Post-quantum cryptography I – Isogeny-based cryptography

Tanja Lange
with lots of slides by Chloe Martindale and Lorenz Panny

Eindhoven University of Technology

01 February 2023

Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

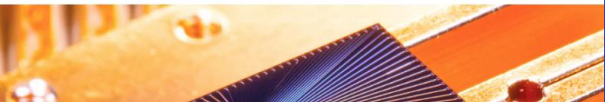
Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their compu-

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum com-

Quantum computing could end encryption within five years, says Google boss



Mr Pichai said a combination of artificial intelligence and quantum would "help us tackle some of the biggest problems we see", but said it was important encryption evolved to match this.

"In a five to ten year time frame, quantum computing will break encryption as we know it today."

This is because current encryption methods, by which information such as texts or passwords is turned into code to make it unreadable, rely upon the fact that classic computers would take billions of years to decipher that code.

Quantum computers, with their ability to be

National Academy of Sciences (US)

4 December 2018: Report on quantum computing

Don't panic. "Key Finding 1: Given the current state of quantum computing and recent rates of progress, it is highly unexpected that a quantum computer that can compromise RSA 2048 or comparable discrete logarithm-based public key cryptosystems will be built within the next decade."

National Academy of Sciences (US)

4 December 2018: Report on quantum computing

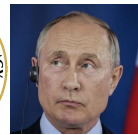
Don't panic. "Key Finding 1: Given the current state of quantum computing and recent rates of progress, it is highly unexpected that a quantum computer that can compromise RSA 2048 or comparable discrete logarithm-based public key cryptosystems will be built within the next decade."

Panic. "Key Finding 10: Even if a quantum computer that can decrypt current cryptographic ciphers is more than a decade off, the hazard of such a machine is high enough—and the time frame for transitioning to a new security protocol is sufficiently long and uncertain—that prioritization of the development, standardization, and deployment of post-quantum cryptography is critical for minimizing the chance of a potential security and privacy disaster."

"[Section 4.4:] In particular, all encrypted data that is recorded today and stored for future use, will be cracked once a large-scale quantum computer is developed."

Commonly used systems

Sender
"Alice"



Untrustworthy network
"Eve"

Receiver
"Bob"

Cryptography with symmetric keys
AES-128. **AES-192**. **AES-256**. **AES-GCM**. **ChaCha20**. **HMAC-SHA-256**. **Poly1305**. **SHA-2**. **SHA-3**. **Salsa20**.

Cryptography with public keys
BN-254. **Curve25519**. **DH**. **DSA**. **ECDH**. **ECDSA**. **EdDSA**. **NIST P-256**. **NIST P-384**. **NIST P-521**. **RSA encrypt**. **RSA sign**. **secp256k1**.

Commonly used systems

Sender
"Alice"

Untrustworthy network
"Eve" with quantum computer

Receiver
"Bob"

Cryptography with symmetric keys
AES-128. **AES-192**. **AES-256**. **AES-GCM**. **ChaCha20**. **HMAC-SHA-256**. **Poly1305**. **SHA-2**. **SHA-3**. **Salsa20**.

Cryptography with public keys
BN-254. **Curve25519**. **DH**. **DSA**. **ECDH**. **ECDSA**. **EdDSA**. **NIST P-256**. **NIST P-384**. **NIST P-521**. **RSA encrypt**. **RSA sign**. **secp256k1**.

Post-quantum cryptography

Cryptography under the assumption that the attacker has a quantum computer.

Post-quantum cryptography

Cryptography under the assumption that the attacker has a quantum computer.

5 Major categories of public-key post-quantum systems

- **Code-based** encryption: McEliece cryptosystem has survived since 1978. Short ciphertexts and large public keys. Security relies on hardness of decoding error-correcting codes.
- **Hash-based** signatures: very solid security and small public keys. Require only a secure hash function (hard to find second preimages).
- **Isogeny-based** encryption: new kid on the block, promising short keys and ciphertexts and non-interactive key exchange. Security relies on hardness of finding isogenies between elliptic curves over finite fields.
- **Lattice-based** encryption and signatures: possibility for balanced sizes. Security relies on hardness of finding short vectors in some (typically special) lattice.
- **Multivariate-quadratic** signatures: short signatures and large public keys. Security relies on hardness of solving systems of multivariate equations over finite fields.

Warning: These are categories of mathematical problems;
individual systems may be totally insecure if the problem is not used correctly.

We have a good understanding of what a quantum computer can do,
but new systems need more analysis.

Diffie–Hellman key exchange '76

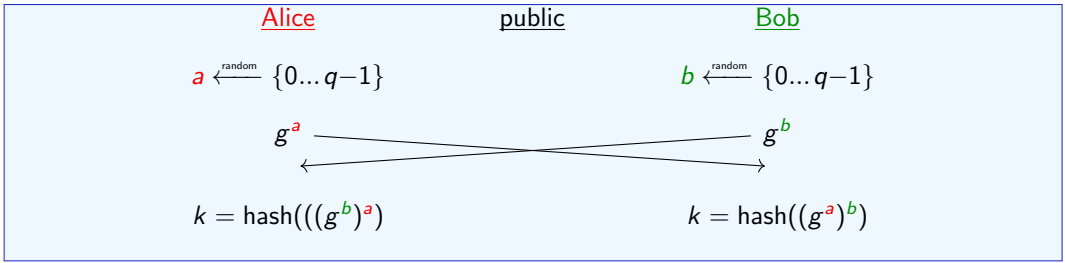
Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q

Diffie–Hellman key exchange '76

Public parameters:

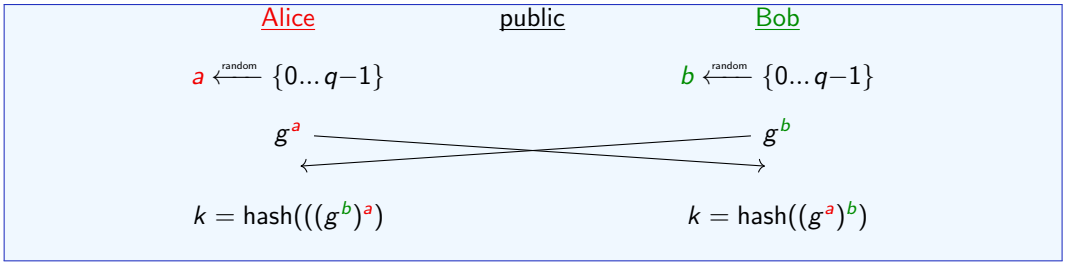
- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q



Diffie–Hellman key exchange '76

Public parameters:

- a finite group G (traditionally $\mathbb{F}_p^*$, today elliptic curves)
- an element $g \in G$ of prime order q



Fundamental reason this works: $\cdot^a$ and $\cdot^b$ *commute*!

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Is this a good idea?

Diffie–Hellman: Bob vs. Eve

Bob

1. Set $t \leftarrow g$.
2. Set $t \leftarrow t \cdot g$.
3. Set $t \leftarrow t \cdot g$.
4. Set $t \leftarrow t \cdot g$.

...

$b-2$. Set $t \leftarrow t \cdot g$.

$b-1$. Set $t \leftarrow t \cdot g$.

b. Publish $B \leftarrow t \cdot g$.

Attacker Eve

1. Set $t \leftarrow g$. If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$. If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$. If $t = B$ return 3.

...

$b-2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-2$.

$b-1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b-1$.

b. Set $t \leftarrow t \cdot g$. If $t = B$ return b.

$b+1$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+1$.

$b+2$. Set $t \leftarrow t \cdot g$. If $t = B$ return $b+2$.

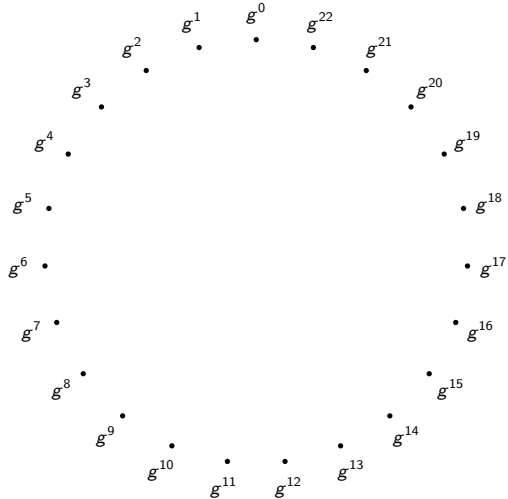
...

