Fast and Frobenius: Rational Isogeny Evaluation over Finite Fields

Gustavo Banegas[1], **Valerie Gilchrist**[2], Anaëlle Le Devehat[3], Benjamin Smith [3]

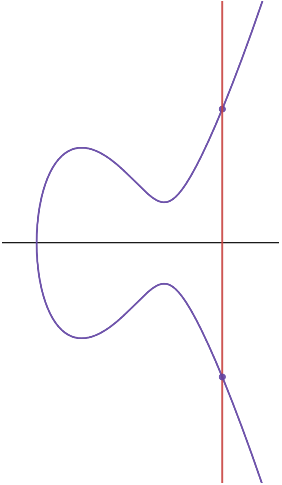
Qualcomm France SARL, Valbonne, France

Université Libre de Bruxelles and FNRS, Brussels, Belgium

Inria and Laboratoire d'Informatique de l'École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

October 3, 2023

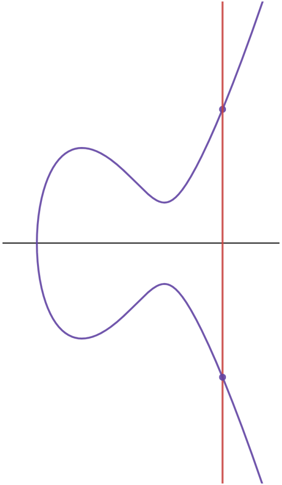
Elliptic curves and isogenies



An *elliptic curve* can (usually) be written in the form

$$E : y^2 = x^3 + ax + b.$$

Elliptic curves and isogenies

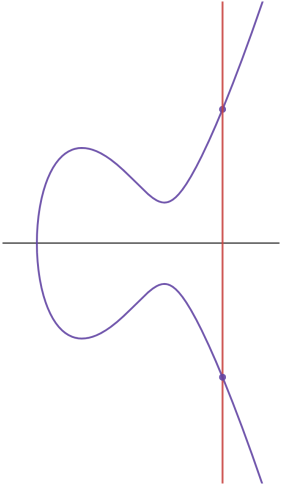


An *elliptic curve* can (usually) be written in the form

$$E : y^2 = x^3 + ax + b.$$

Points on elliptic curves form a *group* under addition.

Elliptic curves and isogenies



An *elliptic curve* can (usually) be written in the form

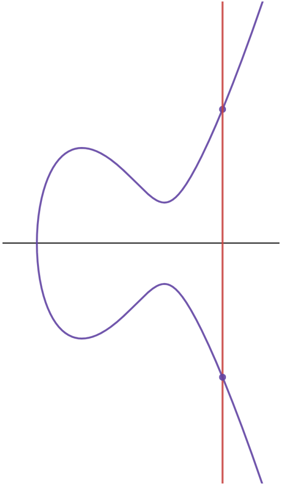
$$E : y^2 = x^3 + ax + b.$$

Points on elliptic curves form a *group* under addition.

Elliptic curves are *symmetric* about the x-axis, so

$$(x, y) \in E \implies (x, -y) \in E.$$

Elliptic curves and isogenies



An *elliptic curve* can (usually) be written in the form

$$E : y^2 = x^3 + ax + b.$$

Points on elliptic curves form a *group* under addition.

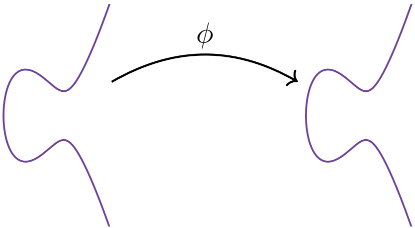
Elliptic curves are *symmetric* about the x-axis, so

$$(x, y) \in E \implies (x, -y) \in E.$$

Often we can restrict to using only x-coordinates.

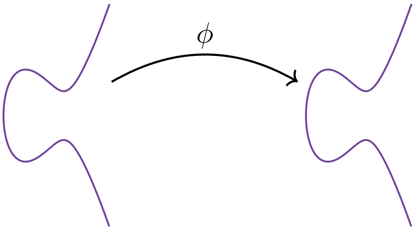
Elliptic curves and isogenies

An *isogeny* is a rational mapping between two elliptic curves.



Elliptic curves and isogenies

An *isogeny* is a rational mapping between two elliptic curves.



It is believed to be classically and quantumly hard to find an isogeny between two fixed elliptic curves.

Motivation

CSIDH is a key exchange scheme using an isogeny group action.

SQISign is a signature candidate in the NIST competition, using isogenies.

