Santiago 2026 CIMPA School: Isogenias como acción de grupo

Valerie Gilchrist

Royal Military Academy of Belgium

gilchrist.valerie@gmail.com

julio 2026

En lo que sigue damos una breve introducción al tema de acciones de grupo de isogenias. Varios de los ejemplos y ejercicios incluidos en estas notas necesitan el uso de SageMath.

1. Logaritmo discreto

La criptografía asimétrica requiere el uso de *funciones asimétricas*. Estas son funciones matemáticas que son fáciles y eficientes de calcular, pero para las cuales su inversión no es razonable. De los primeros ejemplos de funciones asimétricas es el *logaritmo discreto*.

Problema 1 (Logaritmo discreto)**.** *Sea $G = \langle g \rangle$ un grupo cíclico de orden N. Sea $a \in \mathbb{Z}_N$.*

Dado (g, g^a), calcula a.

Toma $G = \mathbb{Z}_p$ por p primo y grande. En este caso, Problema 1 es pensado ser difícil para computadoras clásicas, mientras que calcular $(g, a) \mapsto g^a$ se puede hacer muy rápidamente. Por eso Diffie y Hellman en el año 1976 usaron logaritmos discretos para construir un intercambio de claves[1].

Protocolo 1 (Diffie-Hellman)**.** *Selecciona un grupo conmutativo $G = \langle g \rangle$.*

Alicia y Beto comienzan por escoger enteros secretos $a, b \in \mathbb{Z}$. Calculan y publican sus llaves públicas g^a, g^b respectivamente. Con estas llaves públicas, y sus enteros secretos, los dos pueden calcular el objeto secreto $(g^a)^b = (g^b)^a$.

El siguiente ejercicio fue escrito por Lorenz Panny [27].

[1] Un protocolo similar fue desarrollado dentro de la GCHQ del Reino Unido en 1969, pero no la publicaron sino hasta 1977.

Ejercicio 1.1 Escriba código de Sage que implemente el protocolo Diffie-Hellman donde el grupo utilizado sea el grupo de puntos sobre la curva elíptica $E : y = f(x)$, donde $f(x) = x^3 + 486662x^2 + x$ sobre el campo finito $\mathbb{F}_p$ con $p = 2^{255} - 19$. Usa $G = (9, y)$ como generador.

Puedes usar el siguiente código de base para empezar.

```
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)
```

Para que el protocolo sea seguro, es necesario que el problema de logaritmos discretos (Problema 1) sea difícil. Pero además de esto, también necesitamos que dado (g, g^a, g^b), no se puede aprender nada del secreto g^{ab}. Este problema se llama el *problema Diffie-Hellman computacional*.

Problema 2 (Diffie-Hellman computacional)**.** *Sea $G = \langle g \rangle$ un grupo cíclico de orden N. Sea $a, b \in \mathbb{Z}_N$.*

Dado (g, g^a, g^b), calcula g^{ab}.

También existe el menos fuerte *problema Diffie-Hellman decisional*.

Problema 3 (Diffie-Hellman decisional)**.** *Sea $G = \langle g \rangle$ un grupo cíclico de orden N. Sea $a, b, c \in \mathbb{Z}_N$.*

Dado (g, g^a, g^b, g^c), decide si $c = ab$.

Ejercicio 1.2 Cuál es la relación entre Problema 1 y Problema 2? Son equivalentes?

Ejercicio 1.3 Sea E una curva elíptica, $P, Q \in E[N]$. Un *emparejamiento* (pairing) es un mapa bilinear que se puede definir en E (o otros grupos) de la forma $e : E \times E \to \mu_N$ donde $e(P, [r]Q) = e([r]P, Q) = e(P, Q)^r$.

Usa un emparejamiento para dar un ejemplo en que Problema 3 es fácil pero Problema 2 sigue siendo difícil.

Es importante por Protocolo 1 que Problema 3 sea difícil?

Diffie-Hellman sigue siendo usado hoy, sobre todo con grupos de puntos de curvas elípticas. Estos grupos permiten hacer computaciones muy eficientes y requieren poca memoria. Por ejemplo, Diffie-Hellman con curvas elípticas es usado por Bitcoin, TLS, Whatsapp, Signal, y muchas otras aplicaciones. El problema que está acercándose cada día más viene con el desarrollo de las computadoras cuánticas. Con el algoritmo de Peter Shor [33], un adversario podría calcular eficientemente logaritmos discretos. Esto quiere decir que Diffie-Hellman, usando cualquier grupo, no sería seguro contra adversarios cuánticas.

Ejercicio 1.4 Considera el protocolo de Diffie-Hellman usando puntos de una curva elíptica (ECDH), como en el ejercicio precedente. El algoritmo de Shor permitiría calcular el *periodo* de una función cíclica. Por ejemplo, para la función $f(a,b) := [a]P + [b]Q$, el algoritmo de Shor puede calcular r_a, r_b donde $f(a + r_a, b + r_b) = f(a,b)$.

Usando el algoritmo de Shor, describe un ataque contra ECDH.

Ahora queremos explorar protocolos que podrían remplazar a Diffie-Hellman, pero que tengan seguridad contra adversarios cuánticas. Este tipo de criptografía se llama *criptografía postcuántica*.

2. Acción de grupo

Las acciones de grupo, anteriormente llamadas *espacios homogéneo* por Couveignes [13], nos darán una manera de generalizar la acción usada en Diffie-Hellman.

Definición 1 (Acción de grupo (group action))**.** *Sea G un grupo, y X un conjunto. Un mapa*

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

es una acción de grupo si :

1. *existe un elemento $e \in G$ donde $e \star x = x$ para todos los $x \in X$;*
2. *para cualesquiera $g, h \in G$, $g \star (h \star x) = gh \star x$.*

En el Diffie-Hellman original, se calcula g^a para un elemento de grupo $g \in G$, y un entero a. Esto es un ejemplo de una acción de grupo donde X también es un grupo. Aquí, el grupo de enteros multiplicativos está actuando sobre el grupo G a través de exponenciación. Entonces Diffie-Hellman puede ser descrito de forma más general usando acciones de grupo.