Diffie–Hellman: Bob vs. Eve

Bob	**Attacker** Eve	
1. Set $t \leftarrow g$.	1. Set $t \leftarrow g$.	If $t = B$ return 1.
2. Set $t \leftarrow t \cdot g$.	2. Set $t \leftarrow t \cdot g$.	If $t = B$ return 2.
3. Set $t \leftarrow t \cdot g$.	3. Set $t \leftarrow t \cdot g$.	If $t = B$ return 3.
4. Set $t \leftarrow t \cdot g$.	4. Set $t \leftarrow t \cdot g$.	If $t = B$ return 3.
...	...	
$b-2$. Set $t \leftarrow t \cdot g$.	$b-2$. Set $t \leftarrow t \cdot g$.	If $t = B$ return $b-2$.
$b-1$. Set $t \leftarrow t \cdot g$.	$b-1$. Set $t \leftarrow t \cdot g$.	If $t = B$ return $b-1$.
b. Publish $B \leftarrow t \cdot g$.	b. Set $t \leftarrow t \cdot g$.	If $t = B$ return b.
	$b+1$. Set $t \leftarrow t \cdot g$.	If $t = B$ return $b+1$.
	$b+2$. Set $t \leftarrow t \cdot g$.	If $t = B$ return $b+2$.
	...	

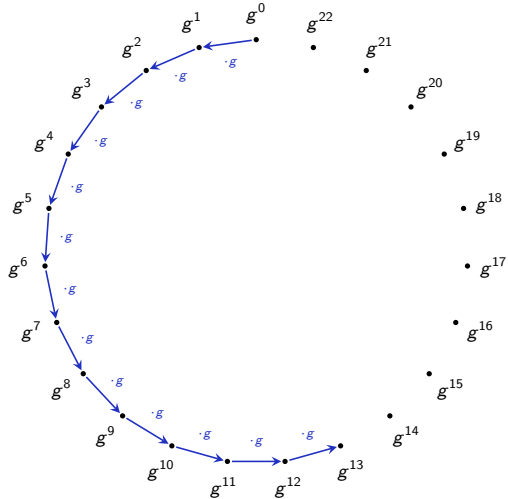
Effort for both: $O(\#G)$. Bob needs to be smarter.

(There also exist better attacks)



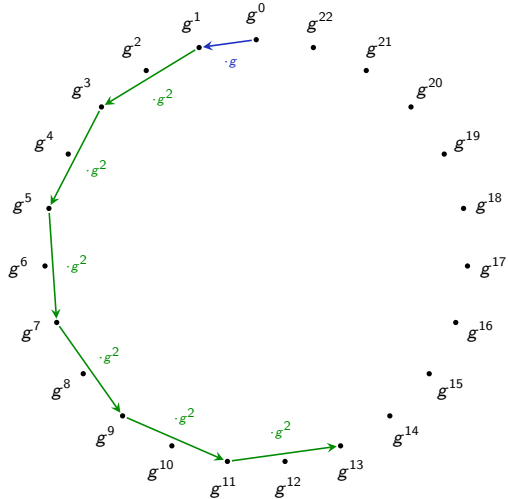
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

multiply



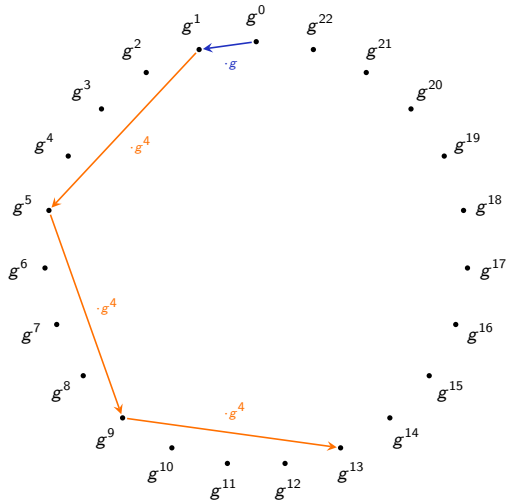
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply



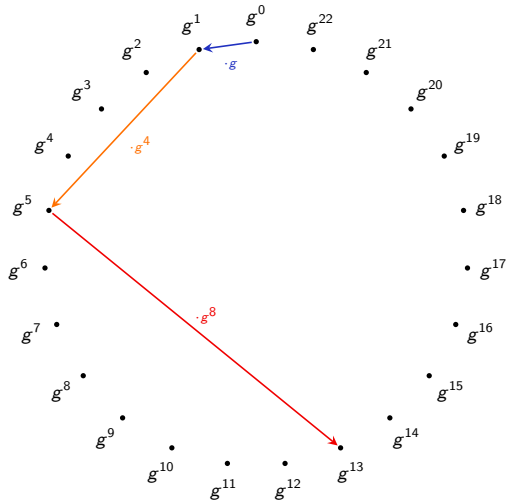
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply-and-square-and-multiply



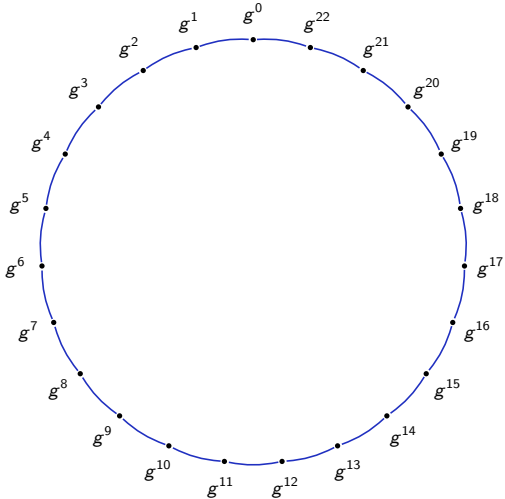
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply-and-square-and-multiply-and-square-and-multiply



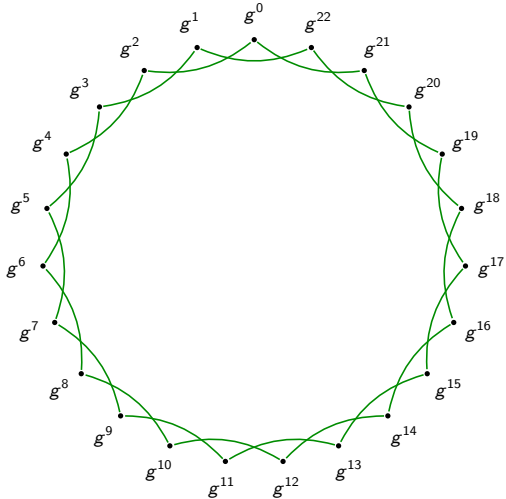
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



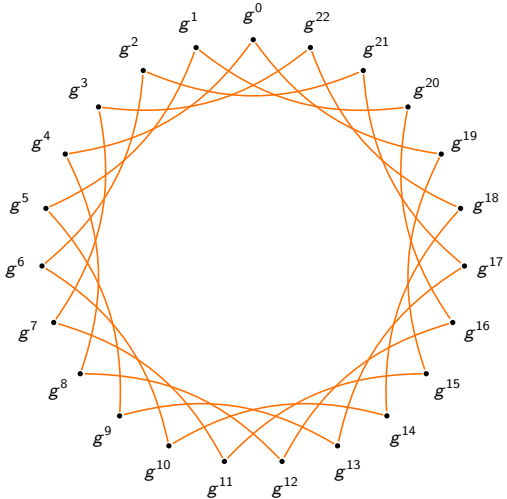
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



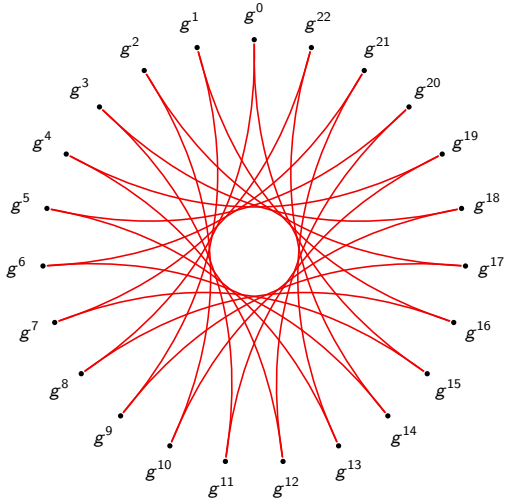
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



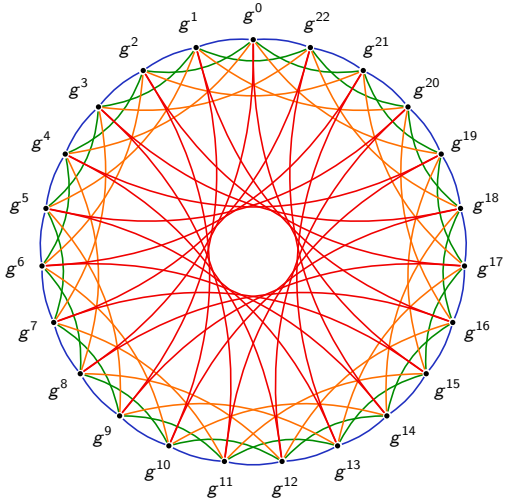
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as graphs



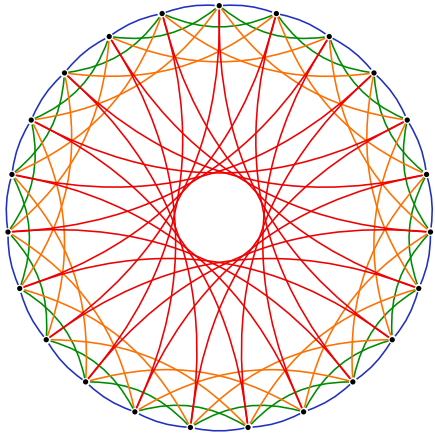
Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



Reminder: DH in group with $\#G = 23$. Bob computes g^{13}.

