Progress in Post-Quantum Cryptography

Tanja Lange

Eindhoven University of Technology

15 May 2019

NIST submission Classic McEliece

- Security asymptotics unchanged by 40 years of cryptanalysis.
- Efficient and straightforward conversion OW-CPA PKE $\rightarrow$ IND-CCA2 KEM.
- Open-source (public domain) implementations.
  - Constant-time software implementations.
  - FPGA implementation of full cryptosystem.
- No patents.

Metric	**mceliece6960119**	**mceliece8192128**
Public-key size	1047319 bytes	1357824 bytes
Secret-key size	13908 bytes	14080 bytes
Ciphertext size	226 bytes	240 bytes
Key-generation time	1108833108 cycles	1173074192 cycles
Encapsulation time	153940 cycles	188520 cycles
Decapsulation time	318088 cycles	343756 cycles

See `https://classic.mceliece.org` for more details.
More parameters in round 2.

Key issues for McEliece

- Very conservative system, expected to last; has strongest security track record.
- Ciphertexts are among the shortest.
- Secret keys can be compressed.
- But public keys are really, really big!
- Sending 1MB takes time and bandwidth.

Key issues for McEliece

- Very conservative system, expected to last; has strongest security track record.
- Ciphertexts are among the shortest.
- Secret keys can be compressed.
- But public keys are really, really big!
- Sending 1MB takes time and bandwidth.
- Google–Cloudlare experiment:

  > *in some cases the public-key + ciphertext size was too large to be viable in the context of TLS*

  and even 10KB messages dropped.

Key issues for McEliece

- Very conservative system, expected to last; has strongest security track record.
- Ciphertexts are among the shortest.
- Secret keys can be compressed.
- But public keys are really, really big!
- Sending 1MB takes time and bandwidth.
- Google–Cloudlare experiment:

  > *in some cases the public-key + ciphertext size was too large to be viable in the context of TLS*

  and even 10KB messages dropped.
- If server accepts 1MB of public key from any client, an attacker can easily flood memory.
  This invites DoS attacks.

Goodness, what big keys you have!

- Public keys look like this:

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix}$$

Left part is $(n-k) \times (n-k)$ identity matrix (no need to send)
right part is random-looking $(n-k) \times k$ matrix.
E.g. $n = 6960$, $k = 5413$, so $n - k = 1547$.

Goodness, what big keys you have!

- Public keys look like this:

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix}$$

  Left part is $(n-k) \times (n-k)$ identity matrix (no need to send) right part is random-looking $(n-k) \times k$ matrix.
  E.g. $n = 6960$, $k = 5413$, so $n - k = 1547$.

- Encryption xors secretly selected columns, e.g.

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.
- On the real Internet, without per-client state:

Can servers avoid storing big keys?

$$K = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & \dots & 1 & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 & \dots & 0 & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & \dots & 1 & 0 & \dots & 1 & 1 & 1 \end{pmatrix} = (I_{n-k}|K')$$

- Encryption xors secretly selected columns.
- With some storage and trusted environment:
  Receive columns of K' one at a time, store and update partial sum.
- On the real Internet, without per-client state:
  Don't reveal intermediate results!
  Which columns are picked is the secret message!
  Intermediate results show whether a column was used or not.

McTiny (Bernstein/Lange)

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit + cookie into network packet.
- Server does computation on $K_{i,j}$, puts partial result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections).
  No per-client memory allocation.
- Client feeds the $K_{i,j}$ to server & handles storage for the server.
- Cookies also encrypted & authenticated to client.
- More stuff to avoid replay & similar attacks.

McTiny (Bernstein/Lange)

Partition key

$$K' = \begin{pmatrix} K_{1,1} & K_{1,2} & K_{1,3} & \dots & K_{1,\ell} \\ K_{2,1} & K_{2,2} & K_{2,3} & \dots & K_{2,\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ K_{r,1} & K_{r,2} & K_{r,3} & \dots & K_{r,\ell} \end{pmatrix}$$

- Each submatrix $K_{i,j}$ small enough to fit + cookie into network packet.
- Server does computation on $K_{i,j}$, puts partial result into cookie.
- Cookies are encrypted by server to itself using some temporary symmetric key (same key for all server connections).
  No per-client memory allocation.
- Client feeds the $K_{i,j}$ to server & handles storage for the server.
- Cookies also encrypted & authenticated to client.
- More stuff to avoid replay & similar attacks.
- Several round trips, but no per-client state on the server.