Protocolo 2 (Diffie-Hellman con acción de grupo)**.** *Selecciona una acción de grupo $(\star, G, X)$. Selecciona un elemento $x \in X$. Sean $(\star, G, X, x)$ los parámetros públicos.*

Ahora Alicia y Beto seleccionan sus elementos secretos del grupo $g_A, g_B \in G$. Cada uno calcula y publica $g_A \star x, g_B \star x$ respectivamente. Ahora Beto puede calcular $g_B \star (g_A \star x)$ y Alicia puede calcular $g_A \star (g_B \star x)$. Si G es conmutativo, entonces Alicia y Beto pueden llegar a calcular el mismo objeto $(g_B \cdot g_A) \star x = (g_A \cdot g_B) \star x$.

Desafortunadamente, las acciones de grupo donde X también es un grupo no son seguros contra el ataque de Shor. Por eso hace falta estudiar acciones de grupo en que X no es un grupo. Además de esto, todavía nos faltan algunas propiedades para que la acción sea útil en criptografía. Estas propiedades son descritas en la siguiente definición tomada de [3], donde lo llaman un *restricted effective group action* (REGA).

Definition 1 (Acción de grupo efectiva restringida (REGA))**.** *Una acción de grupo efectiva restringida es una acción de grupo* $\star : G \times X \to X$*, junto con algún conjunto generador de* $G = \langle g_1, \dots g_n \rangle$*, donde* G *es finito y* $n = poly(\log(|G|))$*, y donde existen algoritmos eficientes en tiempo polinomial para computar las siguientes tareas:*

(a) Prueba de pertenencia: decidir si una entrada dada representa un elemento de X*;*

(b) Representación única de conjuntos: para cualquier $x \in X$*, computar una representación canónica* $\hat{x}$ *de* x*;*

(c) Origen: computar el punto distinguido $x_0 \in X$ *conocido como el origen;*

(d) Acción de grupo: para cualquier $i \in [n], x \in X$*, computar* $(g_i \star x)$ *y* $(g_i^{-1} \star x)$*.*

La noción de restringir la acción de grupo se ve en la propiedad (d). Una acción de grupo efectiva (EGA) que no sea restringida puede evaluar $g \star x$ para cualquier g, no solamente los generadores g_i.

Todas estas propiedades nos aseguran que el Protocolo 2 se puede realizar eficientemente, pero falta una descripción de seguridad. Brevemente, como en Diffie-Hellman, podemos ver que desearíamos que fuera difícil para un adversario computar g_A teniendo acceso a $(x, g_A \star x)$. Por Couveignes [13], este problema se llama *vectorización* (o *vectorization*).

Problema 4 (Vectorización)**.** *Dada una acción de grupo* $(G, X, \star)$*, y elementos del conjunto* $x, g \star x$ *donde* $x \in X, g \in G$*, recupera* g*.*

En la literatura, a menudo se hace referencia a este problema como el *Problema Inverso de la Acción de Grupo* (GAIP, por sus siglas en inglés).

Sin embargo, un adversario aún podría atacar el sistema sin necesidad de recuperar las claves secretas particulares. A esta noción la llamamos *paralelización* (o *parallelization*), que recuerda al Diffie-Hellman computacional.

Problema 5 (Paralelización)**.** *Dada una acción de grupo* $(G, X, \star)$*, y elementos del conjunto* $x, g_1 \star x, g_2 \star x$ *donde* $x \in X, g_1, g_2 \in G$*, computa* $(g_1 \cdot g_2) \star x$*.*

Definimos una *acción de grupo criptográfica* como una REGA tal que los problemas de *vectorización* y *paralelización* son difíciles. De aquí en adelante, a menos que se indique específicamente lo contrario, se asume que la acción de grupo es criptográfica.

Las acciones de grupo criptográficas postcuánticas más estudiadas provienen de redes, códigos e isogenias. De estas, solo la acción de isogenias es conmutativa.

3. Isogenias

Exploraremos como construir acciones de grupo usando isogenias siguiendo el ejemplo de la presentación de De Feo [17].

3.1. Recuerdo de isogenias

Anteriormente en la escuela se ofreció una introducción a las isogenias, pero aquí recordamos algunas de las definiciones fundamentales. Para más detalles, véanse las notas de Amalia Pizarro Madariaga.

Isogenias son objetos geométricos que describen relaciones entre curvas elípticas.

Definición 2. *Sean E_0, E_1 dos curvas elípticas. Una* isogenia *de E_0 a E_1 es un mapa racional (y no constante)*

$$\phi : E_0 \to E_1$$

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

Se dice que dos curvas E_0, E_1 son isógenas *si existe una isogenia de E_0 a E_1.*

Ejemplo 1. *Toma por ejemplo multiplicación por un entero n.*

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

Los puntos sobre una curva elíptica forman un grupo y entonces son cerrados bajo multiplicación por un entero. Esto quiere decir que ϕ manda los puntos de una curva a la misma curva,

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

Este tipo de isogenia se llama un endomorfismo.

Ejemplo 2. *Un ejemplo de una isogenia que no sea un endomorfismo puede ser*

$$\varphi : E_0 \to E_1,$$
$$(x, y) \mapsto (\frac{x^2+1}{x}, \frac{x^2y - y}{x^2}).$$

En general, es difícil construir isogenias usando expresiones racionales. Por eso es muy común construirlas usando su *kernel*, denotado $\ker(\varphi)$. El kernel de $\varphi : E_0 \to E_1$ es $\ker \varphi = \{P \in E_0 : \varphi(P) = O_{E_1}\}$. El kernel es suficiente para definir la isogenia usando las formulas de Vélu [34]. En la mayoría de casos, el tamaño del kernel se llama el *grado* de la isogenia, denotado por $\deg(\varphi)$.

En el ejemplo precedente, la isogenia φ puede ser construida usando un punto P con orden 2. Esto quiere decir que $[2]P = O_{E_1}$. Entonces $\ker(\varphi) = \{O_{E_0}, P\}$ y $\deg(\varphi) = |\ker(\varphi)| = 2$.

Aquí, pudimos construir una isogenia muy fácilmente, mediante la selección de un punto P, y definiendo $\ker(\varphi) := \langle P \rangle$. Esta construcción también nos da información sobre la curva E_1 porque $E_1 \simeq E_0/\ker(\varphi)$ (sobre $\bar{\mathbb{F}}_q$). Para puntos de bajo orden, este método funciona bien. En el caso de isogenias de grados más altos, a veces es posible construirlas usando varias isogenias más chicas. Por ejemplo, si queremos construir una isogenia de grado ℓ^k, donde ℓ es un primo pequeño, lo podemos hacer con k isogenias de grado ℓ cada una. En la Figura 1,

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

Figura 1: Cadena de isogenias.

la composición de las isogenias $\varphi_k \circ \varphi_{k-1} \circ \cdots \circ \varphi_1$ permite construir una isogenia de E_0 a E_k de grado $\prod_{i=1}^{k} \deg(\varphi_i)$.

