This construction also gives us information about the curve E_1 because $E_1 \simeq E_0/\ker(\varphi)$ (over $\bar{\mathbb{F}}_q$). For points of low order, this method works well. In the case of higher-degree isogenies, it is sometimes possible to construct them using several smaller isogenies. For example, if we want to construct an isogeny of degree ℓ^k, where ℓ is a small prime, we can do so with k isogenies of degree ℓ each. In Figure 1, the composition of the isogenies $\varphi_k \circ \varphi_{k-1} \circ \cdots \circ \varphi_1$ allows constructing an isogeny from E_0 to E_k of degree $\prod_{i=1}^{k} \deg(\varphi_i)$.

So far we have seen how to construct isogenies of degrees with small factors. But this method does not give us control over the final curve, E_k. In general, given the curves E_0, E_k, it is extremely difficult to construct an isogeny $\phi : E_0 \to E_k$. This is the reason why cryptography can be constructed using isogenies. The main idea is that someone can select an isogeny $\phi : E_0 \to E_k$ as their private key, and publish the curves E_0 and E_k as their public keys.

$$E_0 \xrightarrow{\varphi_1} E_1 \xrightarrow{\varphi_2} \cdots \xrightarrow{\varphi_k} E_k$$

Figura 1: Isogeny chain.

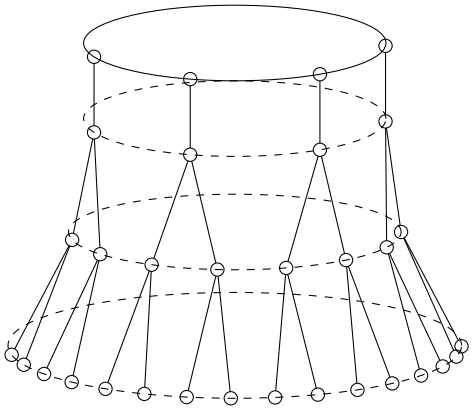

Figura 2: 2-isogeny volcano.

3.2. Isogeny graphs

Consider a graph in which the vertices are elliptic curves and the edges are isogenies of a particular degree, ℓ. This means that two vertices are connected if and only if the curves corresponding to the vertices are isogenous of degree ℓ.

Ejercicio 3.1 According to Tate's theorem, two curves are isogenous over $\mathbb{F}_q$ if and only if they have the same number of points over $\mathbb{F}_q$.

Use Sage to verify Tate's theorem.

a. Select an elliptic curve over the field $\mathbb{F}_q$.

b. Construct an isogeny.

c. Count the number of points on the domain and codomain of the isogeny over $\mathbb{F}_q$.

Consider the graph of degree-ℓ isogenies over $\mathbb{F}_{p^2}$. One of two things can occur. The first option is that there are one or several non-connected components. In this case, each subset has a cycle in which each vertex is the root of a tree. The example of this structure in Figure 2 visually demonstrates why it is called a *volcano*. The curves in this graph are called *ordinary*.

The second option is much less common. In this case the graph is finite, connected, and $(\ell+1)$-regular. The curves in this graph are called *supersingular*. This graph has very good mixing properties. For this reason, it is very common to use supersingular curves in post-quantum protocols.

```
# to check if a curve is supersingular
E = EllipticCurve(GF(7), [0,0,0,1,0]) # y^2 = x^3 + x
E.is_supersingular()
E.is_ordinary()
```

Another reason why supersingular curves are widely used in cryptography is because the number of points over $\mathbb{F}_p$ on these curves is easy to compute: $|E(\mathbb{F}_p)| = p + 1$. This control over the number of points can be attractive for ensuring that points of low order exist over $\mathbb{F}_p$.

```
# how many points there are on E in GF(p)
p = random_prime(2^5)
p
E = EllipticCurve(GF(p), [0,0,0,1,0])
E.is_supersingular()
E.cardinality() == p+1
```

3.3. Endomorphism rings

Let E, E' be two elliptic curves. The functor $\mathrm{Hom}(E, E')$ describes the homomorphisms (isogenies) from E to E'. Let $\varphi, \psi \in \mathrm{Hom}(E, E')$. One can immediately see that $(\varphi + \psi)(P) = \varphi(P) + \psi(P)$ is also a member of $\mathrm{Hom}(E, E')$. In fact, for integers a, b, $(a\varphi + b\psi)(P) = [a]\varphi(P) + [b]\psi(P) \in \mathrm{Hom}(E, E')$. This shows that $\mathrm{Hom}(E, E')$ is an additive group.

Now consider $\mathrm{Hom}(E, E)$, which describes the endomorphisms of E. If $\varphi, \psi \in \mathrm{Hom}(E, E)$, then the composition $\varphi \circ \psi(P) = \varphi(\psi(P)) \in \mathrm{Hom}(E, E)$. Composition serves to define a notion of multiplication in $\mathrm{Hom}(E, E)$, which turns out to be sufficient to conclude that $\mathrm{Hom}(E, E) = \mathrm{End}(E)$ is a ring.

Ejercicio 3.2 Give a proof that $\mathrm{End}(E)$ is a ring.