Square-and-multiply as a graph



Fast mixing: paths of length log(# nodes) to everywhere.

Exponential separation

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b *in any group* in polynomial time.

Exponential separation until quantum computers come

Constructive computation:
With square-and-multiply, applying b takes $\Theta(\log_2 \#G)$.

Attack costs:
For well-chosen groups, recovering b takes $\Theta(\sqrt{\#G})$.

(For less-well chosen groups the attacks are faster.)

As

$$\sqrt{\#G} = 2^{0.5 \log_2 \#G}$$

attacks are exponentially harder.

On a sufficiently large quantum computer, Shor's algorithm quantumly computes b from g^b *in any group* in polynomial time.
Isogeny graphs to the rescue!

What is an elliptic curve?

An elliptic curve is a smooth projective plane curve of genus one with at least one point.

What is an elliptic curve?

An elliptic curve is a smooth projective plane curve of genus one with at least one point.

This information together with the theorem of Riemann Roch is enough to derive that any elliptic curve admits an affine equation of the form

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

with $a_i \in k$, where k is the field where the point is defined.

This equation is the general form of a Weierstrass curve.

In algebraic geometry, smooth means that the curve does not have singularities.

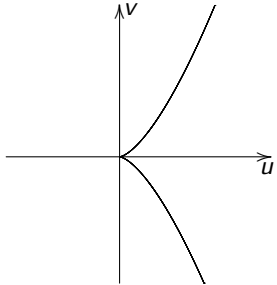
[The indices actually make sense if you give y weight 3, x weight 2 and ask that the weight + index equals 6.]

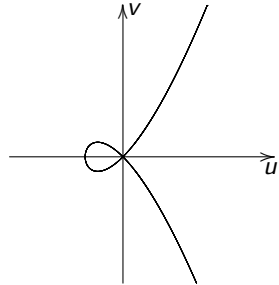
Singularities

Jacobi criterion:
A point $P = (x_P, y_P)$ on E is singular if (x, y) also satisfies the two partial derivatives, $2y + a_1x + a_3 = 0$ and $a_1y = 3x^2 + 2a_2x + a_4$.

A curve is non-singular (or smooth) if it does not have a singular point.

Note that "point on E" means that the point satisfies the curve equation. Note also that you need to check this for all points over any extension field of k.





Isomorphisms

An isomorphism is a map between elliptic curves that is defined everywhere, i.e., that is given by polynomials in x and y.

Valid transformations are those that keep the curve shape the same, so y^2 and x^3 are monic and no other degrees than in the long equation appear.

Isomorphisms

An isomorphism is a map between elliptic curves that is defined everywhere, i.e., that is given by polynomials in x and y.

Valid transformations are those that keep the curve shape the same, so y^2 and x^3 are monic and no other degrees than in the long equation appear. This means we can change $y \leftarrow \alpha^3 y + \beta x + \gamma, x \leftarrow \alpha^2 x + \delta$, and divide both sides by α^6.

For fields of characteristic larger than 3 we can transform this equation to one with fewer variables, called *short Weierstrass form*.

Isomorphisms

An isomorphism is a map between elliptic curves that is defined everywhere, i.e., that is given by polynomials in x and y.

Valid transformations are those that keep the curve shape the same, so y^2 and x^3 are monic and no other degrees than in the long equation appear. This means we can change $y \leftarrow \alpha^3 y + \beta x + \gamma, x \leftarrow \alpha^2 x + \delta$, and divide both sides by α^6.

For fields of characteristic larger than 3 we can transform this equation to one with fewer variables, called *short Weierstrass form*.

Our first target is to get rid of the $a_1xy + a_3y$ term. If the characteristic is not 2 we can use $y \leftarrow y - (a_1x + a_3)/2$ to reach the form $y^2 = x^3 + a'_2x^2 + a'_4x + a'_6$.

If the characteristic is not 3 we can similarly get rid of the a'_2x^2 term by using $x \leftarrow x - a'_2/3$.

The curve equation $y^2 = x^3 + c_4x + c_6$ is called short Weierstrass form.

Short Weierstrass form $y^2 = x^3 + c_4x + c_6$

A singularity exists if and only if the right hand side has a double root, i.e. if its discriminant is zero:

$$4c_4^3 + 27c_6^2 = 0.$$

Within this form the only isomorphisms possible are $y \leftarrow \alpha^3 y, x \leftarrow \alpha^2 x$, and divide both sides by α^6.

Short Weierstrass form $y^2 = x^3 + c_4x + c_6$

A singularity exists if and only if the right hand side has a double root, i.e. if its discriminant is zero:

$$4c_4^3 + 27c_6^2 = 0.$$

Within this form the only isomorphisms possible are $y \leftarrow \alpha^3 y, x \leftarrow \alpha^2 x$, and divide both sides by α^6. This gives $c_4' = c_4/\alpha^4$ and $c_6' = c_6/\alpha^6$.

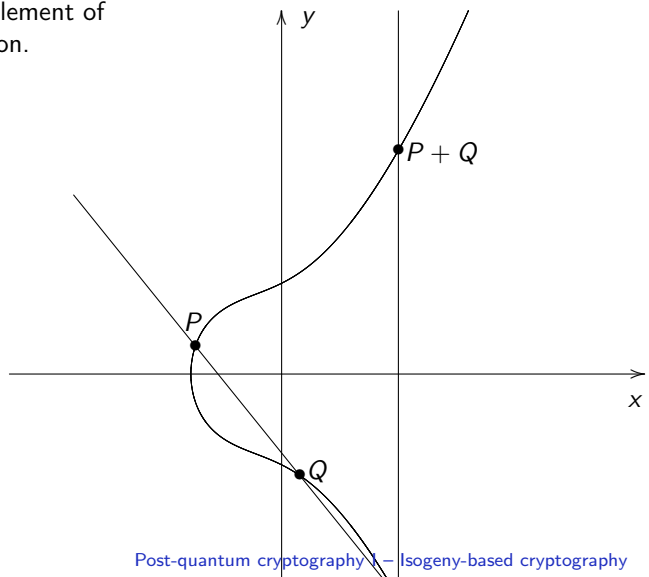
The j-invariant of a curve in short Weierstrass form is

$$j = 1728 \cdot 4c_4^3/(4c_4^3 + 27c_6^2).$$

This is invariant under isomorphisms.

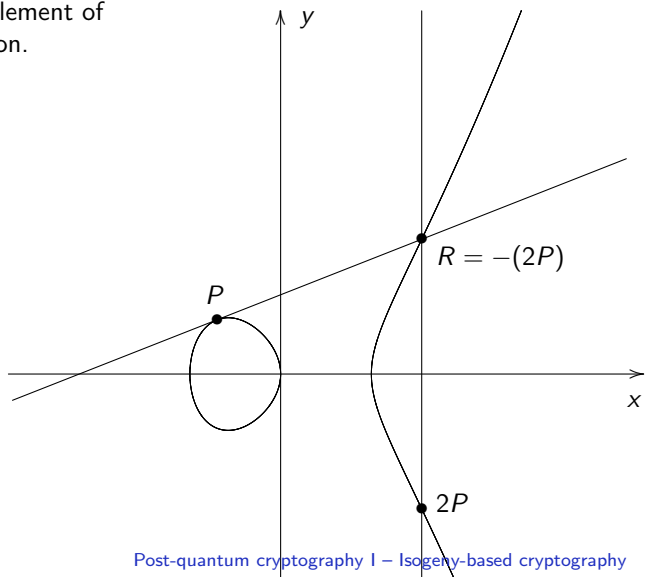
Addition law on the curve

Definition: If P, Q, R are on a line then $P + Q + R = \infty$.
∞ is the neutral element of this group operation.