Hasta ahora hemos visto cómo construir isogenias de grados con factores pequeños. Pero este método no nos da control sobre la curva final, E_k. En general, dadas las curvas E_0, E_k, es extremamente difícil construir una isogenia $\phi : E_0 \to E_k$. Esta es la razón por la cual se puede construir criptografía usando isogenias. La idea principal es que alguien puede seleccionar una isogenia $\phi : E_0 \to E_k$ como su llave privada, y publicar la curvas E_0 y E_k.

3.2. Grafos de isogenias

Considera un grafo en que los vértices son curvas elípticas y las aristas son isogenias de un grado particular, ℓ. Eso quiere decir que dos vértices están conectados si y sólo si las curvas correspondiendo a los vértices son isógenas de grado ℓ.

Ejercicio 3.1 Según el teorema de Tate, dos curvas son isógenas sobre $\mathbb{F}_q$ si y solo si tienen el mismo número de puntos sobre $\mathbb{F}_q$.

Usa Sage para verificar el teorema de Tate.

a. Selecciona una curva elíptica sobre el campo $\mathbb{F}_q$.

b. Construye una isogenia.

c. Cuenta el número de puntos sobre el dominio y codominio de la isogenia sobre $\mathbb{F}_q$.

Considera el grafo de isogenias de grado ℓ sobre $\mathbb{F}_{p^2}$. Una de dos cosas puede ocurrir. La primera opción es que haya uno o varios subconjuntos no conectados. En este caso, cada subconjunto tiene un ciclo en que cada vértice es la raíz de un árbol. El ejemplo de esta estructura en la Figura 2 demuestra visualmente por qué se llama un *volcán*. Las curvas en este grafo se llaman *ordinarias*.

La segunda opción es mucho menos común. En este caso el grafo es finito, conectado, y $(\ell+1)$-regular. Las curvas en este grafo se llaman *supersingulares*. Este grafo tiene propriedades de mezcla muy buenas. Por eso es muy común usar curvas supersingulares en protocolos postcuanticos.

```
# para verificar si una curva es supersingular
E = EllipticCurve(GF(7), [0,0,0,1,0]) # y^2 = x^3 + x
E.is_supersingular()
E.is_ordinary()
```

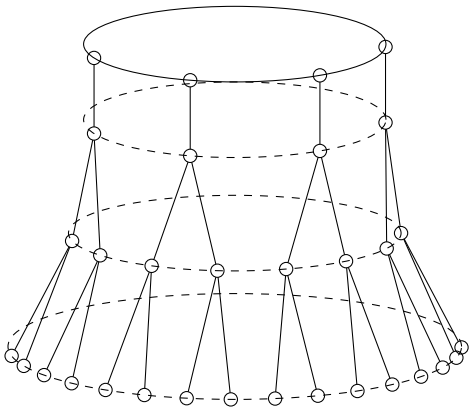

Figura 2: Volcán de 2-isogenias.