For each elliptic curve E, the endomorphism ring $\mathrm{End}(E)$ can have one of three structures:

1. $\mathrm{End}(E) \approx \mathbb{Z}$. These are the multiplication-by-integers isogenies, and nothing else. This structure is too simple to be interesting in cryptographic applications and only appears in elliptic curves over infinite fields.

2. $\mathrm{End}(E)$ is a subset of an imaginary quadratic field, $\mathbb{Q}(\sqrt{\Delta}) = \{a + b\sqrt{\Delta} : a, b \in \mathbb{Q}\}$.

3. $\mathrm{End}(E)$ is a subset of a quaternion algebra, $B_{p,\infty} = \{a + bi + cj + dk : a, b, c, d \in \mathbb{Q}\}$, where $i^2 = -1, j^2 = -p, ij = -ji = k$.

The curves of case (2) are *ordinary*, and those of case (3) are *supersingular*. But with supersingular curves over $\mathbb{F}_p$, one can restrict to endomorphisms over $\mathbb{F}_p$, denoted $\mathrm{End}_{\mathbb{F}_p}(E)$, which turns out to have the structure of case (2). Table 1 summarizes this.

In case (3), $\mathrm{End}(E) \subset B_{p,\infty}$ is non-commutative, but it remains useful in cryptography for constructing protocols in other ways. This case corresponds to

Cuadro 1: The structure of the endomorphism rings of elliptic curves.

	Ordinary	Supersingular
$\mathrm{End}_{\mathbb{F}_p}(E)$	$\mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$	$\mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$
$\mathrm{End}(E)$	$\mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$	$\mathcal{O} \subset B_{p,\infty}$

applications such as SQIsign [1] and its variants. In what follows, we focus on case number (2), where $\mathrm{End}(E)$ and/or $\mathrm{End}_{\mathbb{F}_p}(E)$ are subsets of $\mathbb{Q}(\sqrt{\Delta})$, and are therefore commutative.

4. CSIDH group action

We are going to focus on supersingular curves and their endomorphism rings over $\mathbb{F}_p$, which are commutative.

Again, consider the set of isogenies $\mathrm{Hom}(E, E')$. For any isogeny $\varphi \in \mathrm{Hom}(E, E')$, if we multiply (on the right) by an endomorphism $\omega \in \mathrm{Hom}(E, E)$, we obtain $\varphi \circ \omega \in \mathrm{Hom}(E, E')$. This means that the group $\mathrm{Hom}(E, E')$ is closed under multiplication (on the right) by members of the ring $\mathrm{End}(E)$. That is, the set of isogenies $\mathrm{Hom}(E, E')$ is an *ideal* of the ring $\mathrm{End}(E)$.

Two ideals I, J of $\mathrm{End}(E)$ are said to be equivalent if there exists an $\alpha \in \mathbb{Q}(\sqrt{\Delta})$ where $I = J\alpha$. The set of distinct ideals under this definition of equivalence is called the *ideal class group*, and it is a multiplicative group. It is denoted as $\mathrm{cl}(\mathcal{O})$, where $\mathrm{End}_{\mathbb{F}_p}(E)$ is isomorphic to a certain order $\mathcal{O}$ in the imaginary quadratic field $\mathbb{Q}(\sqrt{\Delta})$. This means that the principal ideals $\langle\alpha\rangle$ form the trivial class of $\mathrm{cl}(\mathcal{O})$. For a more detailed study of ideal class groups, see the book by Cox [14, Sect. 7].

Ejemplo 3. *For example, consider the ideal* $\mathfrak{a} = \langle \pi - 1, 3\rangle$. *Here* π *is the Frobenius map that sends a point* $P = (x, y)$ *to* $\pi(P) = (x^p, y^p)$. *The ideal has two generators, so it is not a principal ideal, and therefore it is not the identity element of cl*$(\mathcal{O})$.

The inverse of $\mathfrak{a}$ *is* $\mathfrak{a}^{-1} = \langle \pi + 1, 3\rangle$.

Ejercicio 4.1 Let E be a supersingular curve, and $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$. Let ℓ_i be a prime where $\ell_i \mid p + 1$. Show that the ideal $\langle \pi + 1, \ell_i\rangle \in \mathcal{O}$ and the ideal $\langle \pi - 1, \ell_i\rangle \in \mathcal{O}$ are inverses.

Hint: Show that $\langle \pi + 1, \ell_i\rangle \cdot \langle \pi - 1, \ell_i\rangle = \langle \ell_i\rangle$. Recall that $\pi^2 = -p$.

Thus, with these structures, we can define a group action where the ideal class group acts on the set of elliptic curves with endomorphism ring isomorphic to $\mathcal{O}$. This set of curves can be defined alternatively as elliptic curves with *complex multiplication* (CM) by $\mathcal{O}$.

Taking X as the set of supersingular elliptic curves over $\mathbb{F}_p$ that have endomorphism group isomorphic to a particular imaginary quadratic order,

$$\mathcal{E}\ell\ell_p(\mathcal{O}) := \{E \text{ supersingular } : \mathrm{End}_{\mathbb{F}_p}(E) \cong \mathcal{O}\}$$

and G as the ideal class group of $\mathcal{O}$, $\mathrm{cl}(\mathcal{O})$, we can construct a group action

$$\star : \mathrm{cl}(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) \to \mathcal{E}\ell\ell_p(\mathcal{O}).$$

We use $[\mathfrak{a}]$ to denote the ideal class of the ideal $\mathfrak{a}$. Then, $[\mathfrak{a}]$ acts on any elliptic curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ through an isogeny

$$\varphi_{\mathfrak{a}} : E \to E/[\mathfrak{a}],$$

where $\ker(\varphi_{\mathfrak{a}}) = \cap_{\alpha \in \mathfrak{a}} \ker(\alpha)$ and $E/[\mathfrak{a}]$ is the codomain of $\varphi_{\mathfrak{a}}$.

