Elliptic-curve and isogeny-based cryptography

Chloe Martindale

University of Bristol

Summer School on real-world crypto and privacy
Sibenik 2022

Why elliptic-curve cryptography (ECC)?

ECC is widely deployed across many use cases. Why? It is:

- Low memory
- Fast
- Flexible
  - TLS, AKE, Signal protocol, IBE (using pairings), . . .
- Robust

Ex: WhatsApp (uses Signal protocol)

Public Key Types

- `Identity Key Pair` – A long-term Curve25519 key pair, generated at install time.
- `Signed Pre Key` – A medium-term Curve25519 key pair, generated at install time, signed by the `Identity Key`, and rotated on a periodic timed basis.
- `One-Time Pre Keys` – A queue of Curve25519 key pairs for one time use, generated at install time, and replenished as needed.

Session Key Types

- `Root Key` – A 32-byte value that is used to create `Chain Keys`.
- `Chain Key` – A 32-byte value that is used to create `Message Keys`.
- `Message Key` – An 80-byte value that is used to encrypt message contents. 32 bytes are used for an AES-256 key, 32 bytes for a HMAC-SHA256 key, and 16 bytes for an IV.

What is a pairing?

Pairings are maps of groups.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.

- If $g \in \mathbb{G}$ and $n \in \mathbb{Z}_{\geq 0}$, write $g^n = \underbrace{g * \cdots * g}_{n \text{ times}}$.

What is a pairing?

Pairings are maps of groups.

- A group $\mathbb{G}$ comes with a group operation $*$.
  - eg. $\mathbb{G} = \mathbb{Z}/p\mathbb{Z} - \{0\}$ with $*$ given by multiplication.

- If $g \in \mathbb{G}$ and $n \in \mathbb{Z}_{\geq 0}$, write $g^n = \underbrace{g * \cdots * g}_{n \text{ times}}$.
  - eg. $(3 \pmod 5)^2 = 3 \cdot 3 \pmod 5$.

What is a pairing?

Pairings are bilinear maps of groups.

What is a pairing?

Pairings are bilinear maps of groups. In particular:

$$\begin{aligned} \mathbb{G}_1 \times \mathbb{G}_2 &\rightarrow \mathbb{G}_3 \\ (g, h) &\mapsto P(g, h) \end{aligned}$$

What is a pairing?

Pairings are bilinear maps of groups. In particular:

$$
\begin{array}{ccc}
\mathbb{G}_1 \times \mathbb{G}_2 & \rightarrow & \mathbb{G}_3 \\
(g, h) & \mapsto & P(g, h) \\
(g^a, h^b) & \mapsto & P(g, h)^{ab}
\end{array}
$$

What is a pairing?

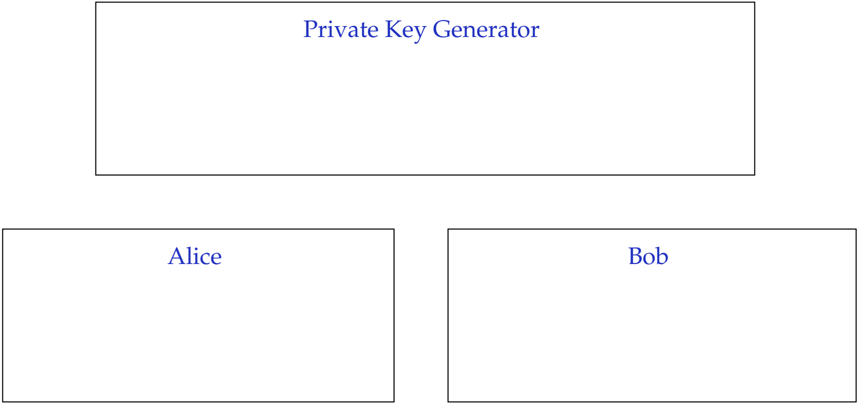
Pairings are bilinear maps of groups. In particular:

$$\begin{array}{ccc} \mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_3 \\ (g, h) & \mapsto & P(g, h) \\ (g^a, h^b) & \mapsto & P(g, h)^{ab} \end{array}$$

Why is this useful?

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.



Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

$$\text{Use a pairing } P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.
Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;

Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.

Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...

Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...

Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;

Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.

Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...

Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...

Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG

Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.
Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...
Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...
Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG
Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice
Compute $\mathsf{ver} = P(\mathsf{sk\text{-}b}, \mathsf{pub}^r)$
Verify that $\mathsf{ver} = \mathsf{enc\text{-}id\text{-}a}^r$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\textsf{id-a} \in \mathbb{G}_1$; Public $\textsf{pub} \in \mathbb{G}_2$;
Master secret key $\textsf{sk-m} \in \mathbb{Z}$; Master public key $\textsf{pk-m} = \textsf{pub}^{\textsf{sk-m} \in \mathbb{G}_2}$.
Computes $\textsf{sk-b} = \textsf{id-a}^{\textsf{sk-m}}$...
Sends $\textsf{sk-b}$ to Bob

Alice

Secret identity $\textsf{id-a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\textsf{enc-id-a} = P(\textsf{id-a}, \textsf{pk-m})$...
Sends $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ to Bob

Bob

Receives secret key $\textsf{sk-b} \in \mathbb{G}_1$ from PKG
Receives $(\textsf{pub}^r, \textsf{enc-id-a}^r)$ from Alice
Compute $\textsf{ver} = P(\textsf{sk-b}, \textsf{pub}^r)$
Verify that $\textsf{ver} = \textsf{enc-id-a}^r$ †

† Bilinearity:

$P(\textsf{sk-b}, \textsf{pub}^r) = P(\textsf{id-a}^{\textsf{sk-m}, \textsf{pub}^r})$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;

Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.

Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...

Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...

Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG

Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice

Compute $\mathsf{ver} = P(\mathsf{sk\text{-}b}, \mathsf{pub}^r)$

Verify that $\mathsf{ver} = \mathsf{enc\text{-}id\text{-}a}^r$ †

† Bilinearity:

$$P(\mathsf{sk\text{-}b}, \mathsf{pub}^r) = P(\mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}, \mathsf{pub}^r}) = P(\mathsf{id\text{-}a}, \mathsf{pub})^{\mathsf{sk\text{-}m} \cdot r}$$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;

Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.

Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...

Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$

Choose random $r \in \mathbb{Z}$...

Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...

Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG

Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice

Compute $\mathsf{ver} = P(\mathsf{sk\text{-}b}, \mathsf{pub}^r)$

Verify that $\mathsf{ver} = \mathsf{enc\text{-}id\text{-}a}^r$ †

† Bilinearity:

$$P(\mathsf{sk\text{-}b}, \mathsf{pub}^r) = P(\mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}, \mathsf{pub}^r}) = P(\mathsf{id\text{-}a}, \mathsf{pub})^{\mathsf{sk\text{-}m} \cdot r} = P(\mathsf{id\text{-}a}, \mathsf{pub}^{\mathsf{sk\text{-}m})^r}$$

Pairings in (simplified) IBE (Boneh-Franklin)

Scenario: Bob authenticates an anonymous Alice.

Use a pairing $P : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$

Private Key Generator

Alice's secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$; Public $\mathsf{pub} \in \mathbb{G}_2$;
Master secret key $\mathsf{sk\text{-}m} \in \mathbb{Z}$; Master public key $\mathsf{pk\text{-}m} = \mathsf{pub}^{\mathsf{sk\text{-}m} \in \mathbb{G}_2}$.
Computes $\mathsf{sk\text{-}b} = \mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}}$...
Sends $\mathsf{sk\text{-}b}$ to Bob

Alice

Secret identity $\mathsf{id\text{-}a} \in \mathbb{G}_1$
Choose random $r \in \mathbb{Z}$...
Compute $\mathsf{enc\text{-}id\text{-}a} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})$...
Sends $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ to Bob

Bob

Receives secret key $\mathsf{sk\text{-}b} \in \mathbb{G}_1$ from PKG
Receives $(\mathsf{pub}^r, \mathsf{enc\text{-}id\text{-}a}^r)$ from Alice
Compute $\mathsf{ver} = P(\mathsf{sk\text{-}b}, \mathsf{pub}^r)$
Verify that $\mathsf{ver} = \mathsf{enc\text{-}id\text{-}a}^r$ †

† Bilinearity:

$$P(\mathsf{sk\text{-}b}, \mathsf{pub}^r) = P(\mathsf{id\text{-}a}^{\mathsf{sk\text{-}m}, \mathsf{pub}^r}) = P(\mathsf{id\text{-}a}, \mathsf{pub})^{\mathsf{sk\text{-}m} \cdot r} = P(\mathsf{id\text{-}a}, \mathsf{pub}^{\mathsf{sk\text{-}m})^r} = P(\mathsf{id\text{-}a}, \mathsf{pk\text{-}m})^r.$$

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.
- Fast pairing computation.