Otra razón por la cual curvas supersingulares son usadas mucho en criptografía es porque el número de puntos sobre $\mathbb{F}_p$ en estas curvas es fácil de calcular: $|E(\mathbb{F}_p)| = p+1$. Este control sobre el número de puntos puede ser atractivo para asegurar que puntos de bajos ordenes existen sobre $\mathbb{F}_p$.

```
# cuantos puntos hay sobre E en 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. Anillo de endomorfismos

Sean E, E' dos curvas elípticas. El functor $\mathrm{Hom}(E, E')$ describe los homomorfismos (isogenias) de E a E'. Sean $\varphi, \psi \in \mathrm{Hom}(E, E')$. Se puede ver de inmediato que $(\varphi + \psi)(P) = \varphi(P) + \psi(P)$ tambien es miembro de $\mathrm{Hom}(E, E')$. De hecho, para enteros a, b, $(a\varphi + b\psi)(P) = [a]\varphi(P) + [b]\psi(P) \in \mathrm{Hom}(E, E')$. Esto muestra que $\mathrm{Hom}(E, E')$ es un grupo aditivo.

Ahora considera $\mathrm{Hom}(E, E)$, que describe los endomorfismos de E. Si $\varphi, \psi \in \mathrm{Hom}(E, E)$, entonces la composición $\varphi \circ \psi(P) = \varphi(\psi(P)) \in \mathrm{Hom}(E, E)$. Composición sirve para definir una noción de multiplicación en $\mathrm{Hom}(E, E)$, que resulta ser suficiente para concluir que $\mathrm{Hom}(E, E) = \mathrm{End}(E)$ es un anillo.

Ejercicio 3.2 De una prueba que $\mathrm{End}(E)$ es un anillo.

Para cada curva elíptica E, el anillo de endomorfismos $\mathrm{End}(E)$ puede tener una de tres estructuras:

1. $\mathrm{End}(E) \approx \mathbb{Z}$. Estos son las isogenias de multiplicación por enteros, y nada mas. Esta estructura es demasiada simple para ser interesante en

aplicaciones criptográficas y solo aparece en curvas elípticas sobre campos infinitos.

2. $\mathrm{End}(E)$ es un subconjunto de un campo imaginario cuadrático, $\mathbb{Q}(\sqrt{\Delta}) = \{a + b\sqrt{\Delta} : a, b \in \mathbb{Q}\}$.

3. $\mathrm{End}(E)$ es un subconjunto de una álgebra de cuaterniones, $B_{p,\infty} = \{a + bi + cj + dk : a, b, c, d \in \mathbb{Q}\}$, donde $i^2 = -1, j^2 = -p, ij = -ji = k$.

Las curvas de caso (2) son *ordinarias*, y las de caso (3) son *supersingulares*. Pero con las curvas supersingulares sobre $\mathbb{F}_p$, se puede restringir a endomorfismos sobre $\mathbb{F}_p$, denotado $End_{\mathbb{F}_p}(E)$, que resulta tener la estructura de caso (2). Cuadro 1 demuestra esto.

Cuadro 1: La estructura de los anillos de endomorfismos de curvas elípticas.

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

En el caso (3), $\mathrm{End}(E) \subset B_{p,\infty}$ no es conmutativo, pero sigue siendo útil en criptografía para construir protocolos de otras maneras. Este caso corresponde a aplicaciones como SQIsign [1] y sus variantes. En lo que sigue nos enfocamos al caso numero (2), donde $\mathrm{End}(E)$ y/o $End_{\mathbb{F}_p}(E)$ son conjuntos de $\mathbb{Q}(\sqrt{\Delta})$, y entonces son conmutativos.

4. Acción de grupo de CSIDH

Vamos a enfocarnos a curvas supersingulares y sus anillos de endomorfismos sobre $\mathbb{F}_p$.

De nuevo, considera el conjunto de isogenias $\mathrm{Hom}(E, E')$. Para cualquier isogenia $\varphi \in \mathrm{Hom}(E, E')$, si multiplicamos (a la derecha) por un endomorfismo $\omega \in \mathrm{Hom}(E, E)$, obtenemos $\varphi \circ \omega \in \mathrm{Hom}(E, E')$. Esto quiere decir que el grupo $\mathrm{Hom}(E, E')$ es cerrado bajo multiplicación (de la derecha) por miembros del anillo $\mathrm{End}(E)$. Es decir, el conjunto de isogenias $\mathrm{Hom}(E, E')$ es un ideal del anillo $\mathrm{End}(E)$.

Dos ideales I, J de $\mathrm{End}(E)$ son equivalentes si existe un $\alpha \in \mathbb{Q}(\sqrt{\Delta})$ donde $I = J\alpha$. El conjunto de ideales distintos bajo esta definición de equivalencia se llama el *grupo de clases de ideales*, y es un grupo multiplicativo. Es denotado como $\mathrm{cl}(\mathcal{O})$, donde $\mathrm{End}_{\mathbb{F}_p}(E)$ es isomorfo a cierto orden $\mathcal{O}$ en el campo imaginario cuadrático $\mathbb{Q}(\sqrt{\Delta})$. Esto quiere decir que los ideales principales $\langle\alpha\rangle$ forman la clase trivial de $\mathrm{cl}(\mathcal{O})$. Para un estudio más detallado de los grupos de clases de ideales, véase el libro de Cox [14, Sect. 7].

Ejemplo 3. *Por ejemplo, considera el ideal* $\mathfrak{a} = \langle \pi - 1, 3\rangle$. *Aquí* π *es el mapa de Frobenius que manda un punto* $P = (x, y)$ *a* $\pi(P) = (x^p, y^p)$. *El ideal tiene*

dos generadores, así que no es un ideal principal, y entonces no es el elemento de identidad de cl($\mathcal{O}$).

El inverso de $\mathfrak{a}$ *es* $\mathfrak{a}^{-1} = \langle \pi + 1, 3 \rangle$.

Ejercicio 4.1 Sea E una curva supersingular, y $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$. Sea ℓ_i un primo donde $\ell_i \mid p+1$. Muestra que el ideal $\langle \pi + 1, \ell_i \rangle \in \mathcal{O}$ y el ideal $\langle \pi - 1, \ell_i \rangle \in \mathcal{O}$ son inversos.

Pista: Muestra que $\langle \pi + 1, \ell_i \rangle \cdot \langle \pi - 1, \ell_i \rangle = \langle \ell_i \rangle$. Por recuerdo, $\pi^2 = -p$.

Entonces, con estas estructuras, podemos definir una acción de grupo donde el grupo de clases de ideales actúa sobre el conjunto de curvas elípticas con anillo de endomorfismos isomorfo a $\mathcal{O}$. Este conjunto de curvas puede definirse alternativamente como curvas elípticas con *multiplicación compleja* (CM) por $\mathcal{O}$.

Tomando X como el conjunto de curvas elípticas supersingulares sobre $\mathbb{F}_p$ que tengan grupo de endomorfismos isomórfico a un orden imaginario cuadrático particular,

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

y G el grupo de clases de ideales de $\mathcal{O}$, cl($\mathcal{O}$), podemos construir una acción de grupo

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

Utilizamos $[\mathfrak{a}]$ para denotar la clase de ideales del ideal $\mathfrak{a}$. Entonces, $[\mathfrak{a}]$ actúa sobre cualquier curva elíptica $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ a través de una isogenia

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

donde $\ker(\varphi_{\mathfrak{a}}) = \cap_{\alpha \in \mathfrak{a}} \ker(\alpha)$ y $E/[\mathfrak{a}]$ es el codominio de $\varphi_{\mathfrak{a}}$.