Ejemplo 4. *From the previous example,* $\mathfrak{a} = \langle \pi - 1, 3 \rangle$. *This ideal corresponds to the isogeny with kernel* $\ker \varphi_{\mathfrak{a}} = \{P \in E(\bar{\mathbb{F}}_p) : [3]P = 0_E \text{ and } \pi(P) - P = 0_E\}$.

This action gives us a new way to take steps in the isogeny graph. For example, starting from a curve $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$, we can select an element $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$, and compute

$$E' = [\mathfrak{a}] \star E.$$

This step corresponds to an isogeny of degree $n(\mathfrak{a})$, where $n(\cdot)$ refers to the norm of the ideal. As we have already seen, computing high-degree isogenies is not easy. So we want to divide $[\mathfrak{a}]$ into smaller steps. Let $\mathfrak{l}_1, \dots \mathfrak{l}_k$ be a set of (prime) elements of $\mathrm{cl}(\mathcal{O})$ of low norm that generate a large part of $\mathrm{cl}(\mathcal{O})$. Then, instead of selecting $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$, we can select a vector $[e_1, \dots e_k] \in [-m, m]^k$ and compute

$$E' = \prod_{i=1}^{k} \mathfrak{l}_i^{e_i} \star E,$$

where m is selected according to the size of $\mathrm{cl}(\mathcal{O})$. Using this construction, the group action becomes a restricted effective group action (REGA).

In this way, we ensure that the corresponding isogeny can be constructed from isogenies with degrees $n(\mathfrak{l}_i)$. In Figure 3, an example of an isogeny graph with three choices of $\mathfrak{l}_i$ can be seen.

4.1. CSIDH

This group action was used in [12] to construct *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH). This protocol is very similar to Protocol 2. CSIDH remains one of the most popular isogeny protocols today.

Protocolo 3. *First, the public parameters are defined: a prime* $p \equiv 3 \bmod 4$, *the set* $\mathcal{E}\ell\ell_p(\mathcal{O})$, *an initial curve* $E_0 \in \mathcal{E}\ell\ell_p(\mathcal{O})$, *the elements* $\mathfrak{l}_1, \dots \mathfrak{l}_k$, *and the integer* m.

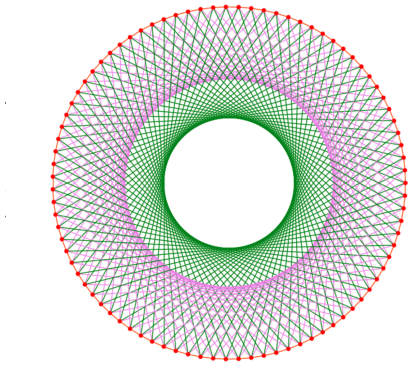

Figura 3: Isogeny graph over $\mathbb{F}_p$, with $p = 4 \cdot 5 \cdot 13 \cdot 61 - 1$. The orange lines correspond to degree-5 isogenies, the green lines to degree-13 isogenies, and the purple lines to degree-61 isogenies. The image was made by Francisco Rodríguez-Henríquez [30].

Now Alicia and Beto choose their secret vectors $(a_1, \dots a_k), (b_1, \dots b_k) \in [-m, m]^k$.

Alicia publishes the curve

$$E_A := \prod_{i=1}^{k} \mathfrak{l}_i^{a_i} \star E_0$$

and Beto the curve

$$E_B := \prod_{i=1}^{k} \mathfrak{l}_i^{b_i} \star E_0.$$

Since the action is commutative, each can compute the secret curve

$$\prod_{i=1}^{k} \mathfrak{l}_i^{b_i} \star E_A = \prod_{i=1}^{k} \mathfrak{l}_i^{a_i} \star E_B =: E_{AB}.$$

This next exercise, which begins with a simplified example of CSIDH, was written by Lorenz Panny [27].

```
p = 38798759
E0 = EllipticCurve(GF(p), [1,0])
factor(p+1)
ls = [3,5,7,11,13,17,19]

def point_of_order(E, l):
    assert l in ls
    while True:
        P = E.random_point()
        P *= (p+1)//l
        if P:
```

```
            return P

point_of_order(E0, 7)
```

Ejercicio 4.2 Using the given code, write a Sage function that takes as input an elliptic curve and a degree ℓ, computes a degree-ℓ isogeny using the .isogeny() function, and returns the codomain of this isogeny.

Ejercicio 4.3 Use the preceding example and your function from the preceding exercise to verify in Sage that the isogeny group action is commutative.

Write functions that implement Protocol 2 using the isogeny group action. Your functions should include key_gen() and compute_shared_secret().

Ejercicio 4.4 Give a proof that CSIDH is a REGA.