Some new isogeny protocols from 2023 :

- FESTA, SQISign HD
- M(D)-SIDH, binSIDH
- SCALLOP
- CSI-Otter, CSI-SharK
- CAPYBARA, TSUBAKI

Motivation

CSIDH is a key exchange scheme using an isogeny group action.

SQISign is a signature candidate in the NIST competition, using isogenies.

Some new isogeny protocols from 2023 :

- FESTA, SQISign HD
- M(D)-SIDH, binSIDH
- SCALLOP
- CSI-Otter, CSI-SharK
- CAPYBARA, TSUBAKI

By speeding up the computation of isogenies, we can speed up protocols that rely on them.
Cryptanalysis also benefits from improved computation times.

Computing isogenies

Vélu gives explicit equations for computing isogenies, given a generating point for its kernel

$$\langle P \rangle = \ker \phi.$$

Computing isogenies

Vélu gives explicit equations for computing isogenies, given a generating point for its kernel

$$\langle P \rangle = \ker \phi.$$

The *kernel polynomial* of an isogeny is used in these formulæ, given by

$$D(X) := \prod_{G \in S} (X - x(G))$$

where $S \subset \langle P \rangle$ is any subset such that

$$S \cap -S = \emptyset \qquad \text{and} \qquad S \cup -S = \langle P \rangle \setminus \{0\}.$$

Computing isogenies

Vélu gives explicit equations for computing isogenies, given a generating point for its kernel

$$\langle P \rangle = \ker \phi.$$

The *kernel polynomial* of an isogeny is used in these formulæ, given by

$$D(X) := \prod_{G \in S} (X - x(G))$$

where $S \subset \langle P \rangle$ is any subset such that

$$S \cap -S = \emptyset \qquad \text{and} \qquad S \cup -S = \langle P \rangle \setminus \{0\}.$$

i.e. S is the set of multiples of P **up to negation**.

Evaluating the kernel polynomial

Our computational building blocks consist of point doubling and adding.

$$\begin{aligned} \texttt{xDBL} : & x(P) & \mapsto x(2P) \\ \texttt{xADD} : & \big(x(P), x(Q), x(P-Q)\big) & \mapsto x(P+Q) \end{aligned}$$

Evaluating the kernel polynomial

Our computational building blocks consist of point doubling and adding.

$$\begin{aligned} \texttt{xDBL} : & x(P) && \mapsto x(2P) \\ \texttt{xADD} : & \big(x(P), x(Q), x(P-Q)\big) && \mapsto x(P+Q) \end{aligned}$$

Note, in point addition we require more information in order to differentiate between $(P+Q)$ and $-(P+Q)$.

Evaluating the kernel polynomial

Our computational building blocks consist of point doubling and adding.

$$\begin{aligned}\texttt{xDBL}: & x(P) && \mapsto x(2P) \\ \texttt{xADD}: & \big(x(P), x(Q), x(P-Q)\big) && \mapsto x(P+Q)\end{aligned}$$

Note, in point addition we require more information in order to differentiate between $(P+Q)$ and $-(P+Q)$.

Complexity costs:

xADD	xDBL
$4\mathbf{M}+2\mathbf{S}$	$2\mathbf{M}+2\mathbf{S}+1\mathbf{C}$

Here **M**, **S**, **C** represent multiplication, squaring, and multiplication by a curve constant, respectively.

Evaluating the kernel polynomial

Take $ord(P) = 13$.
We want to choose a set $S \subset \langle P \rangle$, that contains all multiples of P up to negation (excluding the identity) for use in our kernel polynomial.

Some classic examples are taking the first half.

Evaluating the kernel polynomial

Take $ord(P) = 13$.
We want to choose a set $S \subset \langle P \rangle$, that contains all multiples of P up to negation (excluding the identity) for use in our kernel polynomial.

Some classic examples are taking the first half.

$$P \quad 2P \quad 3P \quad 4P \quad 5P \quad 6P \quad 7P \quad 8P \quad 9P \quad 10P \quad 11P \quad 12P$$

Evaluating the kernel polynomial

Take $ord(P) = 13$.
We want to choose a set $S \subset \langle P \rangle$, that contains all multiples of P up to negation (excluding the identity) for use in our kernel polynomial.

Some classic examples are taking the first half.

$$\overbrace{P \quad 2P \quad 3P \quad 4P \quad 5P \quad 6P}^{P \quad 2P \quad 3P \quad 4P \quad 5P \quad 6P} \quad 7P \quad 8P \quad 9P \quad 10P \quad 11P \quad 12P$$

Evaluating the kernel polynomial

For example, choose

$$S = \{P, [2]P, [3]P, \ldots, [(ord(P) - 1)/2]P\}$$

This method would use one `xDBL` to get $[2]P$, and the rest `xADD`'s.