Ejemplo 4. *Del ejemplo precedente,* $\mathfrak{a} = \langle \pi - 1, 3 \rangle$. *Este ideal corresponde a la isogenia con kernel* $\ker \varphi_{\mathfrak{a}} = \{P \in E(\bar{\mathbb{F}}_p) : [3]P = 0_E \text{ y } \pi(P) - P = 0_E\}$.

Esta acción nos da una nueva manera de tomar pasos en el grafo de isogenias. Por ejemplo, empezando desde una curva $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$, podemos seleccionar un elemento $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$, y calcular

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

Este paso corresponde a una isogenia de grado $n(\mathfrak{a})$, donde $n(\cdot)$ se refiere a la norma del ideal. Como ya vimos, calcular isogenias de alto grado no es fácil. Así que queremos dividir $[\mathfrak{a}]$ en pasos mas chicos. Sea $\mathfrak{l}_1, \ldots \mathfrak{l}_k$ un conjunto de elementos (primos) de cl($\mathcal{O}$) de baja norma y que generen una gran parte de cl($\mathcal{O}$). Entonces, en vez de seleccionar $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$, podemos seleccionar un vector $[e_1, \ldots e_k] \in [-m, m]^k$ y calcular

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

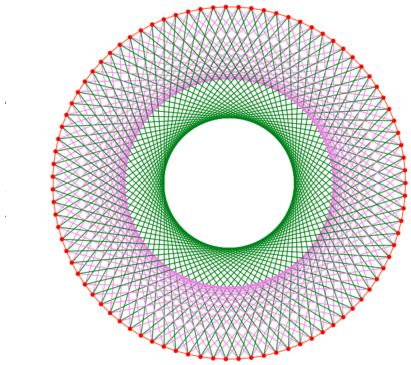

Figura 3: Grafo de isogenias sobre $\mathbb{F}_p$, con $p = 4 \cdot 5 \cdot 13 \cdot 61 - 1$. Las lineas anaranjadas corresponden a isogenias de grado 5, las verdes a isogenias de grado 13, y las moradas a isogenias de grado 61. La imagen fue hecha por Francisco Rodríguez-Henríquez [30].

donde m es seleccionado según el tamaño de $\mathrm{cl}(\mathcal{O})$. Usando esta construcción, la acción de grupo se convierte en una acción de grupo efectiva restringida (REGA).

De esta forma, nos aseguramos que la isogenia correspondiente se pueda construir de isogenias con grados $n(\mathfrak{l}_i)$. En la Figure 3 se ve un ejemplo de un grafo de isogenias con tres opciones de $\mathfrak{l}_i$.

4.1. CSIDH

Esta acción de grupo fue usada en [12] para construir *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH). Este protocolo es muy similar al Protocolo 2. CSIDH sigue siendo de los protocolos de isogenias más populares hoy en día.

Protocolo 3. *Primero se definen los parámetros públicos : un primo* $p \equiv 3$ mód 4*, el conjunto* $\mathcal{E}\ell\ell_p(\mathcal{O})$*, una curva* $E_0 \in \mathcal{E}\ell\ell_p(\mathcal{O})$ *inicial, los elementos* $\mathfrak{l}_1, \dots \mathfrak{l}_k$*, y el entero* m*.*

Ahora Alicia y Beto escogen sus vectores secretos $(a_1, \dots a_k), (b_1, \dots b_k) \in [-m, m]^k$*.*

Alicia publica la curva

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

y Beto la curva

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

Como la acción es conmutativa, cada uno puede calcular la curva secreta

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

Este próximo ejercicio, que empieza un ejemplo simplificado de CSIDH, fue escrita por 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 Usando el código dado, escriba una función en Sage que tome como entrada una curva elíptica y un grado ℓ, calcule una isogenia de grado ℓ usando la función .isogeny(), y devuelva el codominio de esta isogenia.

Ejercicio 4.3 Usa el ejemplo precedente y tu función del ejercicio precedente para verificar en Sage que la acción de grupo de isogenias es conmutativa.

Escriba funciones que implementen el Protocolo 2 usando la acción de grupo de isogenias. Sus funciones deben incluir key_gen() y compute_shared_secret().

Ejercicio 4.4 De una prueba de que CSIDH sea un REGA.

Ejercicio 4.5 El protocolo CRS [13, 31], que fue el antecedente de CSIDH, también usaba una acción de grupo de isogenias, pero donde el conjunto X estaba hecho de curvas elípticas ordinarias. Por qué es más útil usar curvas supersingulares?

Pista: Para curvas supersingulares, $\#E(\mathbb{F}_p) = p + 1$.

4.1.1. Torsión cuadrática

La *torsión cuadrática* de una curva es muy importante en CSIDH. Sea una curva $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ fija tal que $E : y^2 = f(x)$. Entonces, su *torsión cuadrática* es

isomorfa a la curva $E^t : d \cdot y^2 = f(x)$, donde d es cualquier residuo no cuadrático módulo p. Pensando a la acción de grupo, tenemos que en general

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

donde se usa el exponente t para denotar la torsión cuadrática. Por lo general, las curvas $\mathbb{F}_p$-racionales utilizadas en CSIDH serán isomorfas a sus torsiones sobre $\mathbb{F}_{p^2}$, pero no sobre $\mathbb{F}_p$. La curva inicial, sin embargo, se selecciona especialmente para que *sí* sea $\mathbb{F}_p$-isomorfa a su torsión. Al denotar esta curva como E_0 (la cual tiene invariante-j igual a 1728 y donde $p \equiv 3$ mód 4), se obtiene que

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

Esta es la razón por la cual el cómputo de torsiones resulta eficiente en la acción de grupo de CSIDH, y permite calcular vectores secretos con coeficientes negativos eficientemente.

4.2. CSI-FiSh

A pesar del uso de curvas supersingulares y clases de ideales de normas pequeñas para calcular la acción de grupo, esta sigue siendo una computación muy lenta en términos criptográficos. Por lo tanto, los autores de la firma digital CSI-FiSh [7] optaron por precomputar un rede de relaciones de los primos pequeños para poder calcular la acción de grupo.

4.2.1. El rede de relaciones

El grupo de clases de ideales es cíclico, así que con un generador $\mathfrak{g}$ de orden N, se puede describir todos los miembros del grupo de la forma $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$. Pero no queda claro como calcular elementos $\mathfrak{g}^z$ con alta norma.

Ejercicio 4.6 Por qué puede ser mejor calcular elementos del grupo de clases de ideales de la forma $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$, donde $\mathfrak{g}$ es un generador, en vez de usar la forma denotada en CSIDH?

La idea es usar los primos de baja norma $\mathfrak{l}_i$. Definimos un rede

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

Este rede se llama el *rede de relaciones*, y da una idea de la estructura del grupo, que anteriormente faltaba en CSIDH. Se ve que el grupo de clases de ideales es $\mathrm{cl}(\mathcal{O}) \simeq \mathbb{Z}^n / L$. Esto quiere decir que para un elemento $\mathbf{z} = [z_1, \dots z_n]$ (que no tiene que ser de coeficientes bajos) se puede simplificar por encontrar el punto de L más cerca a $\mathbf{z}$, dile $\mathbf{x}$, y envés usar $\mathbf{z} - \mathbf{x}$ que tiene norma más pequeña.