Tangents to the curve and points with multiplicity

Definition: If P, Q, R are on a line then $P + Q + R = \infty$.
∞ is the neutral element of this group operation.



Montgomery curves

Montgomery curves are a special form of elliptic curves which can be written in the form

$$Bv^2 = u^3 + Au^2 + u.$$

This almost matches the Weierstrass equation given above and the addition law is very similar.

If $u_1 \neq u_2$ then $\lambda = (v_1 - v_2)/(u_1 - u_2)$;
if $u_1 = u_2$ and $v_1 = v_2 \neq 0$ then $\lambda = (3u_1^2 + 2Au_1 + 1)/(2Bv_1)$.
In both cases

$$u_3 = B\lambda^2 - A - u_1 - u_2, v_3 = \lambda(u_1 - u_3) - v_1$$

As on Weierstrass curves:
$-(u_1, v_1) = (u_1, -v_1)$ and ∞ is the neutral element.

Montgomery curves always have a point $(0, 0)$ of order 2 and at least one of the following

- $u^2 + Au + 1 = (u - u_1)(u - u_2)$, giving $(u_1, 0), (u_2, 0)$ of order 2;
- there is a point of order 4.

Hence, the group order is always divisible by 4.

See the EFD for more curve shapes and efficient formulas.

Isogenies

An *isogeny* of elliptic curves is a non-zero map $E \to E'$

- given by *rational functions*
- that is a *group homomorphism*.

The *degree* of a separable isogeny is the size of its *kernel*.

Isogenies

An *isogeny* of elliptic curves is a non-zero map $E \to E'$

- given by *rational functions*
- that is a *group homomorphism*.

The *degree* of a separable isogeny is the size of its *kernel*.

Example #1: For each $m \neq 0$, the *multiplication-by-m map*

$$[m]\colon E \to E$$

is an isogeny from E to itself.

If $m \neq 0$ in the base field, its kernel is

$$E[m] \;\cong\; \mathbb{Z}/m \times \mathbb{Z}/m.$$

Thus $[m]$ is a degree-m^2 isogeny.

Isogenies

An *isogeny* of elliptic curves is a non-zero map $E \to E'$

- given by *rational functions*
- that is a *group homomorphism*.

The *degree* of a separable isogeny is the size of its *kernel*.

Example #2: For any a and b, the map $\iota\colon (x,y) \mapsto (-x, \sqrt{-1}\cdot y)$ defines a degree-1 isogeny of the elliptic curves

$$\{y^2 = x^3 + ax + b\} \longrightarrow \{y^2 = x^3 + ax - b\}\,.$$

It is an *isomorphism*; its kernel is $\{\infty\}$.

Isogenies

An *isogeny* of elliptic curves is a non-zero map $E \to E'$
- given by *rational functions*
- that is a *group homomorphism*.

The *degree* of a separable isogeny is the size of its *kernel*.

Example #3:

$$(x, y) \mapsto \left(\frac{x^3-4x^2+30x-12}{(x-2)^2}, \frac{x^3-6x^2-14x+35}{(x-2)^3} \cdot y \right)$$

defines a degree-3 isogeny of the elliptic curves

$$\{y^2 = x^3 + x\} \longrightarrow \{y^2 = x^3 - 3x + 3\}$$

over $\mathbb{F}_{71}$. Its kernel is $\{(2, 9), (2, -9), \infty\}$.

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)
- We can *walk efficiently* on these graphs.

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)
- We can *walk efficiently* on these graphs.
- *Fast mixing*: short paths to (almost) all nodes.

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)
- We can *walk efficiently* on these graphs.
- *Fast mixing*: short paths to (almost) all nodes.
- *No efficient* algorithms to *recover paths* from endpoints.
  (*Both* classical and quantum!)

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)
- We can *walk efficiently* on these graphs.
- *Fast mixing*: short paths to (almost) all nodes.
- *No efficient* algorithms to *recover paths* from endpoints.
  (*Both* classical and quantum!)
- *Enough structure* to *navigate* the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths. More later.

Big picture

- Isogenies are a source of *exponentially*-sized *graphs*.
- Isogeny graph: Nodes are isomorphism classes of curves, edges are isogenies.
  (We usually care about *sub*graphs with certain properties.)
- We can *walk efficiently* on these graphs.
- *Fast mixing*: short paths to (almost) all nodes.
- *No efficient* algorithms to *recover paths* from endpoints.
  (*Both* classical and quantum!)
- *Enough structure* to *navigate* the graph meaningfully.
  That is: some *well-behaved* "directions" to describe paths. More later.

It is easy to construct graphs that satisfy *almost* all of these
not enough for crypto!


CSIDH ['siːˌsaɪd]
(Castryck, Lange, Martindale, Panny, Renes; 2018)

Why CSIDH?

- Closest thing we have in PQC to normal DH key exchange:
  Keys can be reused, blinded; no difference between initiator &responder.
- Public keys are represented by some $A \in \mathbb{F}_p$; p fixed prime.
- Alice computes and distributes her public key A.
  Bob computes and distributes his public key B.
- Alice and Bob do computations on each other's public keys
  to obtain shared secret.
- Fancy math: computations start on some elliptic curve

$$E_A : y^2 = x^3 + Ax^2 + x,$$

  use *isogenies* to move to a different curve.
- Computations need arithmetic (add, mult, div) modulo p and
  elliptic-curve computations.

CSIDH in one slide

- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.

CSIDH in one slide

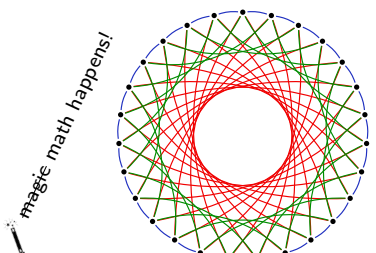
- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.

CSIDH in one slide

- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.

CSIDH in one slide

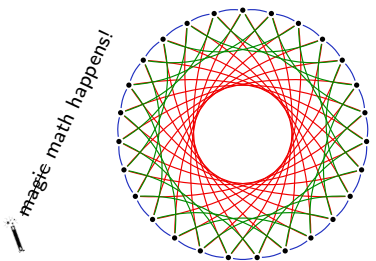
- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.



$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

CSIDH in one slide

- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.

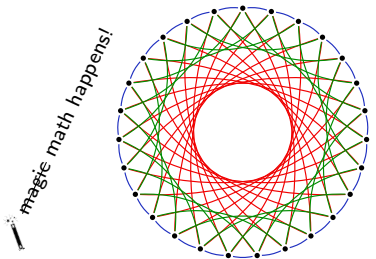


$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

- Walking "left" and "right" on any ℓ_i-subgraph is efficient.

CSIDH in one slide

- Choose some small odd primes $\ell_1, ..., \ell_n$.
- Make sure $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ is prime.
- Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ points$\}$.
- Look at the ℓ_i-isogenies defined over $\mathbb{F}_p$ within X.

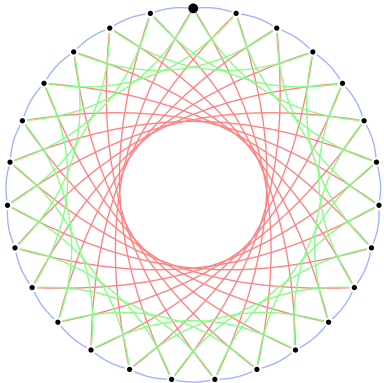


$p = 419$
$\ell_1 = 3$
$\ell_2 = 5$
$\ell_3 = 7$

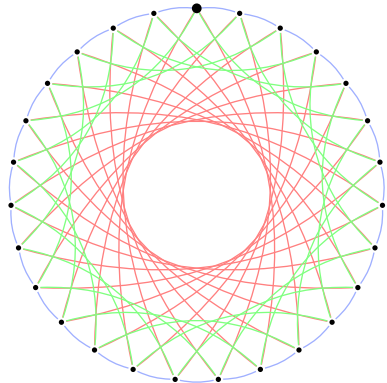
- Walking "left" and "right" on any ℓ_i-subgraph is efficient.
- We can represent $E \in X$ as a single coefficient $A \in \mathbb{F}_p$.