Fast(er)...

Evaluating the kernel polynomial

Another approach...

P

Evaluating the kernel polynomial

Another approach...

$$P \xrightarrow{\times 2} 2P$$

Evaluating the kernel polynomial

Another approach...

$$P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P$$

Evaluating the kernel polynomial

Another approach...

$$P \overset{\times 2}{\longrightarrow} 2P \overset{\times 2}{\longrightarrow} 4P \overset{\times 2}{\longrightarrow} 8P$$

Evaluating the kernel polynomial

Another approach...

$$P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P \xrightarrow{\times 2} 16P$$

Evaluating the kernel polynomial

Another approach...

$$P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P \xrightarrow{\times 2} 3P$$

Evaluating the kernel polynomial

Another approach...

$$P \overset{\times 2}{\curvearrowright} 2P \overset{\times 2}{\curvearrowright} 4P \overset{\times 2}{\curvearrowright} 8P \overset{\times 2}{\curvearrowright} 3P \overset{\times 2}{\curvearrowright} 6P \overset{\times 2}{\curvearrowright} 12P \overset{\times 2}{\curvearrowright} 11P \overset{\times 2}{\curvearrowright} 9P \overset{\times 2}{\curvearrowright} 5P \overset{\times 2}{\curvearrowright} 10P \overset{\times 2}{\curvearrowright} 7P$$

Evaluating the kernel polynomial

Another approach...

$$\underbrace{P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P \xrightarrow{\times 2} 3P \xrightarrow{\times 2} 6P} \xrightarrow{\times 2} 12P \xrightarrow{\times 2} 11P \xrightarrow{\times 2} 9P \xrightarrow{\times 2} 5P \xrightarrow{\times 2} 10P \xrightarrow{\times 2} 7P$$

This uses all `xDBL`'s.

Evaluating the kernel polynomial

Another approach...

$$\underbrace{P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P \xrightarrow{\times 2} 3P \xrightarrow{\times 2} 6P} \xrightarrow{\times 2} 12P \xrightarrow{\times 2} 11P \xrightarrow{\times 2} 9P \xrightarrow{\times 2} 5P \xrightarrow{\times 2} 10P \xrightarrow{\times 2} 7P$$

This uses all `xDBL`'s.

Here, 2 is *primitive* modulo 13, but this is not always the case.

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \overset{\times 2}{\curvearrowright} 2P \overset{\times 2}{\curvearrowright} 4P \overset{\times 2}{\curvearrowright} 8P$$

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \overset{\times 2}{\curvearrowright} 2P \overset{\times 2}{\curvearrowright} 4P \overset{\times 2}{\curvearrowright} 8P \overset{\times 2}{\curvearrowright} 16P$$

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \overset{\times 2}{\curvearrowright} 2P \overset{\times 2}{\curvearrowright} 4P \overset{\times 2}{\curvearrowright} 8P \overset{\times 2}{\curvearrowright} 16P = -P$$

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \overset{\times 2}{\curvearrowright} 2P \overset{\times 2}{\curvearrowright} 4P \overset{\times 2}{\curvearrowright} 8P$$

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P$$

$$3P$$

Generalizing to other ℓ

Example. Take $ord(P) = 17$. Here, 2 is not primitive.

$$P \xrightarrow{\times 2} 2P \xrightarrow{\times 2} 4P \xrightarrow{\times 2} 8P$$

$$3P \xrightarrow{\times 2} 2 \cdot 3P \xrightarrow{\times 2} 4 \cdot 3P \xrightarrow{\times 2} 8 \cdot 3P$$

In summary

Formalizing...

Let

$$M_\ell := (\mathbb{Z}/\ell\mathbb{Z})^\times / \langle \pm 1 \rangle$$

In summary

Formalizing...

Let

$$M_\ell := (\mathbb{Z}/\ell\mathbb{Z})^\times/\langle \pm 1 \rangle$$

For $\ell < 20000$ we have that

- 56% satisfy $M_\ell = \langle 2 \rangle$
- 83% satisfy $M_\ell = \langle 2, 3 \rangle = \bigsqcup \left(3^i \cdot \langle 2 \rangle\right)$

The remaining ℓ can be dealt with in a case-by-case basis.

...and Frobenius

Using extension fields

It is possible for the kernel generator of an isogeny to be taken from an extension field, $E(\mathbb{F}_{q^k})$.

Using extension fields

It is possible for the kernel generator of an isogeny to be taken from an extension field, $E(\mathbb{F}_{q^k})$.