El cálculo del rede se puede hacer como precomputación, pero es muy costosa. Se requirió un estimado de 43 años de núcleo para calcular el rede de

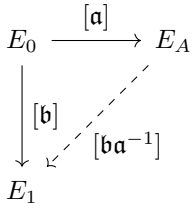

Figura 4: Figura del protocolo CSI-FiSh.

relaciones de CSIDH/CSI-FiSh con un primo p donde $\log_2 p = 512$ (hoy en día es pensado que estos parámetros tienen nivel de seguridad por de bajo de 128 bits).

Este algoritmo funciona para todos los $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$. Entonces la acción de grupo ya no es restringida según la definición de REGA (Definición 1)!

4.2.2. El esquema

CSI-FiSh se basa en las ideas de Stolbunov para obtener un esquema de firma para la acción de grupo de CSIDH. El esquema parte de una prueba de conocimiento estándar.

Protocolo 4. *Se comienza fijando la curva inicial $E_0 : y^2 = x^3 + x$. Luego, el probador escoge su secreto, $[\mathfrak{a}] \in cl(\mathcal{O})$, y calcula su clave pública $E_A = [\mathfrak{a}]E_0$. Para probar el conocimiento de su clave secreta ante un verificador, primero envía un compromiso aleatorio $E_1 = [\mathfrak{b}]E_0$. El verificador responde con un desafío $c \in \{0, 1\}$. Si $c = 0$, entonces el probador envía $r = [\mathfrak{b}]$, y el verificador comprueba que $E_1 = [r]E_0$. De lo contrario, cuando $c = 1$, el probador envía $r = [\mathfrak{b}\mathfrak{a}^{-1}]$, y el verificador comprueba que $E_1 = [r]E_A$.*

El diagrama de isogenias para este esquema se esboza en la Figura 4.

Ejercicio 4.7 El protocolo de CSI-FiSh descrito aquí tiene *soundness* (soundness) igual a 1/2. Por qué?

Es posible bajar la soundness usando varias llaves secretas y públicas. Es decir, la llave pública se vuelve una lista de curvas $(E_1, \dots E_n)$, y la llave secreta una lista de ideales $([\mathfrak{a}_1], \dots [\mathfrak{a}_n])$. Cómo adaptarás el protocolo para este cambio? Cuál es la nueva soundness?

Ejercicio 4.8 Por qué es necesario que la acción de grupo en este esquema sea un EGA, y no un REGA?

Pista: Considera cuando el desafío es $c = 1$.

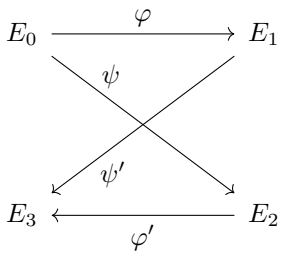

Figura 5: Cuadrado de Kani.

4.2.3. SCALLOP

Los autores de SCALLOP [18] cambian el conjunto de curvas X para obtener grupos de clases de ideales distintas, donde el rede de relaciones se puede calcular más fácilmente. Esto permite el cálculo de grupos para parámetros de seguridad más altos. Mientras que este estudio mejoró las opciones de seguridad, seguía siendo muy lento para uso criptográfico, y necesita un equilibrio de parámetros muy particulares.

5. Acción de grupo de PEGASIS

PEGASIS [15] ofrece todavía otra forma de calcular la acción de grupo de isogenias, pero esta vez usando isogenias de dimensión 4. Se presenta una breve descripción de PEGASIS siguiendo las notas de Eriksen [19].

En 2022, una serie de ataques [11, 25, 29] fueron publicados demostrando como atacar el protocolo SIDH [22] usando isogenias de dimensiones 2, 4, o 8. La idea es incrustar la isogenia objetivo en una de mayor dimensión, en las que se conoce cómo actúa esta última sobre un determinado subgrupo de torsión. Esta información había sido parte de la información pública en SIDH. Los protocolos como CSIDH no publican este tipo de información, por lo que no son vulnerables a este ataque.

No describiremos el ataque de Kani en detalle, ya que está fuera del alcance de estas notas. Para más detalles, consulte los artículos originales del ataque [11, 25, 29]. De manera muy simple, para construir una isogenia de dimensión 2 $\Phi : E_0 \times E_3 \to E_1 \times E_2$, requerimos una construcción como la de Figura 5, donde $\psi' \circ \varphi = \varphi' \circ \psi$.

El grado de Φ es entonces $\deg \Phi = \deg \varphi + \deg \psi' = \deg \varphi' + \deg \psi$. Con información sobre ψ' y Φ (con grados particulares), permite evaluar φ (de cualquier grado) eficientemente.

Ahora supongamos que tenemos una clase de ideales $[\mathfrak{a}]$ que deseamos evaluar. En Clapoti [26], los autores muestran que al encontrar dos ideales equivalentes a $\mathfrak{a}$, $[\mathfrak{a}] = [\mathfrak{b}] = [\mathfrak{c}]$, tales que $n(\mathfrak{b}) + n(\mathfrak{c}) = 2^e$, entonces la acción de $[\mathfrak{a}]$ se puede computar eficientemente utilizando el ataque de Kani. Sin embargo, no es evidente como encontrar estos ideales, y eliminar el requisito del grado

implicaría utilizar isogenias de dimensión 8, las cuales no son eficientes en la práctica (hoy en día).

En KLaPoTi [28], los autores encuentran los ideales $\mathfrak{b}, \mathfrak{c}$ utilizando un algoritmo conocido, KLPT [23]. No obstante, el valor de e es muy grande, lo que hace que todos los cómputos sean más costosos. En PEGASIS, los autores encuentran un punto medio en el que son capaces de utilizar isogenias de dimensión 4, pero con un valor de e que no es demasiado grande. Más adelante, en un trabajo posterior a PEGASIS, qt-PEGASIS [16] usa un enfoque diferente para calcular los ideales equivalentes que aprovecha la resolución de ecuaciones de norma en un álgebra de cuaterniones. Estos cambios terminan proporcionando una aceleración en el cómputo de la acción de grupo, convirtiéndola en la implementación de acción de grupo de isogenias más rápida hasta ahora.

Clapoti, KLaPoTi, y (qt)PEGASIS son acciónes de grupo no restringidas y aplican a más conjuntos de curvas, no solamente el de CSIDH. Tabla 2 muestra una comparación entre varias acciones de grupo.

Cuadro 2: Comparación de los tiempos de ejecución (en segundos) frente a diferentes tamaños de números primos para diversos esquemas. Nota: estas son solo aproximaciones, ya que los esquemas se ejecutaron en hardware diferente. Los valores de la tabla se han tomado de [15, Tabla 3] y [16, Tabla 1].

Esquema	**Artículo**	**Programación**	**Tamaño approx. del primo (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

Ejercicio 5.1 Usa PEGASIS para implementar la firma digital CSI-FiSh. Programa el protocolo de sigma clásico de CSI-FiSh, y también el protocolo descrita en el ejercicio 8.

El código de PEGASIS se encuentra en `https://github.com/KULeuven-COSIC/qt-pegasis`. Puedes usar el siguiente código de base, escrita por Riccardo Invernessi, para empezar.