What is a cryptographic pairing?

For this protocol idea to be useful, we need:

- Fast exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$. Examples:
  - Unit groups of finite fields (square-and-multiply).
  - Elliptic curve groups (double-and-add).
- Hard discrete logarithms problems in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.
  - Bilinearity of $P \rightsquigarrow$ complexity of DLP in each of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ is the fastest algorithm for solving DLP in any of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_3$.
- An explicit pairing formula.
  - Example: the Weil pairing with $\mathbb{G}_1$ and $\mathbb{G}_2$ as elliptic curve groups and $\mathbb{G}_3$ as a finite field group.
- Fast pairing computation.
  - Instances of the Weil pairing can be efficiently computed with Miller's algorithm.

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

$g \in \mathbb{F}_{p^k}^*$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Index calculus+** $\longrightarrow n$

Complexity:
- if p large, non-special, and k small, $L_{p^k}(1/2, c)$
- for most pairing instances, p is special, giving $L_{p^k}(1/3, c)$

How hard is the discrete logarithm problem?

$g \in \mathbb{G}$ (any): $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **DLP solver** $\longrightarrow n$

Complexity: depends on $\mathbb{G}$

$g \in \mathbb{F}_{p^k}^*$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Index calculus+** $\longrightarrow n$

Complexity: if p large, non-special, and k small, $L_{p^k}(1/2, c)$
for most pairing instances, p is special, giving $L_{p^k}(1/3, c)$

$g \in E(\mathbb{F}_{p^k})$: $\underbrace{g * \cdots * g}_{n \text{ times}} \longrightarrow$ **Pollard-ρ** $\longrightarrow n$

Complexity: if p large, non-special, and k small, for most E,
$O(\sqrt{r})$, where r is the largest prime dividing $\#E(\mathbb{F}_{p^k})$

Pairing-friendly families

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

Pairing-friendly families

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- Many constructions: BN curves, BLS curves, [GMT19], [FM19], [BEG19], ...

Pairing-friendly families

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- Many constructions: BN curves, BLS curves, [GMT19], [FM19], [BEG19], . . .
- Disclaimer for papers before 2016: New improvements/refinements to the attack methods in 2016. See eg. [BD17] for an overview.

Pairing-friendly families

Main idea: construct examples of pairings $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ where the complexity of DLP is about the same in each group.

- Many constructions: BN curves, BLS curves, [GMT19], [FM19], [BEG19], . . .
- Disclaimer for papers before 2016: New improvements/refinements to the attack methods in 2016. See eg. [BD17] for an overview.
  - Worst-case asymptotic complexity went from $L_{p^k}[1/3, 1.923]$ to $L_{p^k}[1/3, 1.526]$.

That’s cute, but what about quantum computers?

Cryptography

Sender | Channel with eavesdropper 'Eve' | Receiver

Cryptography

Sender — Channel with eavesdropper ‘Eve’ — Receiver

Problems:

- Communication channels (adversaries) store and spy on our data
- Communication channels are modifying our data

Cryptography

Sender Channel with eavesdropper ‘Eve’ Receiver

Problems:

- Communication channels (adversaries) store and spy on our data
- Communication channels are modifying our data

Goals:

- Confidentiality despite Eve’s espionage.
- Integrity: recognising Eve’s espionage.

(Slide mostly stolen from Tanja Lange)

Post-quantum cryptography

Sender — Channel with eavesdropper 'Eve' — Receiver

Post-quantum cryptography

Sender　　Channel with eavesdropper 'Eve'　　Receiver

- Eve has a quantum computer.
- Harry and Meghan don't have a quantum computer.

(Slide mostly stolen from Tanja Lange)

Why does Eve need a quantum computer?

- Asymmetric cryptography typically relies on the discrete logarithm problem being slow to solve:
  with Shor's quantum algorithm this is no longer true.

Why does Eve need a quantum computer?

- Asymmetric cryptography typically relies on the discrete logarithm problem being slow to solve:
  with Shor's quantum algorithm this is no longer true.
  $\rightsquigarrow$ will make current asymmetric algorithms obselete.

Why does Eve need a quantum computer?

- Asymmetric cryptography typically relies on the discrete logarithm problem being slow to solve:
  with Shor's quantum algorithm this is no longer true.
  $\rightsquigarrow$ will make current asymmetric algorithms obselete.
- Symmetric cryptography typically has less mathematical structure so quantum computers are less devastating, but Grover's quantum algorithm still speeds up attacks.

Why does Eve need a quantum computer?

- Asymmetric cryptography typically relies on the discrete logarithm problem being slow to solve:
  with Shor's quantum algorithm this is no longer true.
  $\rightsquigarrow$ will make current asymmetric algorithms obselete.
- Symmetric cryptography typically has less mathematical structure so quantum computers are less devastating,
  but Grover's quantum algorithm still speeds up attacks.
  $\rightsquigarrow$ reduces security of current symmetric algorithms.

Why does Eve need a quantum computer?

- Asymmetric cryptography typically relies on the discrete logarithm problem being slow to solve:
  with Shor's quantum algorithm this is no longer true.
  $\rightsquigarrow$ will make current asymmetric algorithms obselete.
- Symmetric cryptography typically has less mathematical structure so quantum computers are less devastating,
  but Grover's quantum algorithm still speeds up attacks.
  $\rightsquigarrow$ reduces security of current symmetric algorithms.

> Main goal: replace the use of the discrete logarithm problem in asymmetric cryptography with something quantum-resistant.

Where are we now?

- Post-quantum cryptography discussion dominated by NIST competition for standardization.

Where are we now?

- Post-quantum cryptography discussion dominated by NIST competition for standardization.
- This initiative comes after a US report with:

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.

Recall: Diffie–Hellman key exchange ’76

Public parameters:

- a prime p (experts: uses $\mathbb{F}_p^*$, today also elliptic curves)
- a number $g \pmod{p}$ (nonexperts: think of an integer less than p)

Alice	Eve	Bob
$\mathrm{sk}_A \xleftarrow{\text{random}} \{0 \ldots p-1\}$		$\mathrm{sk}_B \xleftarrow{\text{random}} \{0 \ldots p-1\}$
$\mathrm{pk}_A = g^{\mathrm{sk}_A}$ →		← $\mathrm{pk}_B = g^{\mathrm{sk}_B}$
$ss := (g^{\mathrm{sk}_B})^{\mathrm{sk}_A}$		$ss := (g^{\mathrm{sk}_A})^{\mathrm{sk}_B}$

- Alice and Bob agree on a shared secret key ss, then they can use that to encrypt their messages.
- Eve sees $\mathrm{pk}_A = g^{\mathrm{sk}_A}$, $\mathrm{pk}_B = g^{\mathrm{sk}_B}$; can’t find sk_A, sk_B, ss.

Recall: Diffie–Hellman key exchange '76

Public parameters:

- a prime p (experts: uses $\mathbb{F}_p^*$, today also elliptic curves)
- a number $g \pmod{p}$ (nonexperts: think of an integer less than p)

Alice — Eve

$\mathrm{sk}_A \xleftarrow{\text{random}} \{0 \ldots p-1\}$ … $-1\}$

$\mathrm{pk}_A = g^{\mathrm{sk}_A}$ … $\mathrm{pk}_B = g^{\mathrm{sk}_B}$

$ss := (g^{\mathrm{sk}_A})^{\mathrm{sk}_B}$

Broken by Shor!