Ejercicio 4.5 The CRS protocol [13,31], which was the predecessor of CSIDH, also used an isogeny group action, but where the set X was composed of ordinary elliptic curves. Why is it more useful to use supersingular curves?

Hint: For supersingular curves, $\#E(\mathbb{F}_p) = p + 1$.

4.1.1. Quadratic twist

The *quadratic twist* of a curve is very important in CSIDH. Let $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ be a fixed curve such that $E : y^2 = f(x)$. Then, its *quadratic twist* is isomorphic to the curve $E^t : d \cdot y^2 = f(x)$, where d is any non-quadratic residue modulo p. In regards to the group action, we have that in general

$$(g \star E)^t = g^{-1} \star E^t,$$

where the exponent t is used to denote the quadratic twist. In general, the $\mathbb{F}_p$-rational curves used in CSIDH will be isomorphic to their twists over $\mathbb{F}_{p^2}$, but not over $\mathbb{F}_p$. The initial curve, however, is specially selected so that it *is* $\mathbb{F}_p$-isomorphic to its twist. By denoting this curve as E_0 (which has j-invariant equal to 1728 and where $p \equiv 3 \mod 4$), it is obtained that

$$(g \star E_0)^t = g^{-1} \star E_0^t = g^{-1} \star E_0.$$

This is the reason why computing twists turns out to be efficient in the CSIDH group action, and allows computing secret vectors with negative coefficients efficiently.

4.2. CSI-FiSh

Despite using supersingular curves and ideal classes of small norms to compute the group action, it remains a very slow computation in cryptographic

terms. Therefore, the authors of the digital signature CSI-FiSh [7] opted to precompute a lattice of relations of the small primes in order to compute the group action.

4.2.1. The lattice of relations

The ideal class group is cyclic, so with a generator $\mathfrak{g}$ of order N, all members of the group can be described in the form $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$. However, it is not clear how to compute elements $\mathfrak{g}^z$ with high norm.

Ejercicio 4.6 Why might it be better to compute elements of the ideal class group of the form $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$, where $\mathfrak{g}$ is a generator, instead of using the form denoted in CSIDH?

The idea is to use the low-norm primes $\mathfrak{l}_i$. We define a lattice

$$L := \{[x_1, \dots x_n] \in \mathbb{Z} : \prod_{i=1}^{n} \mathfrak{l}_i^{x_i} = (1)\}.$$

This lattice is called the *relation lattice*, and it gives insight into the group structure that was previously lacking in CSIDH. It can be seen that the ideal class group is $\mathrm{cl}(\mathcal{O}) \simeq \mathbb{Z}^n/L$. This means that for an element $\mathbf{z} = [z_1, \dots z_n]$ (which does not need to have small coefficients), one can simplify it by finding the point in L closest to $\mathbf{z}$, say $\mathbf{x}$, and instead use $\mathbf{z} - \mathbf{x}$, which has a smaller norm.

Computing the lattice can be done as a precomputation, but it is very costly. It required an estimated 43 core-years to compute the relation lattice for CSIDH/CSI-FiSh with a prime p where $\log_2 p = 512$ (today these parameters are thought to provide a quantum security level below 128 bits).

This algorithm works for all $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$. Thus, the group action is no longer restricted according to the definition of REGA (Definition 2)!

4.2.2. The scheme

CSI-FiSh derives from Stolbunov's ideas to obtain a signature scheme for the CSIDH group action. The scheme starts from a standard proof of knowledge.

Protocolo 4. *We begin by fixing the initial curve $E_0 : y^2 = x^3 + x$. Next, the prover chooses their secret, $[\mathfrak{a}] \in cl(\mathcal{O})$, and computes their public key $E_A = [\mathfrak{a}]E_0$. To prove knowledge of their secret key to a verifier, the prover first sends a random commitment $E_1 = [\mathfrak{b}]E_0$. The verifier responds with a challenge $c \in \{0, 1\}$. If $c = 0$, then the prover sends $r = [\mathfrak{b}]$, and the verifier checks that $E_1 = [r]E_0$. Otherwise, when $c = 1$, the prover sends $r = [\mathfrak{b}\mathfrak{a}^{-1}]$, and the verifier checks that $E_1 = [r]E_A$.*

The isogeny diagram for this scheme is sketched in Figure 4.

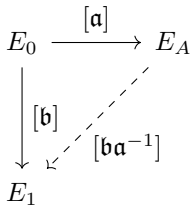

Figura 4: Diagram of the CSI-FiSh protocol.

Ejercicio 4.7 The CSI-FiSh protocol described here has a *soundness error* equal to 1/2. Why?

It is possible to lower the soundness error by using multiple secret and public keys. That is, the public key becomes a list of curves $(E_1, \dots E_n)$, and the secret key a corresponding list of ideals $([\mathfrak{a}_1], \dots [\mathfrak{a}_n])$. How would you adapt the protocol for this change? What is the new soundness?

Ejercicio 4.8 Why is it necessary for the group action in this scheme to be an EGA, and not a REGA?

Hint: Consider when the challenge is $c = 1$.

4.2.3. SCALLOP

The authors of SCALLOP [18] change the set of curves X to obtain different ideal class groups, where the relation lattice can be calculated more easily. This allows the calculation of groups for higher security parameters. While this study improved security choices, it remains too slow for cryptographic use and requires a very specific parameter balance.