```
from qt_pegasis import qtPegasis

lvl = '500'
EGA = qtPegasis(lvl)

// clave secreta
frak_a = EGA.sample_sage_ideal()

```

```
// clave publica
EA = EGA.sage_action(frak_a)
```

6. Problemas abiertos

A continuación se presentan dos áreas de la literatura que están activas y que pueden resultar más accesibles para nuevos investigadores en este campo.

6.1. Protocolos avanzados

La parte más atractiva de una acción de grupo conmutativa es su similitud con el protocolo de Diffie-Hellman. Diffie-Hellman se ha usado para el intercambio de claves, pero también para muchos otros protocolos avanzados. Por lo tanto, contar con un análogo postcuántico deja la puerta abierta a la creación de versiones postcuánticas de muchos otros protocolos. Ya existen varios ejemplos, tales como firmas de umbral [4,20] o intercambio de claves autenticado por contraseña (PAKE) [2], entre otros.

Esto plantea la pregunta abierta: ¿qué otros tipos de esquemas se pueden volver postcuánticos usando una acción de grupo como CSIDH o qt-PEGASIS? También queda mucho por hacer en lo que respecta al desarrollo o la mejora de los protocolos avanzados existentes. Esto podría incluir la programación de algoritmos, la búsqueda de mejoras algorítmicas o la investigación de las declaraciones de seguridad.

Por ejemplo, en [2] dos PAKEs de acciones de grupo conmutativas son presentados. Las construcciones son más bien lentas, sobre todo cuando se usa CSIDH. Por eso los autores no incluyeron implementaciones de los protocolos en el artículo. Ya que la teoría de las acciones de grupo de isogenias se ha desarrollado más, cómo podría mejorar este protocolo en términos de eficiencia? Hay otros cambios que se podrían hacer para mejorarlos?

6.2. Ataque de inyección de fallos

Un *ataque de inyección de fallos* ocurre cuando se manipula el hardware que ejecuta un protocolo con la esperanza de que el protocolo alterado revele algún tipo de información secreta. Este modelo de ataque posee tanto un aspecto de hardware como uno teórico: cómo se puede inducir un fallo en el protocolo y, una vez alterado, cómo revela esto información sobre el secreto? Al diseñar ataques que emplean fallos estándar, uno puede enfocarse mas en el aspecto teórico.

A continuación se presenta una lista de fallos que se consideran estándar y cuya implementación resulta razonable [8]:

- *Manipulación de memoria.* Ocurre cuando se altera un byte de una variable en la memoria. Por ejemplo, asignando un nuevo valor aleatorio a dicho byte. Un ataque más fuerte podría incluir el volteo de bits (bit-flip) o el establecimiento de valores a cero.

Cuadro 3: La tabla describe el número de fallos necesarios para implementar cada ataque (cuando se implementa con un objetivo de $\lambda = 128$ bits de seguridad). Estos valores son independientes de la probabilidad de éxito para los diferentes modelos de inyección de fallos. Todos los fallos son de primer orden. La columna final detalla el nuevo tamaño del espacio de secretos tras los fallos inyectados y el posprocesamiento.

Protocolo	Articulo	# de fallos	Espacio de secreto efectivo
CSIDH	[9, Ataque 3]	178	$x \in [1, 2^{67}]$
	[5, Sec. 5.1]	512	1
	[24, Tabla 1]	263	1

- *Aborto temprano de bucles.* Ocurre cuando un bucle en un algoritmo se interrumpe antes de lo previsto. Cuanta menos precisión se requiera, más razonable resulta el ataque.
- *Omisión de instrucciones.* Ocurre cuando se salta una línea de código específica. De nuevo, cuanto mayor sea la precisión necesaria, más difícil resulta en la práctica.

Por lo tanto, la pregunta abierta es: ¿se puede recuperar información secreta de un protocolo utilizando uno de estos fallos?

6.2.1. Estado del arte

Para CSIDH, ya se han publicado varios artículos sobre ataques de inyección de fallos. Estos ataques de fallos se dirigen contra operaciones ficticias (dummy operations) y obtienen un oráculo al observar si la salida cambia tras un fallo; es decir, si el fallo afectó a una operación necesaria o a una ficticia [9,10,24]. Por lo general, estos ataques de fallos emplean un modelo de fallos que es fácil de lograr en la práctica, ya que no se requiere que sea muy preciso, y puede incluir omisiones de instrucciones, fallos en la memoria o abortos de bucles. Por otro lado, estos ataques requieren muchas muestras de ejecuciones erróneas, habitualmente del orden de cientos como mínimo. De manera similar, un ataque de fallos en el muestreo de puntos en CSIDH presenta requisitos parecidos respecto al modelo de fallos, pero exige una cantidad aún mayor de muestras erróneas [5]. Vea el Cuadro 3.

Hasta ahora no se han publicado ataques contra las otras acciones de grupo. No es claro si los ataques precedentes pueden ser adaptados para las otras acciones, pero normalmente estos ataques se enfocan en implementaciones específicas de cada protocolo entonces adaptarlas no sería trivial. También se puede explorar ataques totalmente nuevos.

6.2.2. Fallo de desorientación

El ataque de [5] falla la "orientación" de uno de los pasos en el calculo de la curva compartida en CSIDH. Aquí, orientación se refiere al signo de un exponente. Por ejemplo, en fallar el primer paso del calculo, obtienen la curva

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

en vez del usual $E_{AB} = \prod_{i=1}^{k} \mathfrak{l}_i^{e_i} \star E_0$. Entre E_1 y E_{AB} existe una isogenia $\varphi : E_1 \to E_{AB}$ de grado $\deg \varphi = (\mathfrak{l}_1^{e_1})^2$. Entonces en obtener E_1 y E_{AB}, un adversario puede buscar una isogenia **corta** entre los dos para descubrir información sobre el vector secreto.

Este fallo se puede obtener con un omisión de instrucciones o con una manipulación de memoria. Vea [5] para detalles sobre el ataque.

6.2.3. Fallos sobre ocultación de puntos de torsión