CSIDH key exchange

Alice
[+, +, −, −]



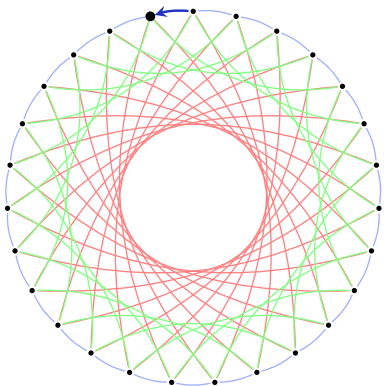
Bob
[−, +, −, −]



CSIDH key exchange

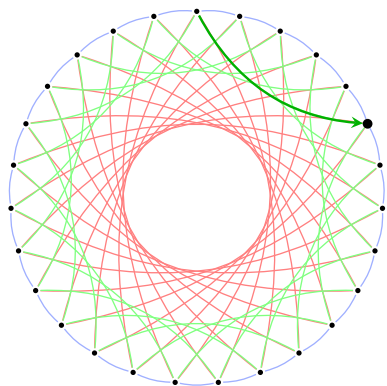
Alice

$[+, +, -, -]$



Bob

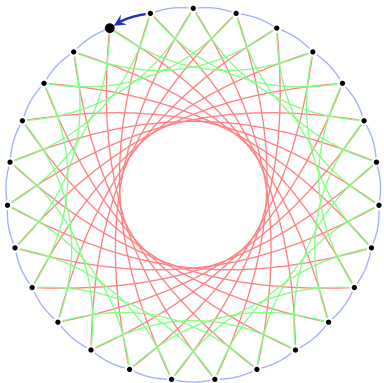
$[-, +, -, -]$



CSIDH key exchange

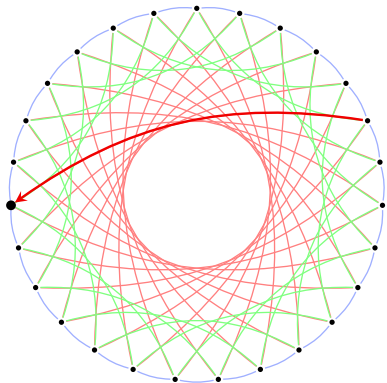
Alice

$[+, +, -, -]$



Bob

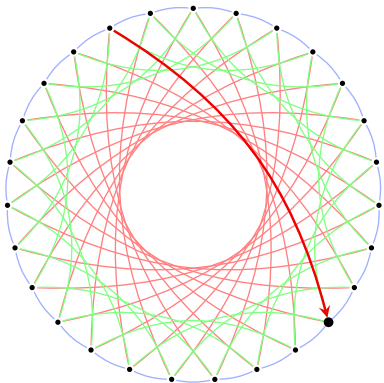
$[-, +, -, -]$



CSIDH key exchange

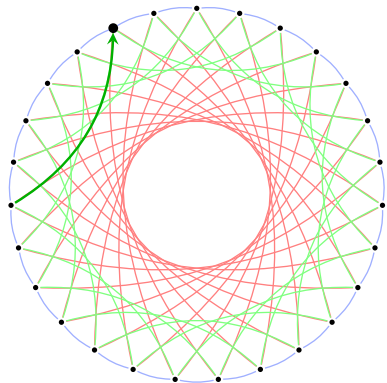
Alice

$[+, +, -, -]$



Bob

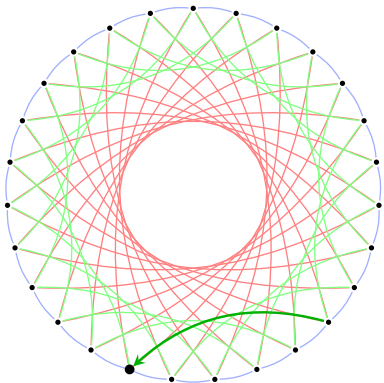
$[-, +, -, -]$



CSIDH key exchange

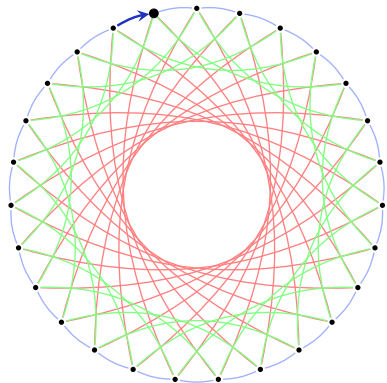
Alice

$[+, +, -, -]$



Bob

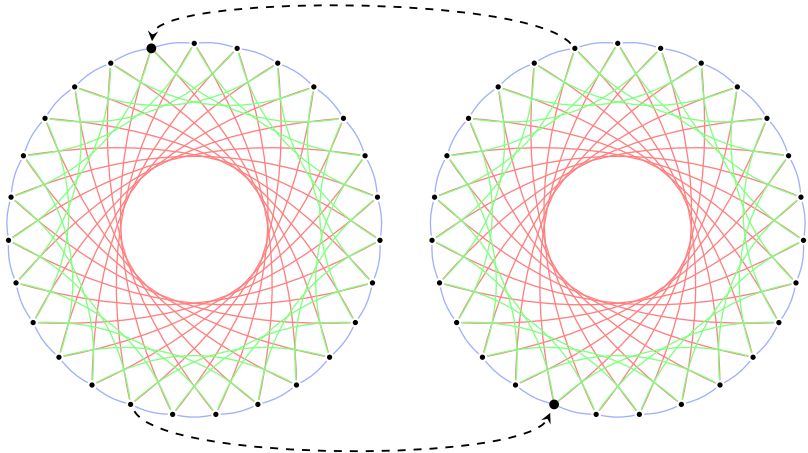
$[-, +, -, -]$



CSIDH key exchange

Alice
$[+,+,-,-]$

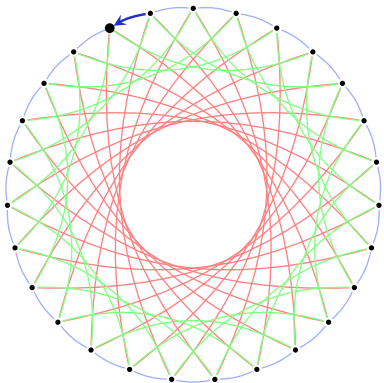
Bob
$[-,+,-,-]$



CSIDH key exchange

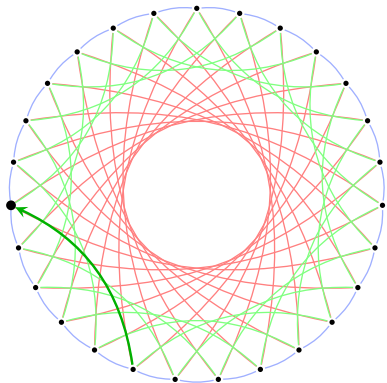
Alice

$[+, +, -, -]$



Bob

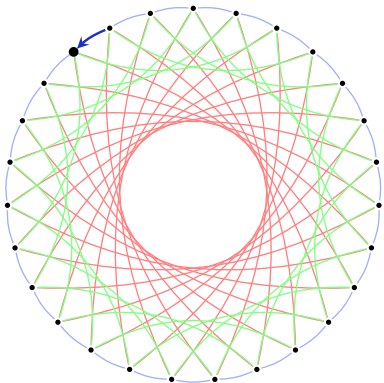
$[-, +, -, -]$



CSIDH key exchange

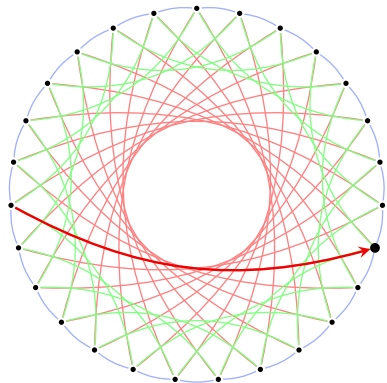
Alice

$[+, +, -, -]$



Bob

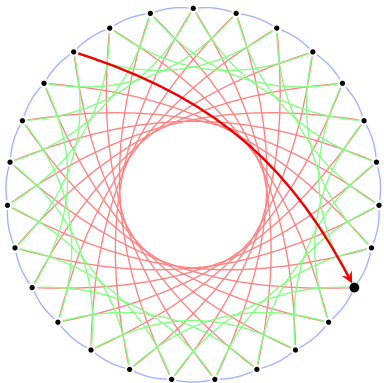
$[-, +, -, -]$



CSIDH key exchange

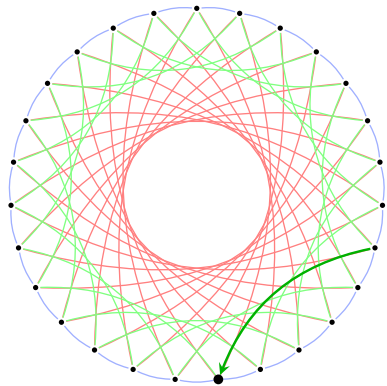
Alice

$[+, +, -, -]$



Bob

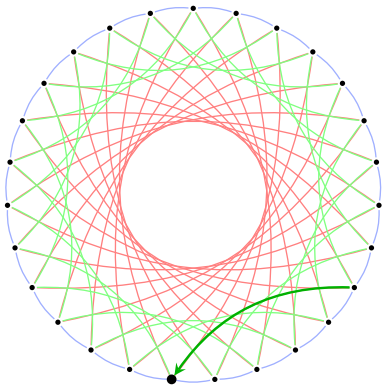
$[-, +, -, -]$



CSIDH key exchange

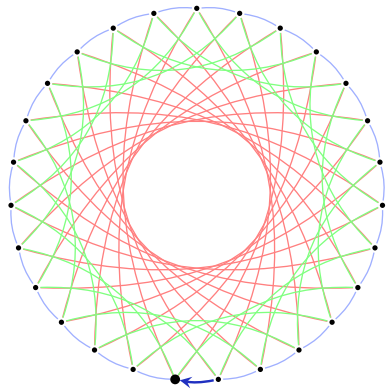
Alice

$[+, +, -, -]$



Bob

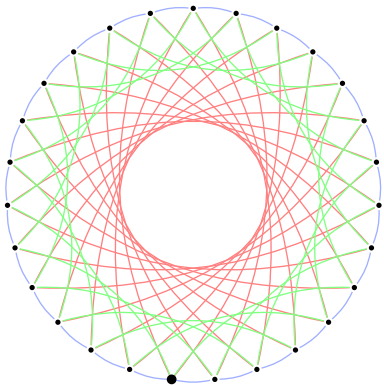
$[-, +, -, -]$



CSIDH key exchange

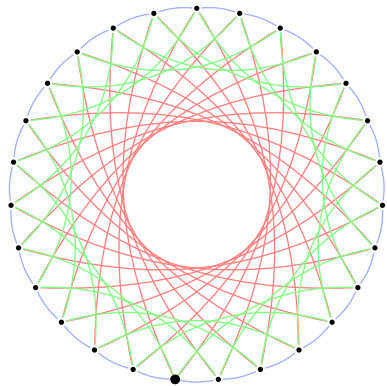
Alice

$[+, +, -, -]$



Bob

$[-, +, -, -]$



Walking in the CSIDH graph

For math details see the bonus slides at the end.

Taking a *"positive" step* on the ℓ_i*-subgraph*.

1. Find a point $(x, y) \in E$ of *order* ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the *isogeny* with *kernel* $\langle (x, y) \rangle$ using Vélu's formulas.

Walking in the CSIDH graph

For math details see the bonus slides at the end.

Taking a *"positive" step* on the ℓ_i*-subgraph*.

1. Find a point $(x, y) \in E$ of *order* ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p + 1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the *isogeny* with *kernel* $\langle (x, y) \rangle$ using Vélu's formulas.

Taking a *"negative" step* on the ℓ_i*-subgraph*.

1. Find a point $(x, y) \in E$ of *order* ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
   Same test as above to find such a point.
2. Compute the *isogeny* with *kernel* $\langle (x, y) \rangle$ using Vélu's formulas.

Walking in the CSIDH graph

For math details see the bonus slides at the end.

Taking a *"positive" step* on the ℓ_i*-subgraph*.

1. Find a point $(x, y) \in E$ of *order* ℓ_i with $x, y \in \mathbb{F}_p$.
   The order of any $(x, y) \in E$ divides $p + 1$, so $[(p+1)/\ell_i](x, y) = \infty$ or a point of order ℓ_i.
   Sample a new point if you get ∞.
2. Compute the *isogeny* with *kernel* $\langle (x, y) \rangle$ using Vélu's formulas.

Taking a *"negative" step* on the ℓ_i*-subgraph*.

1. Find a point $(x, y) \in E$ of *order* ℓ_i with $x \in \mathbb{F}_p$ but $y \notin \mathbb{F}_p$.
   Same test as above to find such a point.
2. Compute the *isogeny* with *kernel* $\langle (x, y) \rangle$ using Vélu's formulas.

Upshot: With "x-only' arithmetic" everything happens *over* $\mathbb{F}_p$.
$\implies$ *Efficient* to implement! There are several more speedups, such as pushing points through isogenies.

Abstract from Diffie-Hellman data flow

"CSIDH: an efficient post-quantum commutative group action"

Abstract from Diffie-Hellman data flow

"CSIDH: an efficient post-quantum commutative group action"

Cycles are *compatible*: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of *total* step *counts* for each ℓ_i.

Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

Abstract from Diffie-Hellman data flow

"CSIDH: an efficient post-quantum commutative group action"

Cycles are *compatible*: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of *total* step *counts* for each ℓ_i.

Example: $[+, +, -, -, -, +, -, -]$ just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

There is a *group action* of $(\mathbb{Z}^n, +)$ on our *set of curves* X!

Abstract from Diffie-Hellman data flow

"CSIDH: an efficient post-quantum commutative group action"

Cycles are *compatible*: [right then left] = [left then right]
$\rightsquigarrow$ only need to keep track of *total* step *counts* for each ℓ_i.

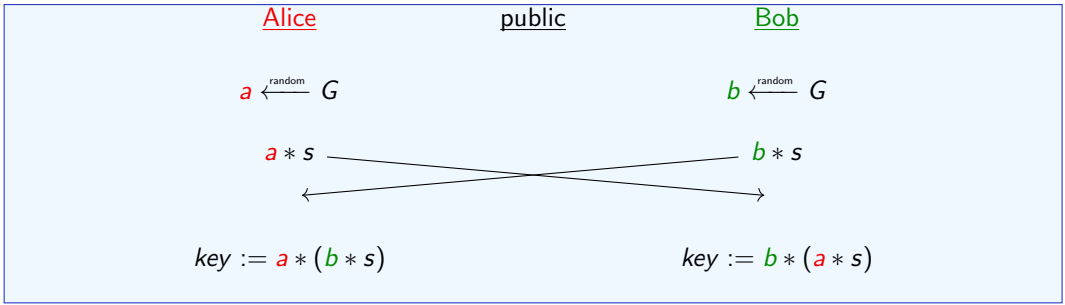
Example: [+, +, −, −, −, +, −, −] just becomes $(+1, \quad 0, -3) \in \mathbb{Z}^3$.

There is a *group action* of $(\mathbb{Z}^n, +)$ on our *set of curves* X!

Many paths are "useless". *Fun fact:* Quotienting out trivial actions yields the *ideal-class group* $\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$.

Cryptographic group actions

Like in the CSIDH example, we *generally* get a DH-like key exchange from a commutative *group action* $G \times S \to S$:



Why no Shor?

Shor computes α from $h = g^\alpha$ by finding the kernel of the map

$$f\colon\ \mathbb{Z}^2 \to G,\ (x,y) \mapsto g^x \underset{\uparrow}{\cdot} h^y$$

For general group actions, we *cannot compose* $x * s$ *and* $y * (b * s)$.

For CSIDH this would require composing two elliptic curves in some form compatible with the action of G.

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. *Evaluate* the group action many times. ("oracle calls")
2. *Combine* the results in a certain way. ("sieving")

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. *Evaluate* the group action many times. ("oracle calls")
2. *Combine* the results in a certain way. ("sieving")

- The algorithm admits many different *tradeoffs*.
- Oracle calls are *expensive*.
- The sieving phase has *classical and quantum* operations.

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. *Evaluate* the group action many times. ("oracle calls")
2. *Combine* the results in a certain way. ("sieving")

- The algorithm admits many different *tradeoffs*.
- Oracle calls are *expensive*.
- The sieving phase has *classical and quantum* operations.

How to compare costs?
(Is one qubit operation $\approx$ one bit operation? a hundred? millions?)

CSIDH vs. Kuperberg

Kuperberg's algorithm consists of two components:

1. *Evaluate* the group action many times. ("oracle calls")
2. *Combine* the results in a certain way. ("sieving")

- The algorithm admits many different *tradeoffs*.
- Oracle calls are *expensive*.
- The sieving phase has *classical and quantum* operations.

  How to compare costs?
  (Is one qubit operation $\approx$ one bit operation? a hundred? millions?)

$\Longrightarrow$ It is still rather unclear how to choose CSIDH parameters.

...but all known attacks cost $\exp\big((\log p)^{1/2+o(1)}\big)$!
Recent improvements to sieving target the $o(1)$.

Kuperberg applies to *all* commutative group actions.

CSIDH security

Core problem:
Given $E, E' \in X$, *find* and compute *isogeny* $E \to E'$.

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
  (More precisely $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ keys.)

Without quantum computer:

- Meet-in-the-middle variants: Time $O(\sqrt[4]{p})$.
  (2016 Delfs–Galbraith)

CSIDH security

Core problem:
Given $E, E' \in X$, *find* and compute *isogeny* $E \to E'$.

Size of key space:

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
  (More precisely $\#\mathrm{cl}(\mathbb{Z}[\sqrt{-p}])$ keys.)