5. PEGASIS group action

PEGASIS [15] offers yet another way to compute the isogeny group action, this time using dimension-4 isogenies. A brief description of PEGASIS is presented following the notes by Eriksen [19].

In 2022, a series of attacks [11, 25, 29] were published demonstrating how to attack the SIDH protocol [22] using isogenies of dimension 2, 4, or 8. The idea is to embed the target isogeny into a higher-dimensional one, where the action of the latter on a specific torsion subgroup is known. This information had been part of the public information in SIDH. Protocols like CSIDH do not publish this type of information, so they are not vulnerable to this attack.

We will not describe Kani's attack in detail, as it is beyond the scope of these notes. For more details, consult the original attack papers [11, 25, 29].

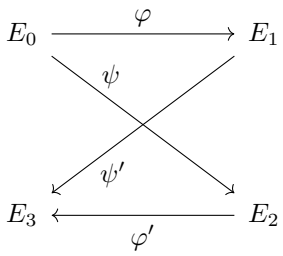

Figura 5: Kani's square.

Very simply, to construct a dimension-2 isogeny $\Phi : E_0 \times E_3 \to E_1 \times E_2$, we require a construction like the one in Figure 5, where $\psi' \circ \varphi = \varphi' \circ \psi$.

The degree of Φ is then $\deg \Phi = \deg \varphi + \deg \psi' = \deg \varphi' + \deg \psi$. With information about ψ' and Φ (with particular degrees), one can evaluate φ (of any degree) efficiently.

Now suppose we have an ideal class $[\mathfrak{a}]$ that we wish to evaluate. In Clapoti [26], the authors show that by finding two ideals equivalent to $\mathfrak{a}$, $[\mathfrak{a}] = [\mathfrak{b}] = [\mathfrak{c}]$, such that $n(\mathfrak{b}) + n(\mathfrak{c}) = 2^e$, then the action of $[\mathfrak{a}]$ can be computed efficiently using Kani's attack. However, it is not obvious how to find these ideals, and removing the degree requirement would imply using dimension-8 isogenies, which are not practical (as of today).

In KLaPoTi [28], the authors find the ideals $\mathfrak{b}, \mathfrak{c}$ using a known algorithm, KLPT [23]. However, the value of e is very large, making all computations more expensive. In PEGASIS, the authors find a middle ground where they are able to use dimension-4 isogenies, but with a value of e that is not too large. Later, in work subsequent to PEGASIS, qt-PEGASIS [16] uses a different approach to calculate the equivalent ideals that takes advantage of solving norm equations in a quaternion algebra. These changes ultimately provide a speedup in computing the group action, making it the fastest implementation of an isogeny group action to date.

Clapoti, KLaPoTi, and (qt)PEGASIS are unrestricted group actions and apply to a wider set of curves, not just that of CSIDH. Table 2 shows a comparison between several group actions.

Ejercicio 5.1 Use PEGASIS to implement the CSI-FiSh digital signature. Program the classic CSI-FiSh sigma protocol, as well as the protocol described in Exercise 8.

The PEGASIS code can be found at https://github.com/KULeuven-COSIC/qt-pegasis. You can use the following starter code, written by Riccardo Invernessi, to begin.

```
from qt_pegasis import qtPegasis

lvl = '500'
```

Cuadro 2: Comparison of execution times (in seconds) against different prime sizes for various schemes. Note: these are only approximations, as the schemes were executed on different hardware. The values in the table are taken from [15, Table 3] and [16, Table 1].

Scheme	**Paper**	**Language**	**Approx. prime size (bits)**	
			500	**2000**
SCALLOP	[18]	C++	35	750
KLaPoTi	[28]	Sage	207	—
		Rust	1.95	—
PEGASIS	[15]	Sage	1.53	21.3
qt-PEGASIS	[16]	Sage	0.85	9.69

```
EGA = qtPegasis(lvl)

# secret key
frak_a = EGA.sample_sage_ideal()

# public key
EA = EGA.sage_action(frak_a)
```

6. Open problems

Below are two active areas in the literature that may be more accessible to new researchers in this field.

6.1. Advanced protocols

The most attractive part of a commutative group action is its similarity to the Diffie-Hellman protocol. Diffie-Hellman has been used for key exchange, but also for many other advanced protocols. Therefore, having a post-quantum analogue leaves the door open to creating post-quantum versions of many other protocols. There are already several examples, such as threshold signatures [4,20] or password-authenticated key exchange (PAKE) [2], among others.

This raises the open question: what other types of schemes can be made post-quantum using a group action like CSIDH or qt-PEGASIS? There is also much to be done regarding the development or improvement of existing advanced protocols. This could include programming algorithms, searching for algorithmic improvements, or investigating security claims.

For example, in [2] two PAKEs based on commutative group actions are presented. The constructions are rather slow, especially when using CSIDH. For this reason, the authors did not include implementations of the protocols in the paper. Since the theory of isogeny group actions has developed further, how

could this protocol be improved in terms of efficiency? Are there other changes that could be made to improve them?

6.2. Fault injection attacks

A *fault injection attack* occurs when the hardware executing a protocol is manipulated in the hope that the altered protocol reveals some secret information. This attack model possesses both a hardware and a theoretical aspect: how to induce a fault into the protocol and, once altered, how it reveals information about the secret. By designing attacks that employ standard faults, one can focus more on the theoretical aspect.