A causa del ataque de Kani, protocolos de hoy en día no pueden publicar información del acción de una isogenia secreta sobre un grupo de torsión. Es decir, si $\varphi : E_0 \to E_1$ es una isogenia, teniendo $\langle P, Q \rangle = E_0[N]$ y $\varphi(P), \varphi(Q)$, para $N \geq 4 \cdot \deg \varphi$, es suficiente para reconstruir φ usando una isogenia de alta dimensión. Por eso protocolos como la firma digital PRISM [6] en vez de publicar $\varphi(P), \varphi(Q)$ en la firma, publican

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

donde los $m_{ij} \in \mathbb{Z}$ son parte del secreto. Esta manera de esconder la información de torsión se llama *ocultación de torsión* (torsion masking).

La idea de [21] consiste en fallar uno de los coeficientes m_{ij} con una manipulación de memoria. Digamos que el m_{11} fue fallado a m'_{11}. Con este cambio, el protocolo publica

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

Entonces, se puede calcular $P' - P'' = [m_{11} - m'_{11}]\varphi(P)$. En general, el valor de $m_{11} - m'_{11}$ tendrá muy pocas opciones, entonces permite obtener el valor de $\varphi(P)$. Repitiendo el ataque, pero esta vez fallando m_{21} o m_{22}, da el valor de $\varphi(Q)$.

7. Otros recursos

Para los que estén interesado en continuar su estudio de isogenias, el *Isogeny Club* [32] organiza charlas cada dos semanas en Zoom sobre temas de isogenias. Todas las charlas las comparten en YouTube.

8. Ejercicios

Ejercicio 1.1 Escriba código de Sage que implemente el protocolo Diffie-Hellman donde el grupo utilizado sea el grupo de puntos sobre la curva elíptica $E : y = f(x)$, donde $f(x) = x^3 + 486662x^2 + x$ sobre el campo finito $\mathbb{F}_p$ con $p = 2^{255} - 19$. Usa $G = (9, y)$ como generador.

Puedes usar el siguiente código de base para empezar.

Ejercicio 1.2 Cuál es la relación entre Problema 1 y Problema 2? Son equivalentes?

Ejercicio 1.3 Sea E una curva elíptica, $P, Q \in E[N]$. Un *emparejamiento* (pairing) es un mapa bilinear que se puede definir en E (o otros grupos) de la forma $e : E \times E \to \mu_N$ donde $e(P, [r]Q) = e([r]P, Q) = e(P, Q)^r$.

Usa un emparejamiento para dar un ejemplo en que Problema 3 es fácil pero Problema 2 sigue siendo difícil.

Es importante por Protocolo 1 que Problema 3 sea difícil?

Ejercicio 1.4 Considera el protocolo de Diffie-Hellman usando puntos de una curva elíptica (ECDH), como en el ejercicio precedente. El algoritmo de Shor permitiría calcular el *periodo* de una función cíclica. Por ejemplo, para la función $f(a, b) := [a]P + [b]Q$, el algoritmo de Shor puede calcular r_a, r_b donde $f(a + r_a, b + r_b) = f(a, b)$.

Usando el algoritmo de Shor, describe un ataque contra ECDH.

Ejercicio 3.1 Según el teorema de Tate, dos curvas son isógenas sobre $\mathbb{F}_q$ si y solo si tienen el mismo número de puntos sobre $\mathbb{F}_q$.

Usa Sage para verificar el teorema de Tate.

a. Selecciona una curva elíptica sobre el campo $\mathbb{F}_q$.

b. Construye una isogenia.

c. Cuenta el número de puntos sobre el dominio y codominio de la isogenia sobre $\mathbb{F}_q$.

Ejercicio 3.2 De una prueba que $\mathrm{End}(E)$ es un anillo.

Ejercicio 4.1 Sea E una curva supersingular, y $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} \subset \mathbb{Q}(\sqrt{\Delta})$. Sea ℓ_i un primo donde $\ell_i \mid p+1$. Muestra que el ideal $\langle \pi + 1, \ell_i \rangle \in \mathcal{O}$ y el ideal $\langle \pi - 1, \ell_i \rangle \in \mathcal{O}$ son inversos.

Pista: Muestra que $\langle \pi + 1, \ell_i \rangle \cdot \langle \pi - 1, \ell_i \rangle = \langle \ell_i \rangle$. Por recuerdo, $\pi^2 = -p$.

Ejercicio 4.2 Usando el código dado, escriba una función en Sage que tome como entrada una curva elíptica y un grado ℓ, calcule una isogenia de grado ℓ usando la función .isogeny(), y devuelva el codominio de esta isogenia.

Ejercicio 4.3 Usa el ejemplo precedente y tu función del ejercicio precedente para verificar en Sage que la acción de grupo de isogenias es conmutativa.

Escriba funciones que implementen el Protocolo 2 usando la acción de grupo de isogenias. Sus funciones deben incluir key_gen() y compute_shared_secret().

Ejercicio 4.4 De una prueba de que CSIDH sea un REGA.

Ejercicio 4.5 El protocolo CRS [13, 31], que fue el antecedente de CSIDH, también usaba una acción de grupo de isogenias, pero donde el conjunto X estaba hecho de curvas elípticas ordinarias. Por qué es más útil usar curvas supersingulares?

Pista: Para curvas supersingulares, $\#E(\mathbb{F}_p) = p + 1$.

Ejercicio 4.6 Por qué puede ser mejor calcular elementos del grupo de clases de ideales de la forma $\mathfrak{g}^z$, $z \in \mathbb{Z}_N$, donde $\mathfrak{g}$ es un generador, en vez de usar la forma denotada en CSIDH?

Ejercicio 4.7 El protocolo de CSI-FiSh descrito aquí tiene *soundness* (soundness) igual a 1/2. Por qué?

Es posible bajar la soundness usando varias llaves secretas y públicas. Es decir, la llave pública se vuelve una lista de curvas $(E_1, \dots E_n)$, y la llave secreta una lista de ideales $([\mathfrak{a}_1], \dots [\mathfrak{a}_n])$. Cómo adaptarás el protocolo para este cambio? Cuál es la nueva soundness?

Ejercicio 4.8 Por qué es necesario que la acción de grupo en este esquema sea un EGA, y no un REGA?

Pista: Considera cuando el desafío es $c = 1$.

Ejercicio 5.1 Usa PEGASIS para implementar la firma digital CSI-FiSh. Programa el protocolo de sigma clásico de CSI-FiSh, y también el protocolo descrita en el ejercicio 8.

El código de PEGASIS se encuentra en https://github.com/KULeuven-COSIC/qt-pegasis. Puedes usar el siguiente código de base, escrita por Riccardo Invernessi, para empezar.

```
% para el ejercicio 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)
```

```
% para los ejercicios 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)
```

```
% para el ejercicio 5.1
from qt_pegasis import qtPegasis

lvl = '500'
EGA = qtPegasis(lvl)

% clave secreta
frak_a = EGA.sample_sage_ideal()

% clave publica
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.