Parallel-to-NIST-Post-Quantum-“Competition”
Post-Quantum Cryptography

Stateful hash-based signatures

- Only one prerequisite: a good hash function, e.g. SHA3-512.
  Hash functions map long strings to fixed-length strings.
  Signature schemes use hash functions in handling plaintext.
- Old idea: 1979 Lamport one-time signatures.
- 1979 Merkle extends to more signatures.

Pros:

- Post quantum
- Only need secure hash function
- Security well understood
- Fast

Cons:

- Biggish signature though some tradeoffs possible
- Stateful, i.e., ever reusing a subkey breaks security.
  Adam Langley "for most environments it's a huge foot-cannon."

Stateful hash-based signatures

- Only one prerequisite: a good hash function, e.g. SHA3-512. Hash functions map long strings to fixed-length strings. Signature schemes use hash functions in handling plaintext.
- Old idea: 1979 Lamport one-time signatures.
- 1979 Merkle extends to more signatures.

Pros:

- Post quantum
- Only need secure hash function
- Security well understood
- Fast
- We can count: OS update, code signing, . . . naturally keep state.

Cons:

- Biggish signature though some tradeoffs possible
- Stateful, i.e., ever reusing a subkey breaks security. Adam Langley "for most environments it's a huge foot-cannon."

Standardization progress

- CFRG has published 2 RFCs: RFC 8391 and RFC 8554

Datatracker Groups Documents Meetings Other User

```
Internet Research Task Force (IRTF)                          A. Huelsing
Request for Comments: 8391                                 TU Eindhoven
Category: Informational                                        D. Butin
ISSN: 2070-1721                                            TU Darmstadt
                                                                S. Gazdag
                                                               genua GmbH
                                                             J. Rijneveld
                                                      Radboud University
                                                              A. Mohaisen
                                         University of Central Florida
                                                                 May 2018

                XMSS: eXtended Merkle Signature Scheme
```

Datatracker Groups Documents Meetings Other User

```
Internet Research Task Force (IRTF)                           D. McGrew
Request for Comments: 8554                                    M. Curcio
Category: Informational                                       S. Fluhrer
ISSN: 2070-1721                                            Cisco Systems
                                                              April 2019

               Leighton-Micali Hash-Based Signatures
```

Standardization progress

- CFRG has published 2 RFCs: RFC 8391 and RFC 8554
- NIST has gone through two rounds of requests for public input, most are positive and recommend standardizing XMSS and LMS. Only concern is about statefulness in general.

Standardization progress

- CFRG has published 2 RFCs: RFC 8391 and RFC 8554
- NIST has gone through two rounds of requests for public input, most are positive and recommend standardizing XMSS and LMS. Only concern is about statefulness in general.

- ISO SC27 JTC1 WG2 has started a study period on stateful hash-based signatures.

Post-NIST-Post-Quantum-“Competition”
Post-Quantum Cryptography


[ˈsiːˌsaɪd]

CSIDH: An Efficient Post-Quantum Commutative Group Action

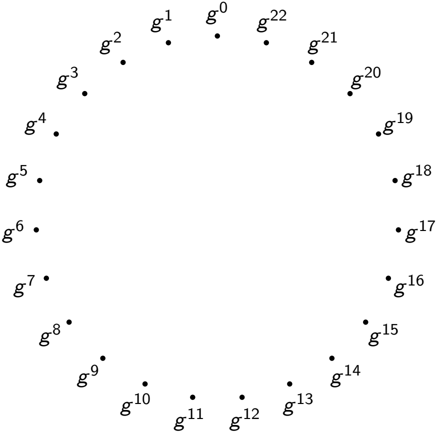
CSIDH: An Efficient Post-Quantum Commutative Group Action

Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, Joost Renes 2018

- 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 $A \in \mathbf{F}_p$; 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.

Square-and-multiply

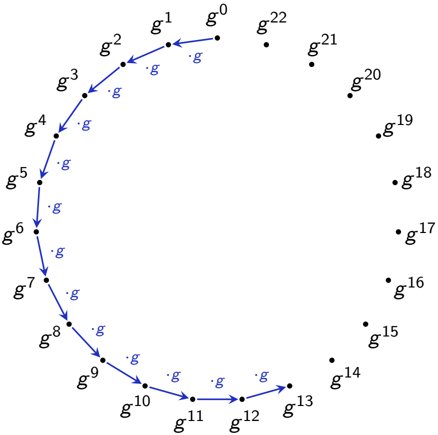
Reiminder: DH in group with $\#G = 23$. Alice computes g^{13}.



Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply

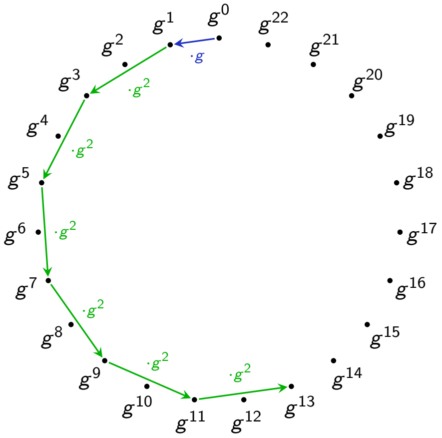
Reiminder: DH in group with $\#G = 23$. Alice computes g^{13}.



Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply

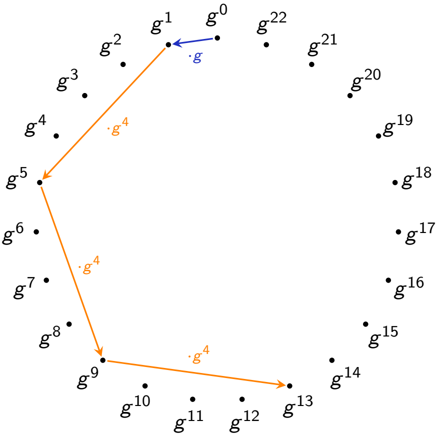
Reiminder: DH in group with $\#G = 23$. Alice computes g^{13}.



Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply

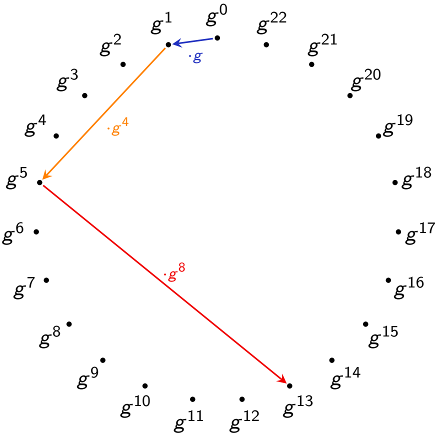
Reiminder: DH in group with $\#G = 23$. Alice computes g^{13}.



Pretty pictures by Chloe Martindale and Lorenz Panny.

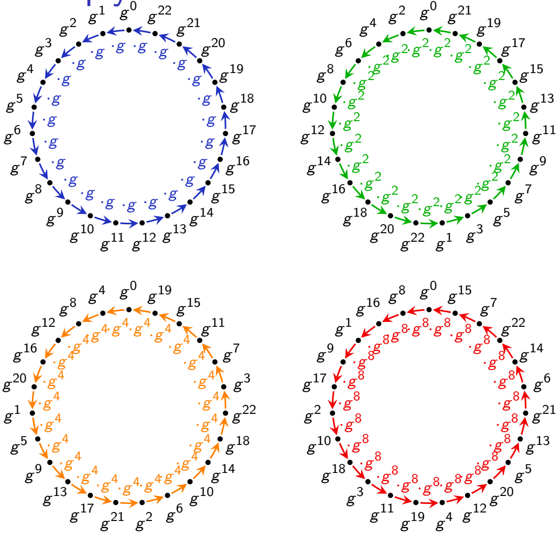
Square-and-multiply

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



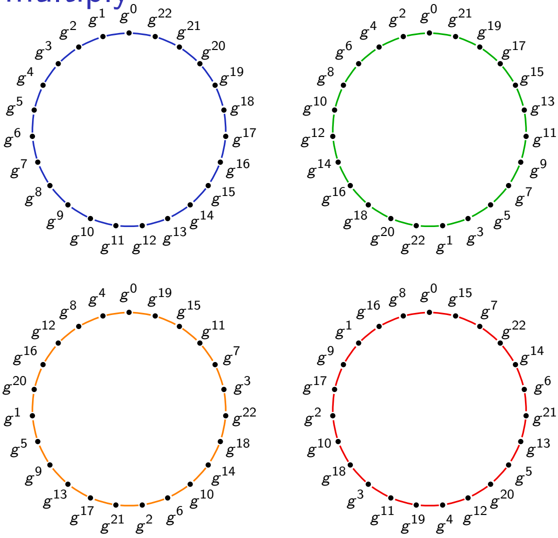
Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply



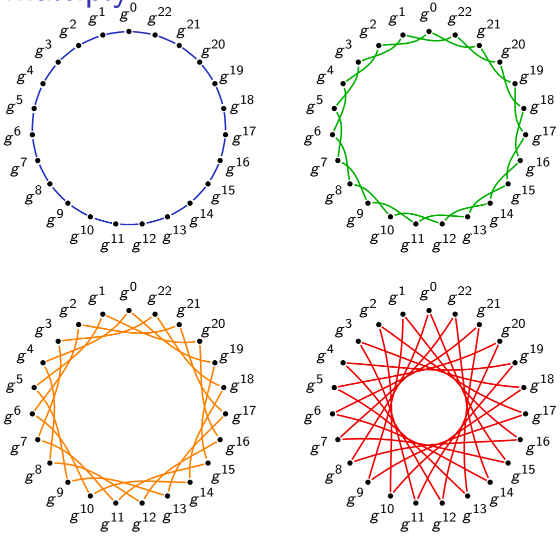
Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply



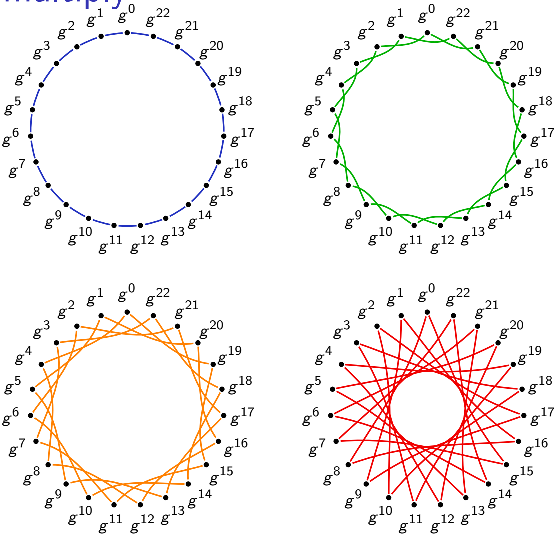
Pretty pictures by Chloe Martindale and Lorenz Panny.

Square-and-multiply



Pretty pictures by Chloe Martindale and Lorenz Panny.

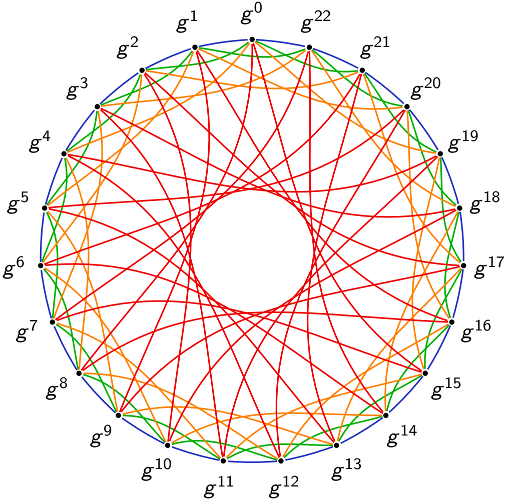
Square-and-multiply



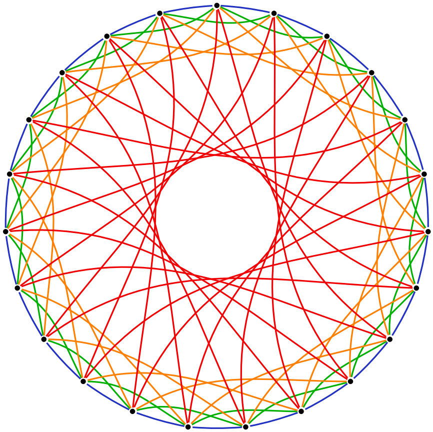
Cycles are *compatible*: [right, then left] = [left, then right], etc.

Pretty pictures by Chloe Martindale and Lorenz Panny.