Below is a list of faults considered standard and whose implementation is reasonable [8]:

- *Memory manipulation.* Occurs when a byte of a variable in memory is altered. For example, by assigning a new random value to that byte. A stronger attack could include bit-flipping or setting values to zero.
- *Early loop abort.* Occurs when a loop in an algorithm is interrupted earlier than planned. The less precision required, the more reasonable the attack becomes.
- *Instruction skip.* Occurs when a specific line of code is skipped. Again, the higher the required precision, the more difficult it is in practice.

Therefore, the open question is: can secret information be recovered from a protocol using one of these faults?

6.2.1. State of the art

For CSIDH, several papers on fault injection attacks have already been published. These fault attacks target dummy operations and obtain an oracle by observing whether the output changes after a fault; that is, whether the fault affected a necessary operation or a dummy one [9, 10, 24]. Generally, these fault attacks employ a fault model that is easy to achieve in practice, as high precision is not required, and can include instruction skips, memory faults, or loop aborts. On the other hand, these attacks require many samples of faulty executions, typically on the order of hundreds at a minimum. Similarly, a fault attack on point sampling in CSIDH presents similar requirements regarding the fault model, but demands an even larger amount of faulty samples [5]. See Table 3.

So far, no attacks have been published against the other isogeny group actions. It is unclear whether the preceding attacks can be adapted for other actions, as these attacks usually focus on specific implementations of each protocol, making adaptation non-trivial. Entirely new attacks can also be explored.

Cuadro 3: The table describes the number of faults required to implement each attack (when implemented targeting $\lambda = 128$ bits of security). These values are independent of the success probability for the different fault injection models. All faults are first-order. The final column details the new effective secret space size after injected faults and post-processing.

Protocol	Paper	# of faults	Effective secret space
CSIDH	[9, Attack 3]	178	$x \in [1, 2^{67}]$
	[5, Sec. 5.1]	512	1
	[24, Table 1]	263	1

6.2.2. Disorientation fault

The attack in [5] fault-injects the "orientation" of one of the steps in computing the shared curve in CSIDH. Here, orientation refers to the sign of an exponent. For example, by faulting the first step of the computation, they obtain the curve

$$E_1 := \mathfrak{l}_1^{-e_1} \star \prod_{i=2}^{k} \mathfrak{l}_i^{e_i} \star E_0,$$

instead of the usual $E_{AB} = \prod_{i=1}^{k} \mathfrak{l}_i^{e_i} \star E_0$. Between E_1 and E_{AB} there exists an isogeny $\varphi : E_1 \to E_{AB}$ of degree $\deg \varphi = (\mathfrak{l}_1^{e_1})^2$. Thus, upon obtaining E_1 and E_{AB}, an adversary can search for a **short** isogeny between the two to discover information about the secret vector.

This fault can be obtained with an instruction skip or with a memory manipulation. See [5] for details on the attack.

6.2.3. Faults on torsion point masking

Due to Kani's attack, today's protocols cannot publish information about the action of a secret isogeny on a torsion group. That is, if $\varphi : E_0 \to E_1$ is an isogeny, having $\langle P, Q \rangle = E_0[N]$ and $\varphi(P), \varphi(Q)$, for $N \geq 4 \cdot \deg \varphi$, is sufficient to reconstruct φ using a higher-dimensional isogeny. Because of this, protocols like the PRISM digital signature [6], instead of publishing $\varphi(P), \varphi(Q)$ in the signature, publish

$$\begin{pmatrix} P' \\ Q' \end{pmatrix} := \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{pmatrix} [m_{11}]\varphi(P) + [m_{12}]\varphi(Q) \\ [m_{21}]\varphi(P) + [m_{22}]\varphi(Q) \end{pmatrix},$$

where the $m_{ij} \in \mathbb{Z}$ are part of the secret. This way of hiding torsion information is called *torsion masking*.

The idea of [21] consists of faulting one of the coefficients m_{ij} with a memory manipulation. Suppose m_{11} is faulted to m'_{11}. With this change, the protocol

publishes

$$\begin{pmatrix} P'' \\ Q' \end{pmatrix} := \begin{bmatrix} m'_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{pmatrix} \varphi(P) \\ \varphi(Q) \end{pmatrix} = \begin{pmatrix} [m'_{11}]\varphi(P) + [m_{12}]\varphi(Q) \\ [m_{21}]\varphi(P) + [m_{22}]\varphi(Q) \end{pmatrix}.$$

Then, one can compute $P' - P'' = [m_{11} - m'_{11}]\varphi(P)$. In general, $m_{11} - m'_{11}$ will have very few possibilities, allowing one to obtain the value of $\varphi(P)$. Repeating the attack, but this time faulting m_{21} or m_{22}, yields the value of $\varphi(Q)$.

7. Other resources

For those interested in continuing their study of isogenies, the *Isogeny Club* [32] organizes biweekly talks on Zoom on topics related to isogenies. All talks are shared on YouTube.

8. Exercises

Ejercicio 1.1 Write Sage code that implements the Diffie-Hellman protocol where the group used is the group of points from the elliptic curve $E : y = f(x)$, where $f(x) = x^3 + 486662x^2 + x$ over the finite field $\mathbb{F}_p$ with $p = 2^{255} - 19$. Use $G = (9, y)$ as a generator.