- A… agree on a shared secret key ss, then they can …e that to encrypt their messages.
- Eve sees $\mathrm{pk}_A = g^{\mathrm{sk}_A}$, $\mathrm{pk}_B = g^{\mathrm{sk}_B}$; can't find sk_A, sk_B, ss.

Alternatives

Ideas to replace Diffie-Hellman key exchange:

Alternatives

Ideas to replace Diffie-Hellman key exchange:

- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.

Alternatives

Ideas to replace Diffie-Hellman key exchange:

- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.
- Hash-based signatures: uses hard-to-invert functions.
  Well-studied security, small public keys.

Alternatives

Ideas to replace Diffie-Hellman key exchange:

- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.
- Hash-based signatures: uses hard-to-invert functions.
  Well-studied security, small public keys.
- Isogeny-based encryption and signatures: based on finding maps between (elliptic) curves.
  Smallest keys, slow encryption.

Alternatives

Ideas to replace Diffie-Hellman key exchange:

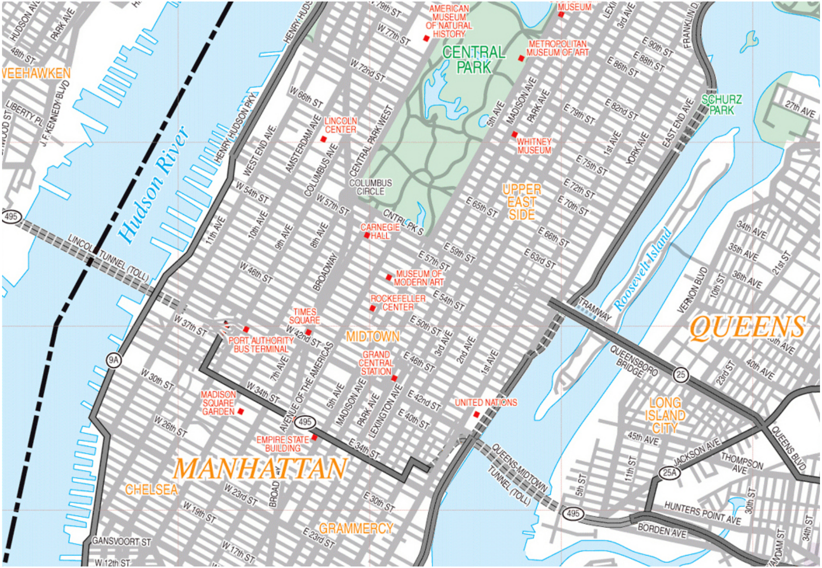
- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.
- Hash-based signatures: uses hard-to-invert functions.
  Well-studied security, small public keys.
- Isogeny-based encryption and signatures: based on finding maps between (elliptic) curves.
  Smallest keys, slow encryption.
- Lattice-based encryption and signatures: based on finding short vectors in high-dimensional lattices.
  Fastest encryption, huge keys, slow signatures.

Alternatives

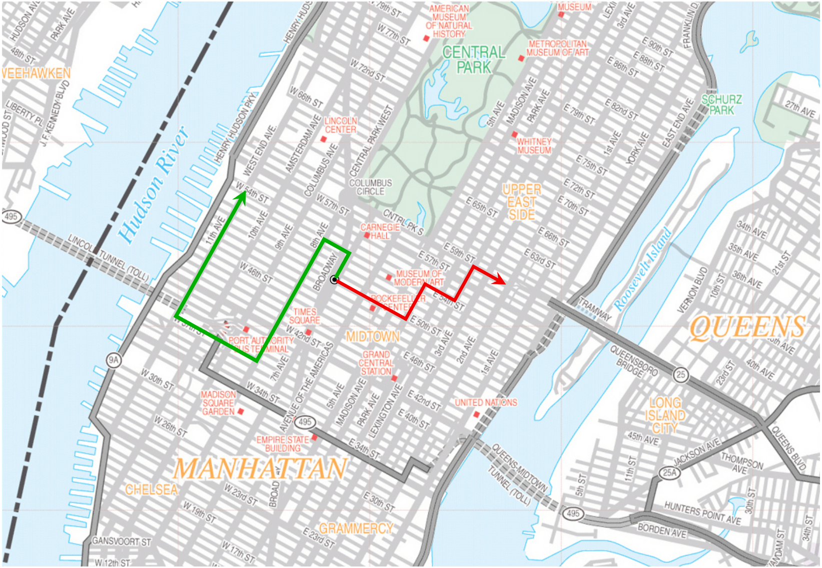
Ideas to replace Diffie-Hellman key exchange:

- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.
- Hash-based signatures: uses hard-to-invert functions.
  Well-studied security, small public keys.
- Isogeny-based encryption and signatures: based on finding maps between (elliptic) curves.
  Smallest keys, slow encryption.
- Lattice-based encryption and signatures: based on finding short vectors in high-dimensional lattices.
  Fastest encryption, huge keys, slow signatures.
- Multivariate signatures: based on solving simulateneous multivariate equations.
  Short signatures, large public keys, slow.

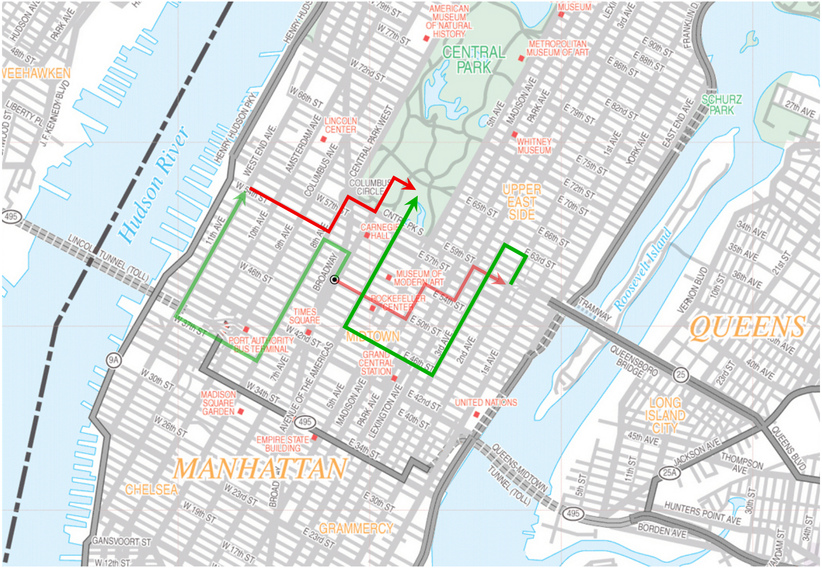
Graph walking Diffie–Hellman?



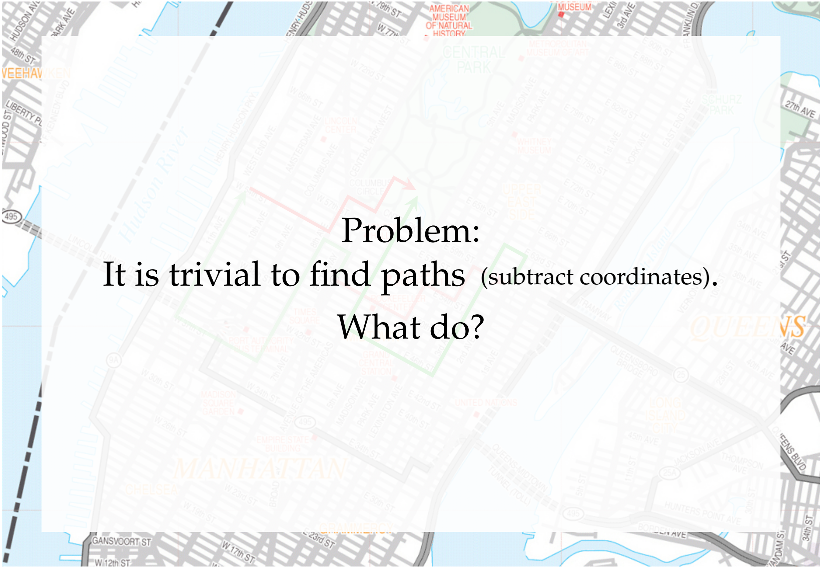
Graph walking Diffie–Hellman?