Without quantum computer:

- Meet-in-the-middle variants: Time $O(\sqrt[4]{p})$.
  (2016 Delfs–Galbraith)

With quantum computer:

- 2014 Childs–Jao–Soukharev: Kuperberg applies to family
  - These have subexponential complexity.
  - Not vulnerable to Shor's attack.

CSIDH security:

- Public-key validation:
  Quickly check that $E_A : y^2 = x^3 + Ax^2 + x$ has $p + 1$ points.

CSIDH-512 https://csidh.isogeny.org/

Definition:

- $p = 4\prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \ldots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes (just one $\mathbb{F}_p$ element).

Performance on typical Intel Skylake laptop core:

- Clock cycles: about $12 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.

CSIDH-512 https://csidh.isogeny.org/

Definition:

- $p = 4\prod_{i=1}^{74} \ell_i - 1$ with $\ell_1, \ldots, \ell_{73}$ first 73 odd primes. $\ell_{74} = 587$.
- Exponents $-5 \leq e_i \leq 5$ for all $1 \leq i \leq 74$.

Sizes:

- Private keys: 32 bytes. (37 in current software for simplicity.)
- Public keys: 64 bytes (just one $\mathbb{F}_p$ element).

Performance on typical Intel Skylake laptop core:

- Clock cycles: about $12 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.
- Post-quantum: complicated.
  Recent work analyzing cost: see https://quantum.isogeny.org.
  Several papers analyzing quantum approaches. (2018 Biasse–Iezzi-Jacobson, 2018-2020 Bonnetain–Schrottenloher, 2020 Peikert)
  https://csidh.isogeny.org/analysis.html

Further information (also on PQC in general)

- YouTube channel Tanja Lange: Post-quantum cryptography.
- Isogeny-based cryptography school.
- https://2017.pqcrypto.org/school: PQCRYPTO summer school with 21 lectures on video, slides, and exercises.
- https://2017.pqcrypto.org/exec and https://pqcschool.org/index.html: Executive school (less math, more perspective).
- https://pqcrypto.org our overview page.
- ENISA report on PQC, co-authored.
- https://pqcrypto.eu.org: PQCRYPTO EU Project.
  - PQCRYPTO recommendations.
  - Free software libraries (libpqcrypto, pqm4, pqhw).
  - Many reports, scientific articles, (overview) talks.
- Quantum Threat Timeline from Global Risk Institute, 2019; 2021 update.
- Status of quantum computer development (by German BSI).
- NIST PQC competition.
- PQCrypto 2016, PQCrypto 2017, PQCrypto 2018, PQCrypto 2019, PQCrypto 2020, PQCrypto 2021, PQCrypto 2022 with many slides and videos online.

Math details

Isogenies and endomorphism rings

An *isogeny* of elliptic curves is a non-zero map $\varphi : E \to E'$

- given by *rational functions*
- that is a *group homomorphism*.

The *degree* d of a separable isogeny is the size of its *kernel* $d = \ker(\varphi)$.

For isogeny $\varphi : E \to E'$ there exists a unique *dual isogeny* $\hat{\varphi} : E' \to E$.

The composition $\hat{\varphi} \circ \varphi$ is the multiplication-by-d map on E and $\varphi \circ \hat{\varphi}$ the multiplication-by-d map on E', where $d = \deg(\varphi) = \deg(\hat{\varphi})$.

An *endomorphism* is an isogeny from a curve E to itself.

The set of endomorphisms forms a ring $\mathrm{End}(E)$ under $+$ and $\circ$.

The ring of k-rational endomorphisms of E/k is denoted $\mathrm{End}_k(E)$.

Elliptic curves over finite fields

We now focus on curves over finite fields $\mathbb{F}_q, q = p^k$.

There are only finitely many pairs (x, y) that can satisfy the curve equation, thus there are only finitely many points on $E(\mathbb{F}_q)$.

Elliptic curves over finite fields

We now focus on curves over finite fields $\mathbb{F}_q, q = p^k$.

There are only finitely many pairs (x, y) that can satisfy the curve equation, thus there are only finitely many points on $E(\mathbb{F}_q)$.

Hasse Interval:

$$\#E(\mathbb{F}_q) \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$$

Elliptic curves over finite fields

We now focus on curves over finite fields $\mathbb{F}_q, q = p^k$.

There are only finitely many pairs (x, y) that can satisfy the curve equation, thus there are only finitely many points on $E(\mathbb{F}_q)$.

Hasse Interval:

$$\#E(\mathbb{F}_q) \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$$

The following are equivalent definitions of *supersingular* curves:

- $\#E(\mathbb{F}_q) = q + 1 - t$ with $t \equiv 0 \bmod p$.
- $E[p] = \{\infty\}$.
  Note that $E[n] = \{P \in E(\overline{\mathbb{F}_p}) \mid nP = \infty\}$.

Elliptic curves over finite fields

We now focus on curves over finite fields $\mathbb{F}_q, q = p^k$.

There are only finitely many pairs (x, y) that can satisfy the curve equation, thus there are only finitely many points on $E(\mathbb{F}_q)$.

Hasse Interval:

$$\#E(\mathbb{F}_q) \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$$

The following are equivalent definitions of *supersingular* curves:

- $\#E(\mathbb{F}_q) = q + 1 - t$ with $t \equiv 0 \bmod p$.
- $E[p] = \{\infty\}$.
  Note that $E[n] = \{P \in E(\overline{\mathbb{F}_p}) \mid nP = \infty\}$.

For $p > 3$ the only $t \in [-2\sqrt{p}, 2\sqrt{p}]$ with $t \equiv 0 \bmod p$ is $t = 0$.

Elliptic curves over finite fields

We now focus on curves over finite fields $\mathbb{F}_q, q = p^k$.

There are only finitely many pairs (x, y) that can satisfy the curve equation, thus there are only finitely many points on $E(\mathbb{F}_q)$.

Hasse Interval:

$$\#E(\mathbb{F}_q) \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$$

The following are equivalent definitions of *supersingular* curves:

- $\#E(\mathbb{F}_q) = q + 1 - t$ with $t \equiv 0 \bmod p$.
- $E[p] = \{\infty\}$.
  Note that $E[n] = \{P \in E(\overline{\mathbb{F}_p}) \mid nP = \infty\}$.

For $p > 3$ the only $t \in [-2\sqrt{p}, 2\sqrt{p}]$ with $t \equiv 0 \bmod p$ is $t = 0$.

Thus $\#E(\mathbb{F}_p) = p + 1$, $\#E(\mathbb{F}_{p^2}) \in \{(p-1)^2, p^2 + 1, (p+1)^2\}$
for supersingular curves and $p > 3$.

Quadratic twists

E'/k is a *twist* of elliptic curve E/k if E' is isomorphic to E over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$ $E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a *quadratic twist*.

Quadratic twists

E'/k is a *twist* of elliptic curve E/k if E' is isomorphic to E over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$ $E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a *quadratic twist*.

E' is not in Weierstrass form (does not have the right shape).
E' is isomorphic to $E'' : y^2 = x^3 - Ax^2 + x$ via $(x, y) \mapsto (-x, y)$ over $\mathbb{F}_p$.

Quadratic twists

E'/k is a *twist* of elliptic curve E/k if E' is isomorphic to E over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$ $E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a *quadratic twist*.

E' is not in Weierstrass form (does not have the right shape).
E' is isomorphic to $E'' : y^2 = x^3 - Ax^2 + x$ via $(x, y) \mapsto (-x, y)$ over $\mathbb{F}_p$.

Each $x \in \mathbb{F}_p$ satisfies one of

- $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$, thus there are two points $(x, \pm\sqrt{x^3 + Ax^2 + x})$ in $E(\mathbb{F}_p)$.
- $x^3 + Ax^2 + x$ is not a square in $\mathbb{F}_p$, thus there are two points $(x, \pm\sqrt{-(x^3 + Ax^2 + x)})$ in $E'(\mathbb{F}_p)$.
- $x^3 + Ax^2 + x = 0$, thus $(x, 0)$ is a point in $E(\mathbb{F}_p)$ and in $E'(\mathbb{F}_p)$.

Quadratic twists

E'/k is a *twist* of elliptic curve E/k if E' is isomorphic to E over $\bar{k}$.

For $E : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$ $E' : -y^2 = x^3 + Ax^2 + x$ is isomorphic to E via

$$(x, y) \mapsto (x, \sqrt{-1}y).$$

This map is defined over $\mathbb{F}_{p^2}$, so this is a *quadratic twist*.

E' is not in Weierstrass form (does not have the right shape).
E' is isomorphic to $E'' : y^2 = x^3 - Ax^2 + x$ via $(x, y) \mapsto (-x, y)$ over $\mathbb{F}_p$.