We can still use classic Vélu, but the arithmetic over the extension fields makes it very costly.

Using extension fields

It is possible for the kernel generator of an isogeny to be taken from an extension field, $E(\mathbb{F}_{q^k})$.

We can still use classic Vélu, but the arithmetic over the extension fields makes it very costly.

We will try to use Frobenius: an endomorphism that maps

$$(x, y) \mapsto (x^q, y^q)$$

Exploiting the action of Frobenius

P is in $E(\mathbb{F}_{q^k})$.

Exploiting the action of Frobenius

P is in $E(\mathbb{F}_{q^k})$.

$\langle P \rangle$ is Galois stable, so Frobenius acts as an eigenvalue, λ, on $\langle P \rangle$.

Exploiting the action of Frobenius

P is in $E(\mathbb{F}_{q^k})$.

$\langle P \rangle$ is Galois stable, so Frobenius acts as an eigenvalue, λ, on $\langle P \rangle$.

$$\begin{array}{ccccccccc} P & \mapsto & \pi(P) & \mapsto & \pi^2(P) & \mapsto & \cdots & \pi^{k-1}(P) \\ P & \mapsto & \lambda P & \mapsto & \lambda^2 P & \mapsto & \cdots & \lambda^{k-1} P \end{array}$$

Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:

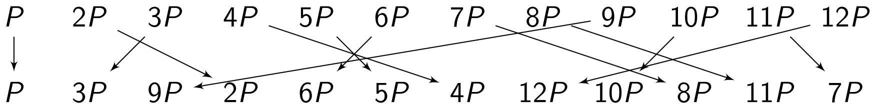
Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:

$$P \quad 2P \quad 3P \quad 4P \quad 5P \quad 6P \quad 7P \quad 8P \quad 9P \quad 10P \quad 11P \quad 12P$$

Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:



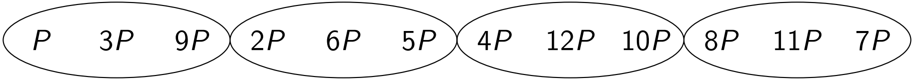
Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:

$$P \quad 3P \quad 9P \quad 2P \quad 6P \quad 5P \quad 4P \quad 12P \quad 10P \quad 8P \quad 11P \quad 7P$$

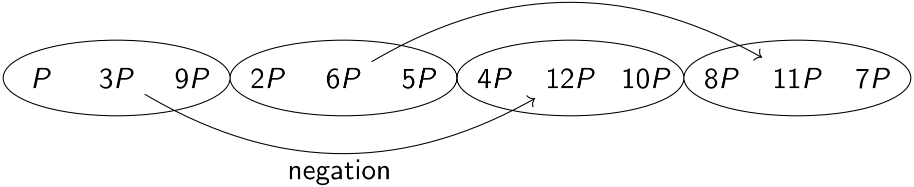
Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:



Exploiting the action of Frobenius

Take $ord(P) = 13$ and $k = 3$.
Here π acts as multiplication by 3 on $\langle P \rangle$:



The (low) cost of Frobenius

Note, the ideal way to implement Frobenius depends on the context, but can generally be done for cheap.

The (low) cost of Frobenius

Note, the ideal way to implement Frobenius depends on the context, but can generally be done for cheap.

Example. If $k = 2$ and $\mathbb{F}_{q^2} = \mathbb{F}_q(\sqrt{\Delta})$, then it maps $[a, b]$ to $[a, -b]$, so $\mathbf{F} \approx 0$.

In the worst case, for $k \leq 12$, $\mathbf{F} \approx \mathbf{M}$.

Experimental results

ℓ	k'	**M**	**S**	**C**	**a**	**F**	Algorithm
13	any	30	12	15	54	0	Costello–Hisil
	1	22	12	19	46	0	This work
	3	10	4	7	14	4	This work
19	any	48	18	21	84	0	Costello–Hisil
	1	34	18	28	70	0	This work
	3	14	6	10	22	4	This work
	9	18	2	4	6	16	This work
23	any	60	22	25	104	0	Costello–Hisil
	1	42	22	34	86	0	This work
	11	22	2	4	6	20	This work

Cost of evaluating an ℓ-isogeny at a single point over $\mathbb{F}_q$, using a kernel generator with x-coordinate in $\mathbb{F}_{q^{k'}}$.

In this table,
M = multiplications,
S = squares,
C = multiplications of elements of $\mathbb{F}_{q^{k'}}$ by elements of $\mathbb{F}_q$ (including, but not limited to, curve constants),
a = adds,
F = calls to Frobenius.