Graph walking Diffie–Hellman?



Graph walking Diffie–Hellman?



Problem:

It is trivial to find paths (subtract coordinates).

What do?

Big picture

- Isogenies are a source of exponentially-sized graphs.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- 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.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No known efficient algorithms to recover paths from endpoints.

Big picture

- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No known efficient algorithms to recover paths from endpoints.
- 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.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No known efficient algorithms to recover paths from endpoints.
- 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!

Stand back!

We’re going to do maths.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- 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 a degree-m^2 isogeny. If $m \neq 0$ in the base field, its kernel is

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

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- 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 $\{\infty\}$.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- 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 $\{(2, 9), (2, -9), \infty\}$.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

An endomorphism of E is an isogeny $E \to E$, or the zero map.
The ring of endomorphisms of E is denoted by $\mathrm{End}(E)$.

Maths background #1/3: Isogenies *(edges)*

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

- given by rational functions.
- a group homomorphism.

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

An endomorphism of E is an isogeny $E \to E$, or the zero map.
The ring of endomorphisms of E is denoted by $\text{End}(E)$.

Each isogeny $\varphi\colon E \to E'$ has a unique dual isogeny $\widehat{\varphi}\colon E' \to E$ characterized by $\widehat{\varphi} \circ \varphi = \varphi \circ \widehat{\varphi} = [\deg \varphi]$.

Maths background #2/3: 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 denoted by E/G. (cf. 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')

Maths background #2/3: Isogenies and kernels

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

The curve E' is denoted by E/G. (cf. 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')

Maths background #2/3: Isogenies and kernels

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

The curve E' is denoted by E/G. (cf. 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 supersingular curves!

[1](up to isomorphism of E')

Math slide #3/3: Supersingular isogeny graphs

Let p be a prime, q a power of p, and ℓ a positive integer $\notin p\mathbb{Z}$.

An elliptic curve $E/\mathbb{F}_q$ is *supersingular* if $p \mid (q + 1 - \#E(\mathbb{F}_q))$.

We care about the cases $\#E(\mathbb{F}_p) = p + 1$ and $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$.

$\rightsquigarrow$ easy way to control the group structure by choosing p!

Math slide #3/3: Supersingular isogeny graphs

Let p be a prime, q a power of p, and ℓ a positive integer $\notin p\mathbb{Z}$.

An elliptic curve $E/\mathbb{F}_q$ is *supersingular* if $p \mid (q + 1 - \#E(\mathbb{F}_q))$.

We care about the cases $\#E(\mathbb{F}_p) = p + 1$ and $\#E(\mathbb{F}_{p^2}) = (p + 1)^2$.

$\rightsquigarrow$ easy way to control the group structure by choosing p!

Let $S \not\ni p$ denote a set of prime numbers.

The supersingular S-isogeny graph over $\mathbb{F}_q$ consists of:

- vertices given by isomorphism classes of supersingular elliptic curves,
- edges given by equivalence classes[1] of ℓ-isogenies ($\ell \in S$),

both defined over $\mathbb{F}_q$.

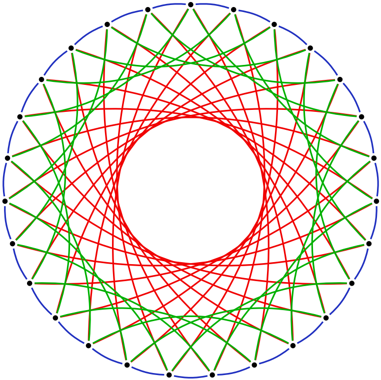
[1] Two isogenies $\varphi\colon E \to E'$ and $\psi\colon E \to E''$ are identified if $\psi = \iota \circ \varphi$ for some isomorphism $\iota\colon E' \to E''$.

The beauty and the beast

Components of the isogeny graphs look like this:

The beauty and the beast

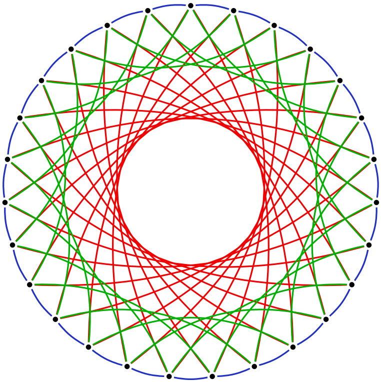
Components of the isogeny graphs look like this:



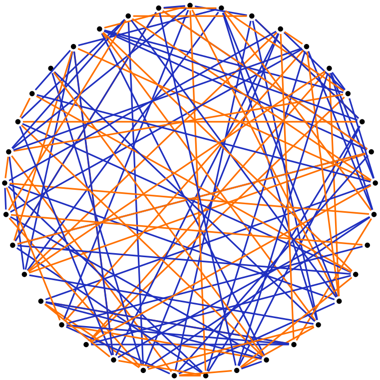
$S = \{3, 5, 7\}$, $q = 419$

The beauty and the beast

Components of the isogeny graphs look like this:



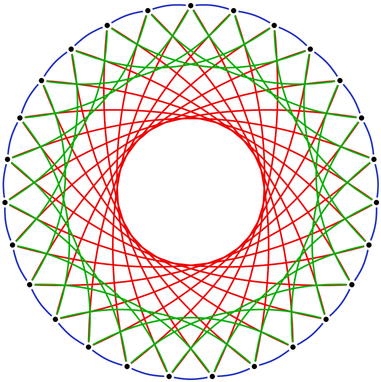
$S = \{3, 5, 7\}$, $q = 419$



$S = \{2, 3\}$, $q = 431^2$

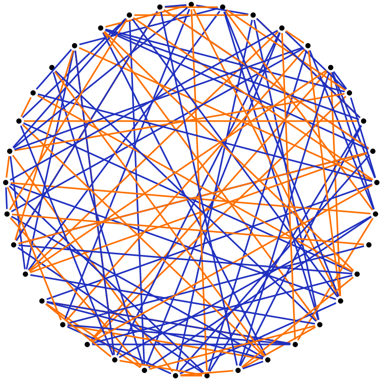
The beauty and the beast

For key exchange/KEM, there are <u>two families</u> of systems:



$q = p$

CSIDH [ˈsiːˌsaɪd]
`https://csidh.isogeny.org`

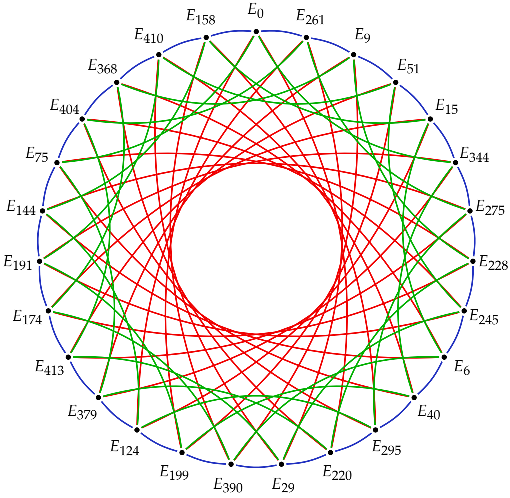


$q = p^2$

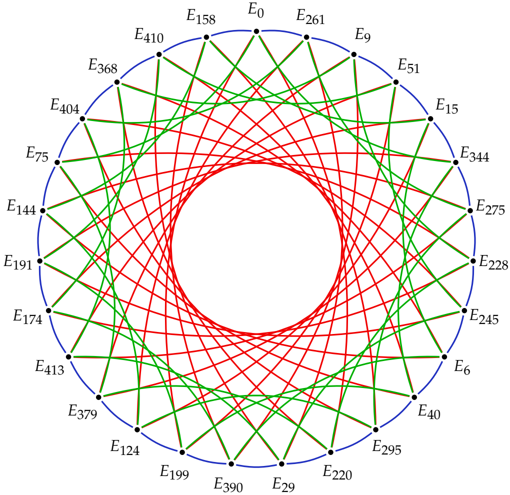
SIDH
`https://sike.org`


[ˈsiːˌsaɪd]

Isogeny graphs at the CSIDH

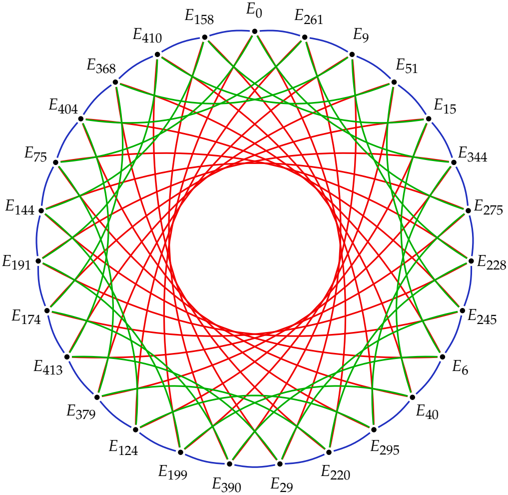


Isogeny graphs at the CSIDH



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

Isogeny graphs at the CSIDH



Nodes: Supersingular curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
Edges: 3-, 5-, and 7-isogenies.