You can use the following starter code to begin.

Ejercicio 1.2 What is the relationship between Problem 1 and Problem 2? Are they equivalent?

Ejercicio 1.3 Let E be an elliptic curve, $P, Q \in E[N]$. A *pairing* is a bilinear map that can be defined on E (or other groups) of the form $e : E \times E \to \mu_N$ where $e(P, [r]Q) = e([r]P, Q) = e(P, Q)^r$.

Use a pairing to give an example in which Problem 3 is easy but Problem 2 remains hard.

Is it important for Protocol 1 that Problem 3 is hard?

Ejercicio 1.4 Consider the Diffie-Hellman protocol using points on an elliptic curve (ECDH), as in the preceding exercise. Shor's algorithm would allow calculating the *period* of a cyclic function. For example, for the function $f(a, b) := [a]P + [b]Q$, Shor's algorithm can compute r_a, r_b where $f(a + r_a, b + r_b) = f(a, b)$.

Using Shor's algorithm, describe an attack against ECDH.

Ejercicio 3.1 According to Tate's theorem, two curves are isogenous over $\mathbb{F}_q$ if and only if they have the same number of points over $\mathbb{F}_q$.

Use Sage to verify Tate's theorem.

a. Select an elliptic curve over the field $\mathbb{F}_q$.

b. Construct an isogeny.

c. Count the number of points on the domain and codomain of the isogeny over $\mathbb{F}_q$.

Ejercicio 3.2 Give a proof that $\mathrm{End}(E)$ is a ring.

Ejercicio 4.1 Let E be a supersingular curve, and $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$. Let ℓ_i be a prime where $\ell_i \mid p+1$. Show that the ideal $\langle \pi + 1, \ell_i \rangle \in \mathcal{O}$ and the ideal $\langle \pi - 1, \ell_i \rangle \in \mathcal{O}$ are inverses.

Hint: Show that $\langle \pi + 1, \ell_i \rangle \cdot \langle \pi - 1, \ell_i \rangle = \langle \ell_i \rangle$. Recall that $\pi^2 = -p$.

Ejercicio 4.2 Using the given code, write a Sage function that takes as input an elliptic curve and a degree ℓ, computes a degree-ℓ isogeny using the .isogeny() function, and returns the codomain of this isogeny.

Ejercicio 4.3 Use the preceding example and your function from the preceding exercise to verify in Sage that the isogeny group action is commutative.

Write functions that implement Protocol 2 using the isogeny group action. Your functions should include key_gen() and compute_shared_secret().

Ejercicio 4.4 Give a proof that CSIDH is a REGA.

Ejercicio 4.5 The CRS protocol [13, 31], which was the predecessor of CSIDH, also used an isogeny group action, but where the set X was composed of ordinary elliptic curves. Why is it more useful to use supersingular curves?

Hint: For supersingular curves, $\#E(\mathbb{F}_p) = p + 1$.

Ejercicio 4.6 Why might it be better to compute elements of the ideal class group of the form $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$, where $\mathfrak{g}$ is a generator, instead of using the form denoted in CSIDH?

Ejercicio 4.7 The CSI-FiSh protocol described here has a *soundness error* equal to 1/2. Why?

It is possible to lower the soundness error by using multiple secret and public keys. That is, the public key becomes a list of curves $(E_1, \dots E_n)$, and the secret key a corresponding list of ideals $([\mathfrak{a}_1], \dots [\mathfrak{a}_n])$. How would you adapt the protocol for this change? What is the new soundness?

Ejercicio 4.8 Why is it necessary for the group action in this scheme to be an EGA, and not a REGA?

Hint: Consider when the challenge is $c = 1$.

Ejercicio 5.1 Use PEGASIS to implement the CSI-FiSh digital signature. Program the classic CSI-FiSh sigma protocol, as well as the protocol described in Exercise 8.

The PEGASIS code can be found at https://github.com/KULeuven-COSIC/qt-pegasis. You can use the following starter code, written by Riccardo Invernessi, to begin.

```
# for Exercise 1.1
p = 2^255-19
E = EllipticCurve(GF(p), [0,486662,0,1,0])

y = Mod(f(9), p).sqrt()
G = E(9, y)
```

```
# for Exercises 4.2, 4.3
p = 38798759
E0 = EllipticCurve(GF(p), [1,0])
factor(p+1)
ls = [3,5,7,11,13,17,19]

def point_of_order(E, l):
    assert l in ls
    while True:
        P = E.random_point()
        P *= (p+1)//l
        if P:
            return P

point_of_order(E0, 7)
```

```
# for Exercise 5.1
from qt_pegasis import qtPegasis

lvl = '500'
EGA = qtPegasis(lvl)

# secret key
frak_a = EGA.sample_sage_ideal()

```

```
10 # public key
11 EA = EGA.sage_action(frak_a)
```

Referencias

[1] Marius A. Aardal, Gora Adj, F. Austin, et al. SQIsign 2.0: Algorithm specifications and supporting documentation. Technical report, National Institute of Standards and Technology, 2025. NIST Post-Quantum Cryptography Standardization Contest, Round 2. Available at https://sqisign.org/spec/sqisign-20250205.pdf.