Each $x \in \mathbb{F}_p$ satisfies one of

- $x^3 + Ax^2 + x$ is a square in $\mathbb{F}_p$, thus there are two points $(x, \pm\sqrt{x^3 + Ax^2 + x})$ in $E(\mathbb{F}_p)$.
- $x^3 + Ax^2 + x$ is not a square in $\mathbb{F}_p$, thus there are two points $(x, \pm\sqrt{-(x^3 + Ax^2 + x)})$ in $E'(\mathbb{F}_p)$.
- $x^3 + Ax^2 + x = 0$, thus $(x, 0)$ is a point in $E(\mathbb{F}_p)$ and in $E'(\mathbb{F}_p)$.

$\#E(\mathbb{F}_p) + \#E'(\mathbb{F}_p) = 2p + 2$, thus
$\#E(\mathbb{F}_p) = p + 1 - t$ implies $\#E'(\mathbb{F}_p) = p + 1 + t$.

Isogenies and kernels

For any *finite* subgroup G of E, there exists a *unique*[1] separable isogeny $\varphi_G : E \to E'$ with *kernel* G.

The curve E' is called E/G. ($\approx$ quotient groups)

If G is defined over k, then φ_G and E/G are also *defined over* k.

[1] (up to isomorphism of E')

Isogenies and kernels

For any *finite* subgroup G of E, there exists a *unique*[1]
separable isogeny $\varphi_G : E \to E'$ with *kernel* G.

The curve E' is called E/G. ($\approx$ quotient groups)

If G is defined over k, then φ_G and E/G are also *defined over* k.

Vélu '71:
Formulas for *computing* E/G and *evaluating* φ_G at a point.

Complexity: $\Theta(\#G) \rightsquigarrow$ only suitable for *small degrees*.

[1] (up to isomorphism of E')

Isogenies and kernels

For any *finite* subgroup G of E, there exists a *unique*[1] separable isogeny $\varphi_G : E \to E'$ with *kernel* G.

The curve E' is called E/G. ($\approx$ quotient groups)

If G is defined over k, then φ_G and E/G are also *defined over* k.

Vélu '71:
Formulas for *computing* E/G and *evaluating* φ_G at a point.

Complexity: $\Theta(\#G) \rightsquigarrow$ only suitable for *small degrees*.

Vélu operates in the field where the *points* in G live.

$\rightsquigarrow$ need to make sure extensions stay small for desired G
$\rightsquigarrow$ this is why we use *special* p and curves with $p+1$ *points*!

Not all k-rational points of E/G are in the image of k-rational points on E; but $\#E(k) = \#((E/G)(k))$.

[1] (up to isomorphism of E')

Vélu's formulas

Let P have odd prime order ℓ on E_A.
For $1 \leq i < \ell$ let x_i be the x-coordinate of iP.
Let

$$\tau = \prod_{i=1}^{\ell-1} x_i, \quad \sigma = \sum_{i=1}^{\ell-1} \left(x_i - \frac{1}{x_i} \right), \quad f(x) = x \prod_{i=1}^{\ell-1} \frac{xx_i - 1}{x - x_i}.$$

Then the ℓ-isogeny with kernel $\langle P \rangle$ is given by

$$\varphi_\ell : E_A \to E_B, (x, y) \mapsto (f(x), c_0 y f'(x))$$

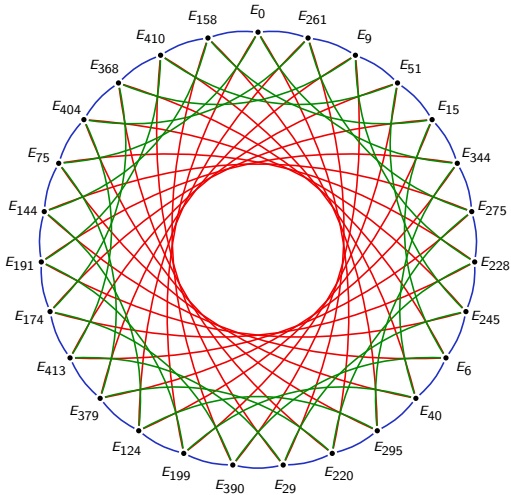
where $B = \tau(A - 3\sigma)$, and $c_0^2 = \tau$.

Main operation is to compute the x_i, just some elliptic-curve additions.
Note that $(\ell - i)P = -iP$ and both have the same x-coordinate.

Implementations often use *projective* formulas to avoid (or delay) inversions.

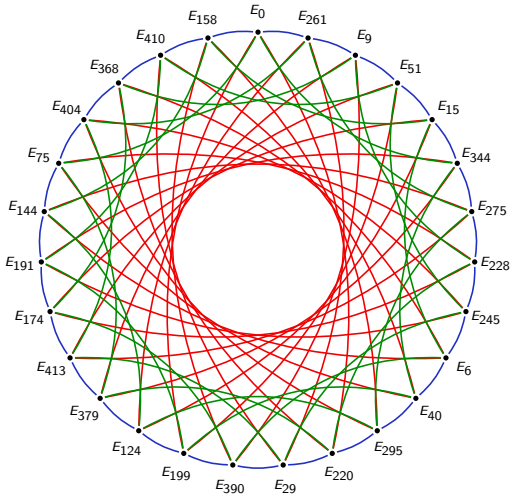
Montgomery curves have efficient arithmetic using only x-coordinates.

Graphs of elliptic curves



Nodes: Supersingular elliptic curves $E_A\colon y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.

Graphs of elliptic curves



Nodes: Supersingular elliptic curves $E_A\colon y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.

Each E_A on the left has E_{-A} on the right.

Negative direction means: flip to twist, go positive direction, flip back.

Class groups for supersingular curves over $\mathbb{F}_p$

Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ *points*$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\iff$ action of $\mathfrak{l}_i$ on X.

Class groups for supersingular curves over $\mathbb{F}_p$

Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ *points*$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\iff$ action of $\mathfrak{l}_i$ on X.

More precisely:
Subgroup corresponding to $\mathfrak{l}_i$ is $E[\mathfrak{l}_i] = E(\mathbb{F}_p)[\ell_i]$.
(Note that $\ker(\pi - 1)$ is just the $\mathbb{F}_p$-rational points!)

Subgroup corresponding to $\overline{\mathfrak{l}_i}$ is

$$E[\overline{\mathfrak{l}_i}] = \{P \in E[\ell_i] \mid \pi(P) = -P\}.$$

Class groups for supersingular curves over $\mathbb{F}_p$

Let $X = \{y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_p$ with $p+1$ *points*$\}$.
All curves in X have $\mathbb{F}_p$-endomorphism ring $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$.

Let π the Frobenius endomorphism. Ideal in $\mathcal{O}$ above ℓ_i.

$$\mathfrak{l}_i = (\ell_i, \pi - 1).$$

Moving $+$ in X with ℓ_i isogeny $\iff$ action of $\mathfrak{l}_i$ on X.

More precisely:
Subgroup corresponding to $\mathfrak{l}_i$ is $E[\mathfrak{l}_i] = E(\mathbb{F}_p)[\ell_i]$.
(Note that $\ker(\pi - 1)$ is just the $\mathbb{F}_p$-rational points!)

Subgroup corresponding to $\overline{\mathfrak{l}_i}$ is

$$E[\overline{\mathfrak{l}_i}] = \{P \in E[\ell_i] \mid \pi(P) = -P\}.$$

For supersingular Montgomery curves over $\mathbb{F}_p, p \equiv 3 \bmod 4$

$$E[\overline{\mathfrak{l}_i}] = \{(x, y) \in E[\ell_i] \mid x \in \mathbb{F}_p;\ y \notin \mathbb{F}_p\} \cup \{\infty\}.$$

Commutative group action

$\mathrm{cl}(\mathcal{O})$ acts on $X = \{y^2 = x^3 + Ax^2 + x \text{ over } \mathbb{F}_p \text{ with } p+1 \textit{ points}\}$.
For most ideal classes the kernel is big and formulas are expensive to compute.

$$I = \mathfrak{l}_1^{10}\mathfrak{l}_2^{-7}\mathfrak{l}_3^{27}$$

is a "big" ideal, but we can compute the action iteratively.

$\mathrm{cl}(\mathcal{O})$ is commutative[2] so we get a commutative group action..

The choice for CSIDH:
Let $K = \{[\mathfrak{l}_1^{e_1} \cdots \mathfrak{l}_n^{e_n}] \mid (e_1, ..., e_n) \text{ is 'short'}\} \subseteq \mathrm{cl}(\mathcal{O})$.
The action of K on X is very *efficient*!
Pick K as the keyspace

[2] Important to use the $\mathbb{F}_p$-endomorphism ring.