Quantumifying Exponentiation

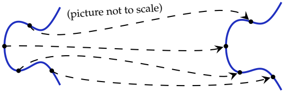
- Idea to replace DLP: replace exponentiation

$$\begin{array}{ccc} \mathbb{Z} \times G & \to & G \\ (x, g) & \mapsto & g^x \end{array}$$

  by a group action on a set.
- Replace G by the set S of supersingular elliptic curves $E_A : y^2 = x^3 + Ax^2 + x$ over $\mathbb{F}_{419}$.
- Replace $\mathbb{Z}$ by a commutative group H that acts via isogenies.
- The action of $h \in H$ on S moves the elliptic curves one step around one of the cycles.

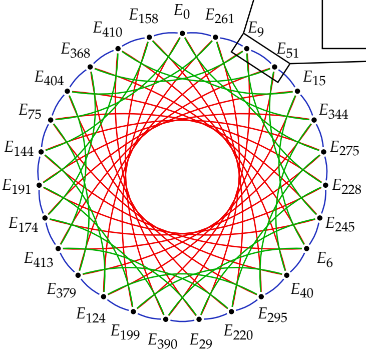
Graphs of elliptic curves

A 3-isogeny



$$E_{51}: y^2 = x^3 + 51x^2 + x \longrightarrow E_9: y^2 = x^3 + 9x^2 + x$$

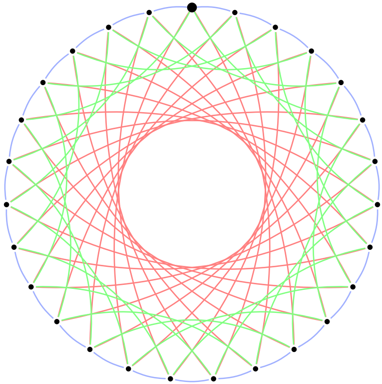
$$(x, y) \longmapsto \left(\frac{97x^3 - 183x^2 + x}{x^2 - 183x + 97}, \; y \cdot \frac{133x^3 + 154x^2 - 5x + 97}{-x^3 + 65x^2 + 128x - 133} \right)$$



Diffie and Hellman go to the CSIDH

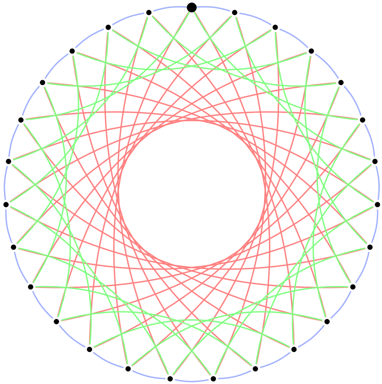
Alice

[+, −, +, −]



Bob

[+, +, −, +]



Diffie and Hellman go to the CSIDH

Alice

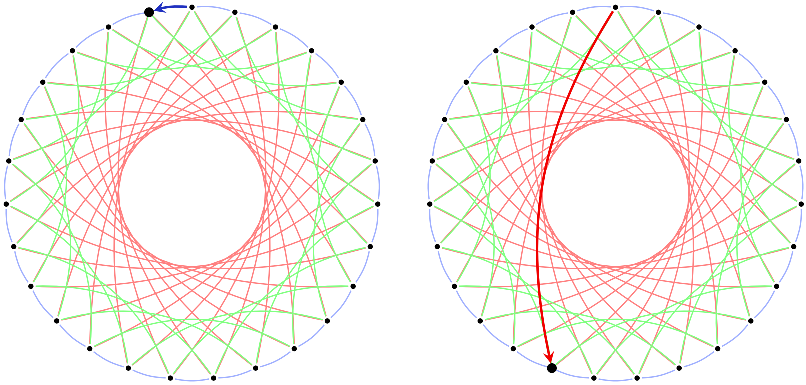
[+, −, +, −]

↑

Bob

[+, +, −, +]

↑

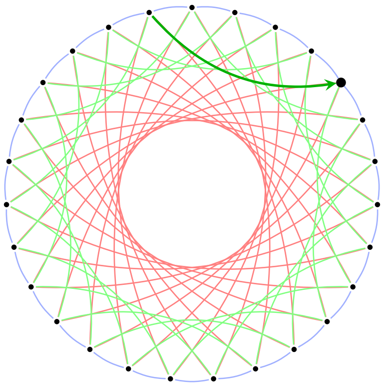


Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

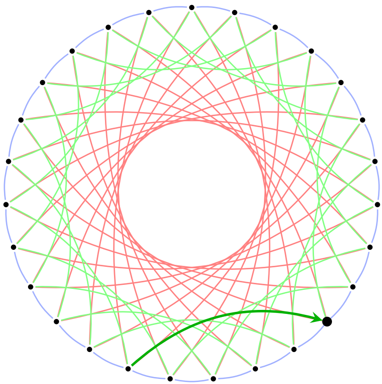
↑



Bob

[+, +, −, +]

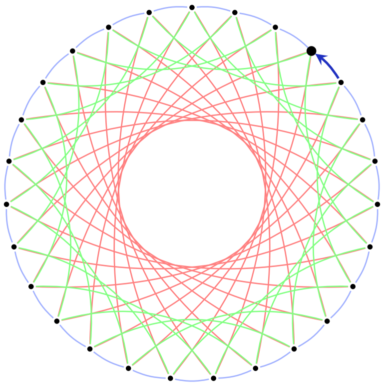
↑



Diffie and Hellman go to the CSIDH

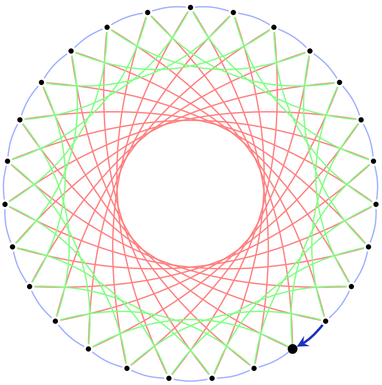
Alice

$[+, -, +, -]$
$\uparrow$



Bob

$[+, +, -, +]$
$\uparrow$

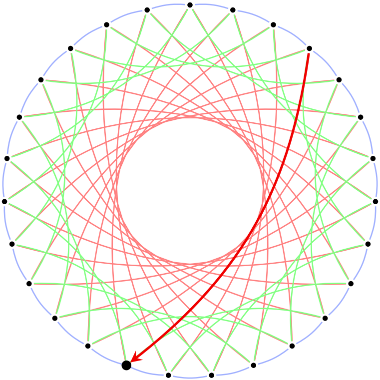


Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

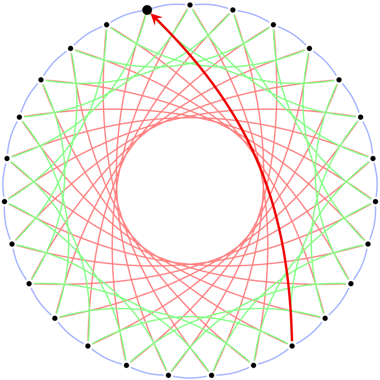
↑



Bob

[+, +, −, +]