[2] Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022*, volume 13508 of *Lecture Notes in Computer Science*, pages 699–728. Springer, 2022.

[3] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 411–439. Springer, December 2020.

[4] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. Csi-shark: Csi-fish with sharing-friendly keys. In *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings*, volume 13915 of *Lecture Notes in Computer Science*, pages 471–502. Springer, 2023.

[5] Gustavo Banegas, Juliane Krämer, Tanja Lange, Michael Meyer, Lorenz Panny, Krijn Reijnders, Jana Sotáková, and Monika Trimoska. Disorientation faults in CSIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 310–342. Springer, 2023.

[6] Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: simple and compact identification and signatures from large prime degree isogenies. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part III*, Lecture Notes in Computer Science.

[7] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. *Advances in Cryptology – ASIACRYPT 2019*, pages 227–247, 2019.

[8] Jakub Breier and Xiaolu Hou. How practical are fault injection attacks, really? *IEEE Access*, 10:113122–113130, 2022.

[9] Fabio Campos, Matthias J. Kannwischer, Michael Meyer, Hiroshi Onuki, and Marc Stöttinger. Trouble at the CSIDH: protecting CSIDH with dummy-operations against fault injection attacks. In *17th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2020, Milan, Italy, September 13, 2020*, pages 57–65. IEEE, 2020.

[10] Fabio Campos, Juliane Krämer, and Marcel Müller. Safe-error attacks on SIKE and CSIDH. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering - 11th International Conference, SPACE 2021, Kolkata, India, December 10-13, 2021, Proceedings*, volume 13162 of *Lecture Notes in Computer Science*, pages 104–125. Springer, 2021.

[11] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.

[12] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

[13] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006.

[14] David A. Cox. *Primes of the form x^2+ny^2: Fermat, Class Field Theory, and Complex Multiplication*. AMS Chelsea Publishing/American Mathematical Society, third edition, 2022.

[15] Pierrick Dartois, Jonathan Komada Eriksen, Tako Boris Fouotsa, Arthur Herlédan Le Merdy, Riccardo Invernizzi, Damien Robert, Ryan Rueger, Frederik Vercauteren, and Benjamin Wesolowski. PEGASIS: Practical effective class group action using 4-dimensional isogenies. In *Advances in Cryptology – CRYPTO 2025*. Springer, 2025.

[16] Pierrick Dartois, Jonathan Komada Eriksen, Riccardo Invernizzi, and Frederik Vercauteren. qt-Pegasis: Simpler and faster effective class group actions. In *Advances in Cryptology – EUROCRYPT 2026*. Springer, 2026.

[17] Luca De Feo. Group and groupoid actions – as a foundation for post-quantum cryptography. Presentation at the Summer School on Real-World Crypto and Privacy, Dubrovnik, Croatia, June 2025. https://summerschool-croatia.cs.ru.nl/2025/slides/2025-06-30-croatia.pdf.

[18] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: scaling the CSI-FiSh. In Carmit Hazay and Martijn Stam, editors, *Public-Key Cryptography – PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography*, volume 13941 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2023.

[19] Jonathan Komada Eriksen. The road to PEGASIS. Presentation slides, 2025. Accessed: 2026.

[20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography*, volume 12111 of *Lecture Notes in Computer Science*, pages 187–212. Springer, 2020.

[21] Valerie Gilchrist, Yi-Fu Lai, and Michael Meyer. Fault injection attacks on torsion masking. Submitted for publication, 2026.

[22] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[23] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem.

[24] Jason T. LeGrow and Aaron Hutchinson. (short paper) analysis of a strong fault attack on static/ephemeral CSIDH. In Toru Nakanishi and Ryo Nojima, editors, *Advances in Information and Computer Security - 16th International Workshop on Security, IWSEC 2021, Virtual Event, September 8-10, 2021, Proceedings*, volume 12835 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 2021.

[25] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471. Springer, 2023.

[26] Aurel Page and Damien Robert. Introducing Clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Paper 2023/1294, 2023.

[27] Lorenz Panny. Some exercises on isogeny-based cryptography. https://yx7.cc/docs/isog/isog_taipei_exercises.pdf#page=3.08, July 2022. Post-Quantum Crypto Mini-School 2022, Taipei, Taiwan.

[28] Lorenz Panny, Christophe Petit, and Miha Stopar. KLaPoTi: An asymptotically efficient isogeny group action from 2-dimensional isogenies. *IACR Communications in Cryptology*, 2(3):470–494, 2025.

[29] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503. Springer, 2023.

[30] Francisco Rodríguez-Henríquez. A tale of two isogeny-based protocols: State-of-the-art review of key-exchange and signature isogeny-based protocols, June 2024. EPFL, Lausanne, Switzerland.

[31] Alexander Rostovtsev and Anton Stolbunov. PUBLIC-KEY CRYPTOSYSTEM BASED ON ISOGENIES. Cryptology ePrint Archive, Paper 2006/145, 2006.

[32] Maria Corte-Real Santos and Jonathan Komada Eriksen. Isogeny club. https://isogeny.club/.

[33] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[34] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes rendus hebdomadaires des séances de l'Académie des sciences, Série A*, 273:238–241, 7 1971.