↑



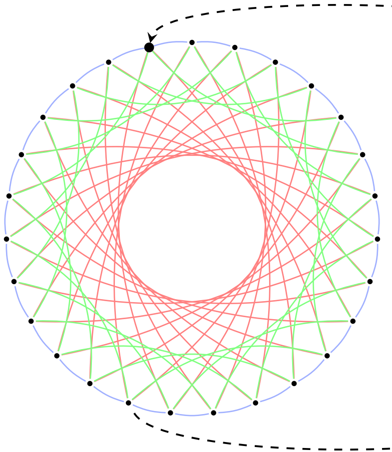
Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

Bob

[+, +, −, +]

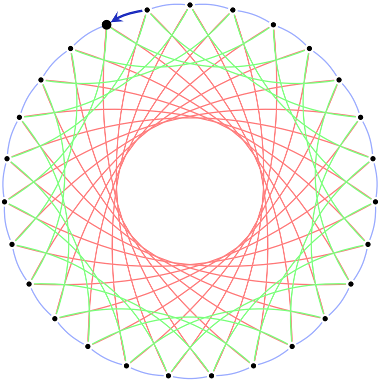


Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

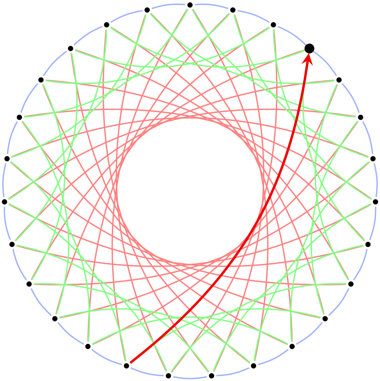
↑



Bob

[+, +, −, +]

↑



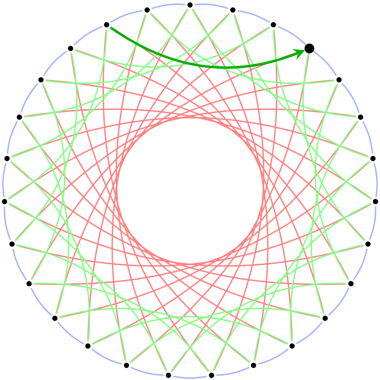
Diffie and Hellman go to the CSIDH

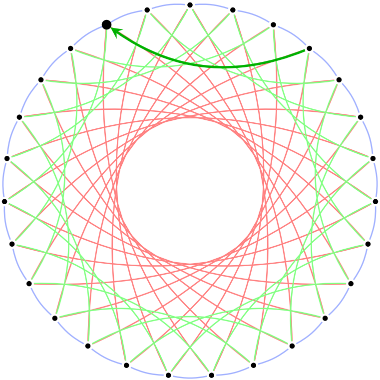
Alice

[+, −, +, −]
↑

Bob

[+, +, −, +]
↑



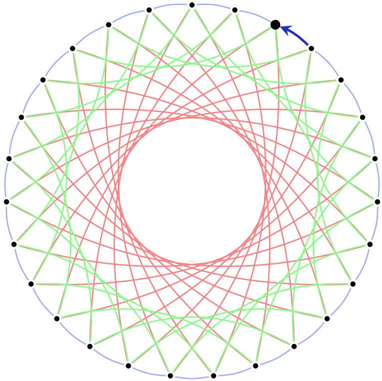


Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

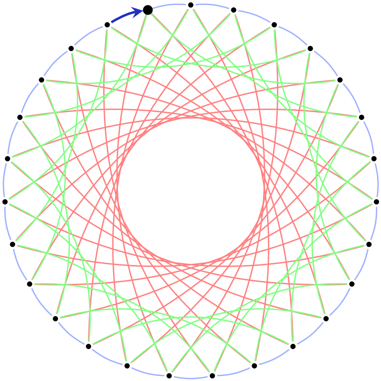
↑



Bob

[+, +, −, +]

↑

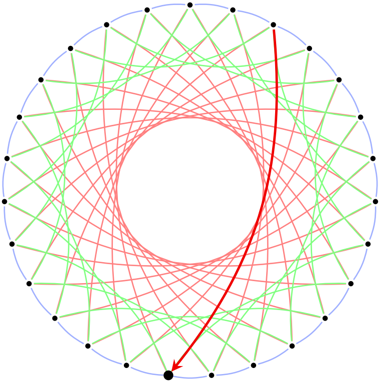


Diffie and Hellman go to the CSIDH

Alice

[+, −, +, −]

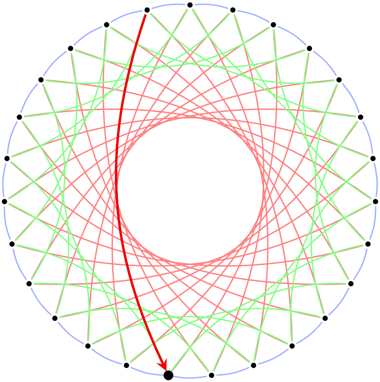
↑



Bob

[+, +, −, +]

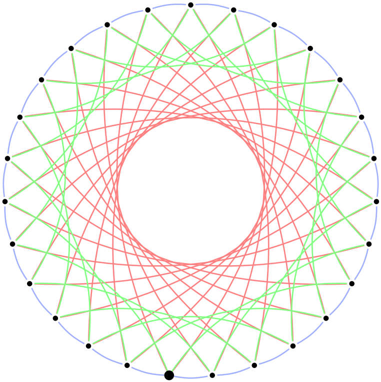
↑



Diffie and Hellman go to the CSIDH

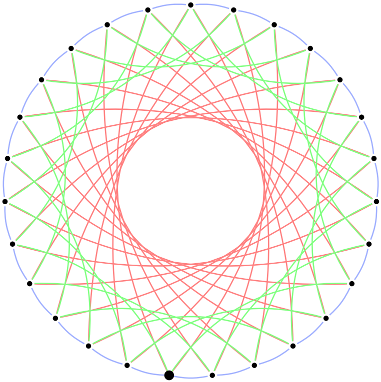
Alice

[+, −, +, −]



Bob

[+, +, −, +]



Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.

- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.

- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p + 1$ or $(p + 1)/2$.

- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p+1$ or $(p+1)/2$.
  - For every odd prime $\ell | (p+1)$, the point $\frac{p+1}{\ell}P$ is a point of order ℓ.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).

Compute neighbours in the graph

To compute a neighbour of E, we have to compute an ℓ-isogeny from E. To do this:

- Find a point P of order ℓ on E.
  - Let $E/\mathbb{F}_p$ be supersingular and $p \geq 5$. Then $E(\mathbb{F}_p) \cong C_{p+1}$ or $C_2 \times C_{(p+1)/2}$.
  - Suppose we have found $P = E(\mathbb{F}_p)$ of order $p + 1$ or $(p + 1)/2$.
  - For every odd prime $\ell | (p + 1)$, the point $\frac{p+1}{\ell} P$ is a point of order ℓ.
- Compute the isogeny with kernel $\{P, 2P, \ldots, \ell P\}$ using Vélu's formulas* (implemented in Sage).
  - Given a $\mathbb{F}_p$-rational point of order ℓ, the isogeny computations can be done over $\mathbb{F}_p$.

Representing nodes of the graph

- Every node of G_{ℓ_i} is

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

Representing nodes of the graph

- Every node of G_{ℓ_i} is

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

$\Rightarrow$ Can compress every node to a single value $A \in \mathbb{F}_p$.

Representing nodes of the graph

- Every node of G_{ℓ_i} is

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

$\Rightarrow$ Can compress every node to a single value $A \in \mathbb{F}_p$.

$\Rightarrow$ Tiny keys!

Does any A work?

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Does any A work?

No.

[1]This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p+1$ points.

Does any A work?

No.

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Does any A work?

No.

- About $\sqrt{p}$ of all $A \in \mathbb{F}_p$ are valid keys.
- Public-key validation: Check that E_A has $p + 1$ points.

  Easy Monte-Carlo algorithm: Pick random P on E_A and check $[p + 1]P = \infty$.[1]

[1] This algorithm has a small chance of false positives, but we actually use a variant that *proves* that E_A has $p + 1$ points.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.
- For fastest variant of Kuperberg, total cost of CSIDH-512 attack is at least 2^{56} qubit operations.

Quantum Security

Original proposal in 2018 paper: $\mathbb{F}_p \approx 512$ bits.

- The exact cost of the Kuperberg/Regev/CJS attack is subtle – it depends on:
  - Choice of time/memory trade-off (Regev/Kuperberg)
  - Quantum evaluation of isogenies

  (and much more).
- [BLMP19] computes one query (i.e. CSIDH-512 group action) using $765325228976 \approx 0.7 \cdot 2^{40}$ nonlinear bit operations.
- Peikert's sieve technique [P19] on fastest variant of Kuperberg requires 2^{16} queries using 2^{40} bits of quantum accessible classical memory.
- For fastest variant of Kuperberg, total cost of CSIDH-512 attack is at least 2^{56} qubit operations.
- Overheads from error correction, high quantum memory etc., not yet understood.

Venturing beyond the CSIDH

A selection of advances since original publication (2018):

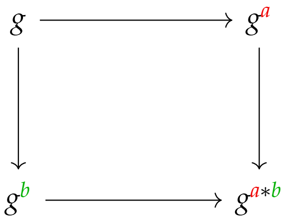
- CSURF [CD19]: exploiting 2-isogenies.
- sqrtVelu [BDLS20]: square-root speed-up on computation of large-degree isogenies.
- Radical isogenies [CDV20]: significant speed-up on isogenies of small-ish degree.
- Some work on different curve forms (e.g. Edwards, Huff).
- Knowledge of $\mathrm{End}(E_0)$ and $\mathrm{End}(E_A)$ breaks CSIDH in classical polynomial time [Wes21].
- The SQALE of CSIDH [CCJR22]: carefully constructed CSIDH parameters less susceptible to Kuperberg's algorithm.
- CTIDH [$B^2C^2LMS^2$]: Efficient constant-time CSIDH-style construction.

Now:

SIDH

Supersingular Isogeny Diffie–Hellman

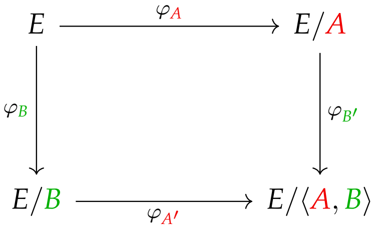
Diffie-Hellman: High-level view



SIDH: High-level view

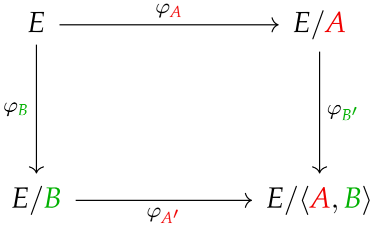
$$\begin{array}{ccc} E & \xrightarrow{\varphi_A} & E/A \\ {\scriptstyle \varphi_B}\downarrow & & \downarrow{\scriptstyle \varphi_{B'}} \\ E/B & \xrightarrow[\varphi_{A'}]{} & E/\langle A, B\rangle \end{array}$$

SIDH: High-level view



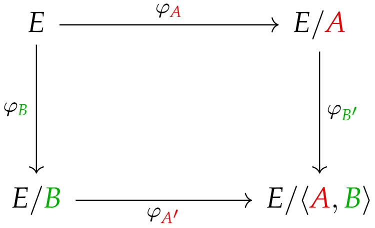
- Alice & Bob pick secret subgroups A and B of E.

SIDH: High-level view



- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.

SIDH: High-level view



- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
- Alice and Bob transmit the values E/A and E/B.

SIDH: High-level view

$$\begin{array}{ccc} E & \xrightarrow{\varphi_A} & E/A \\ \big\downarrow{\scriptstyle \varphi_B} & & \big\downarrow{\scriptstyle \varphi_{B'}} \\ E/B & \xrightarrow[\varphi_{A'}]{} & E/\langle A, B\rangle \end{array}$$

- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A \colon E \to E/A$; Bob computes $\varphi_B \colon E \to E/B$.
- Alice and Bob transmit the values E/A and E/B.
- Alice somehow obtains $A' := \varphi_B(A)$. (Similar for Bob.)

SIDH: High-level view

$$\begin{array}{ccc} E & \xrightarrow{\varphi_A} & E/A \\ {\scriptstyle \varphi_B}\downarrow & & \downarrow{\scriptstyle \varphi_{B'}} \\ E/B & \xrightarrow[\varphi_{A'}]{} & E/\langle A, B\rangle \end{array}$$

- Alice & Bob pick secret subgroups A and B of E.
- Alice computes $\varphi_A : E \to E/A$; Bob computes $\varphi_B : E \to E/B$.
- Alice and Bob transmit the values E/A and E/B.
- Alice somehow obtains $A' := \varphi_B(A)$. (Similar for Bob.)
- They both compute the shared secret

$$(E/B)/A' \cong E/\langle A, B\rangle \cong (E/A)/B'.$$

SIDH's auxiliary points

Previous slide: "Alice somehow obtains $A' := \varphi_B(A)$."

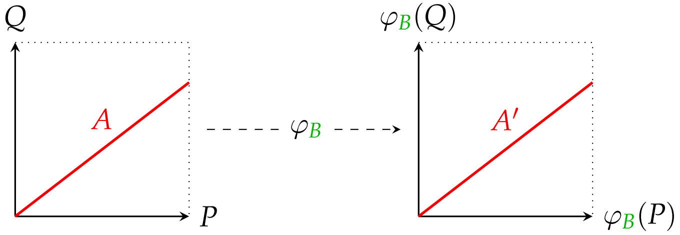
Alice knows only A, Bob knows only φ_B. Hm.

SIDH's auxiliary points

Previous slide: "Alice somehow obtains $A' := \varphi_B(A)$."

Alice knows only A, Bob knows only φ_B. Hm.

Solution: φ_B is a group homomorphism!

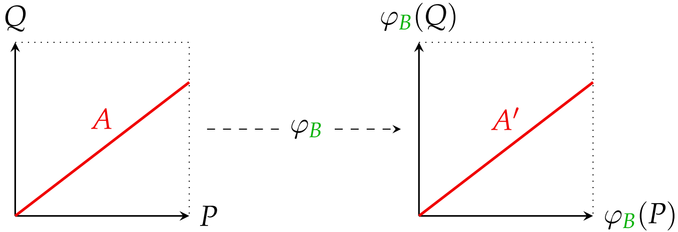


SIDH's auxiliary points

Previous slide: "Alice somehow obtains $A' := \varphi_B(A)$."

Alice knows only A, Bob knows only φ_B. Hm.

Solution: φ_B is a group homomorphism!



- Alice picks A as $\langle P + [a]Q \rangle$ for fixed public $P, Q \in E$.
- Bob includes $\varphi_B(P)$ and $\varphi_B(Q)$ in his public key.

$\Longrightarrow$ Now Alice can compute A' as $\langle \varphi_B(P) + [a]\varphi_B(Q) \rangle$!

SIDH in one slide

Public parameters:

- a large prime $p = 2^n 3^m - 1$ and a supersingular $E/\mathbb{F}_p$
- bases (P_A, Q_A) and (P_B, Q_B) of $E[2^n]$ and $E[3^m]$

Alice	public	Bob
$a \xleftarrow{\text{random}} \{0 \ldots 2^n - 1\}$		$b \xleftarrow{\text{random}} \{0 \ldots 3^m - 1\}$
$A := \langle P_A + [a] Q_A \rangle$ compute $\varphi_A : E \to E/A$		$B := \langle P_B + [b] Q_B \rangle$ compute $\varphi_B : E \to E/B$
$E/A,\ \varphi_A(P_B),\ \varphi_A(Q_B)$	⟶ / ⟵ (exchange)	$E/B,\ \varphi_B(P_A),\ \varphi_B(Q_A)$
$A' := \langle \varphi_B(P_A) + [a] \varphi_B(Q_A) \rangle$ $s := j\big((E/B)/A'\big)$		$B' := \langle \varphi_A(P_B) + [b] \varphi_A(Q_B) \rangle$ $s := j\big((E/A)/B'\big)$

Break it by: given public info, find secret key–φ_A or just A.

Security

Hard Problem:

Given

- supersingular public elliptic curves $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ connected by a secret 2^n-degree isogeny $\varphi_A : E_0 \to E_A$, and
- the action of φ_A on the 3^m-torsion of E_0,

find the secret key recover φ_A.

Security

Hard Problem:
Given

- supersingular public elliptic curves $E_0/\mathbb{F}_{p^2}$ and $E_A/\mathbb{F}_{p^2}$ connected by a secret 2^n-degree isogeny $\varphi_A : E_0 \to E_A$, and
- the action of φ_A on the 3^m-torsion of E_0,

find the secret key recover φ_A.

- Knowledge of $\mathrm{End}(E_0)$ and $\mathrm{End}(E_A)$ is sufficient to efficiently break it.
- Active attacker can recover secret.
- In SIDH, $\mathrm{End}(E_0)$ is fixed and $3^m \approx 2^n \approx \sqrt{p}$.
- If $3^m > 2^n$ or $3^m, 2^n > \sqrt{p}$, security claims are weakened.

Security of SIKE

- Best known attacks on SIKE, where $E_0/\mathbb{F}_p : y^2 = x^3 + x$ and $2^n \approx 3^m$ are on the Isogeny Problem:

Security of SIKE

- Best known attacks on SIKE, where $E_0/\mathbb{F}_p : y^2 = x^3 + x$ and $2^n \approx 3^m$ are on the Isogeny Problem:
  - The isogeny problem: given two elliptic curves, find an isogeny between them.

Security of SIKE

- Best known attacks on SIKE, where $E_0/\mathbb{F}_p : y^2 = x^3 + x$ and $2^n \approx 3^m$ are on the Isogeny Problem:
  - The isogeny problem: given two elliptic curves, find an isogeny between them.
- Best classical attack: meet-in-the-middle $O(p^{1/4})$.

Security of SIKE

- Best known attacks on SIKE, where $E_0/\mathbb{F}_p : y^2 = x^3 + x$ and $2^n \approx 3^m$ are on the Isogeny Problem:
  - The isogeny problem: given two elliptic curves, find an isogeny between them.
- Best classical attack: meet-in-the-middle $O(p^{1/4})$.
- Best quantum attack: meet-in-the-middle + Grover $O(p^{1/4})$, but slightly better in practise.

Security of SIKE

- Best known attacks on SIKE, where $E_0/\mathbb{F}_p : y^2 = x^3 + x$ and $2^n \approx 3^m$ are on the Isogeny Problem:
  - The isogeny problem: given two elliptic curves, find an isogeny between them.
- Best classical attack: meet-in-the-middle $O(p^{1/4})$.
- Best quantum attack: meet-in-the-middle + Grover $O(p^{1/4})$, but slightly better in practise.
- No commutative group action to exploit here*

What about signatures?

Ex: CSI-FiSh (S '06, D-G '18, Beullens-Kleinjung-Vercauteren '19)

Identification scheme from $H \times S \to S$:

Prover	**Public**	**Verifier**
	$E \in S, \mathfrak{l}_i \in H$	
$s_i \leftarrow \$\, \mathbb{Z}$		
$\mathsf{sk} = \prod \mathfrak{l}_i^{s_i}$,		
$\mathsf{pk} = \mathsf{sk} * E \xrightarrow{\mathsf{pk}}$	pk	
	$\xleftarrow{c}$	$c \leftarrow \$\, \{0,1\}$
$t_i \leftarrow \$\, \mathbb{Z}$		
$\mathsf{esk} = \prod \mathfrak{l}_i^{t_i}$,		
$\mathsf{epk}_1 = \mathsf{esk} * E$,		
$\mathsf{epk}_2 = \mathsf{esk} \cdot \mathsf{sk}^{-c}$	$\xrightarrow{\mathsf{pk}, \mathsf{epk}_1, \mathsf{epk}_2}$	check:
		$\mathsf{epk}_1 = \mathsf{epk}_2 * ([\mathsf{sk}^c] * E)$.

After k challenges c, an imposter succeeds with prob 2^{-k}.

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find $\mathfrak{a} \in H$ such that
$\mathfrak{a} * E = E'$.

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$

(*rational map + group homomorphism)

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

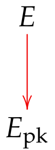
Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$
(*rational map + group homomorphism)

SQISign is a newer signature scheme based on this idea:

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$
(*rational map + group homomorphism)

SQISign is a newer signature scheme based on this idea:

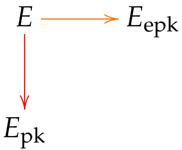


public, secret, ephemeral secret, public challenge, public proof

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$
(*rational map + group homomorphism)

SQISign is a newer signature scheme based on this idea:

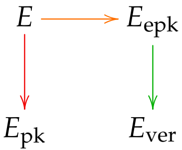


public, secret, ephemeral secret, public challenge, public proof

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$
(*rational map + group homomorphism)

SQISign is a newer signature scheme based on this idea:

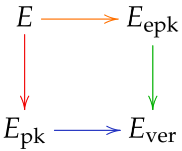


public, secret, ephemeral secret, public challenge, public proof

Ex: SQISign (De Feo-Kohel-Leroux-Petit-Wesolowski '20)

Hard Problem in CSIDH, CSI-FiSh, etc:
Given elliptic curves E and $E' \in S$, find an isogeny* $E \to E'$
(*rational map + group homomorphism)

SQISign is a newer signature scheme based on this idea:



public, secret, ephemeral secret, public challenge, public proof

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.
- Pairings. Low memory, flexible, advanced protocols.

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.
- Pairings. Low memory, flexible, advanced protocols.

Post-quantum:

- SIKE '11 KEM. Best-studied, in NIST, fast, small-ish, torsion-point attacks most likely attack avenue.

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.
- Pairings. Low memory, flexible, advanced protocols.

Post-quantum:

- SIKE ‘11 KEM. Best-studied, in NIST, fast, small-ish, torsion-point attacks most likely attack avenue.
- CSIDH ‘18 Key exchange. Small, many applications (c.f. group actions), fast-ish, known quantum attack needs further study, other attack avenues non-obvious.

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.
- Pairings. Low memory, flexible, advanced protocols.

Post-quantum:

- SIKE '11 KEM. Best-studied, in NIST, fast, small-ish, torsion-point attacks most likely attack avenue.
- CSIDH '18 Key exchange. Small, many applications (c.f. group actions), fast-ish, known quantum attack needs further study, other attack avenues non-obvious.
- CSI-FiSh '19 Digital signature. Small-ish, flexible, fast-ish, known quantum attack needs further study.

Summary and overview*

Classical:

- ECDLP. Fast, low memory, robust, well-studied, broken by Shor. Building block in TLS, Signal protocol, etc.
- Pairings. Low memory, flexible, advanced protocols.

Post-quantum:

- SIKE ‘11 KEM. Best-studied, in NIST, fast, small-ish, torsion-point attacks most likely attack avenue.
- CSIDH ‘18 Key exchange. Small, many applications (c.f. group actions), fast-ish, known quantum attack needs further study, other attack avenues non-obvious.
- CSI-FiSh ‘19 Digital signature. Small-ish, flexible, fast-ish, known quantum attack needs further study.
- SQISign ‘20 Digital signature. Small, slow, clean security assumption, no known attack avenues.

Thank you!

References

[$B^2C^2LMS^2$]	ctidh.isogeny.org
[BD17]	ia.cr/2017/334
[BDLS20]	velusqrt.isogeny.org
[BEG19]	ia.cr/2019/485
[BLMP19]	quantum.isogeny.org
[CCJR22]	ia.cr/2020/1520
[CD19]	ia.cr/2019/1404
[CDV20]	ia.cr/2020/1108
[FM19]	ia.cr/2019/555
[GMT19]	ia.cr/2019/431
[Wes21]	ia.cr/2021/1583