Union of cycles: rapid mixing



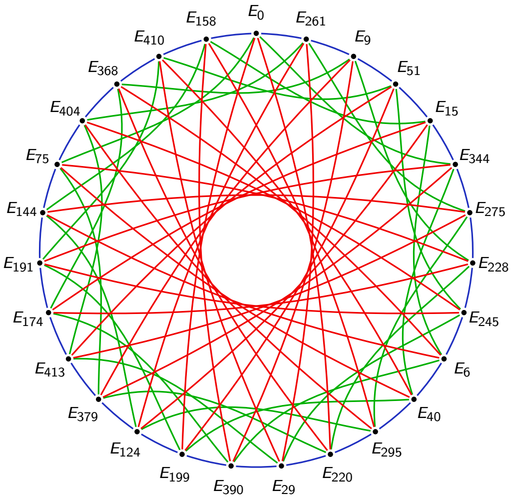
Union of cycles: rapid mixing



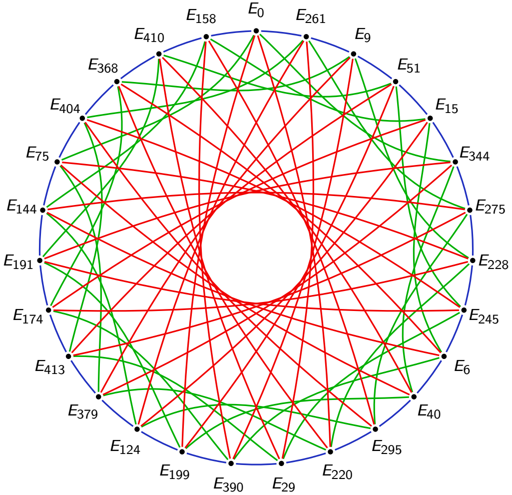
CSIDH: Nodes are now *elliptic curves* and edges are *isogenies*.

Pretty pictures by Chloe Martindale and Lorenz Panny.

Graphs of elliptic curves

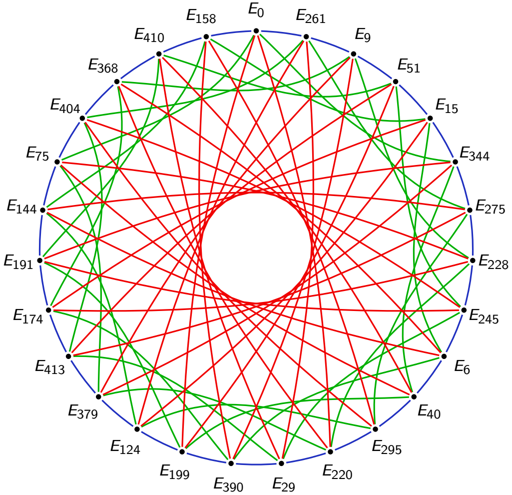


Graphs of elliptic curves



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

Graphs of elliptic curves



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

Edges: 3-, 5-, and 7-isogenies.

Pretty pictures by Chloe Martindale and Lorenz Panny.

Security

Size of key space:

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

Without quantum computer:

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

Security

Size of key space:

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

Without quantum computer:

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

With quantum computer:

- Hidden-shift algorithms apply: Subexponential complexity.
  - Literature contains mostly asymptotics.
  - Recent work analyzing cost: see `https://quantum.isogeny.org`.

CSIDH security:

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

CSIDH-512

Sizes:

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

Performance on typical Intel Skylake laptop core:

- Wall-clock time: 27ms per operation.
- Clock cycles: about $7 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.

CSIDH-512

Sizes:

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

Performance on typical Intel Skylake laptop core:

- Wall-clock time: 27ms per operation.
- Clock cycles: about $7 \cdot 10^7$ per operation.
- Somewhat more for constant-time implementations.

Security:

- Pre-quantum: at least 128 bits.
- Post-quantum: complicated. AFAWK similar to AES-128.

Website:

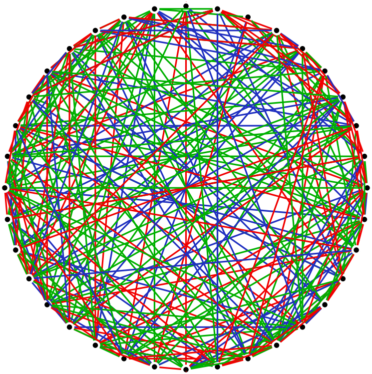
- `https://csidh.isogeny.org/`

SIDH vs. CSIDH

Nodes: Supersingular elliptic curves defined over k up to $\cong_k$.
Edges: 3-, 5-, and 7-isogenies defined over k up to $\cong_k$.

SIDH vs. CSIDH

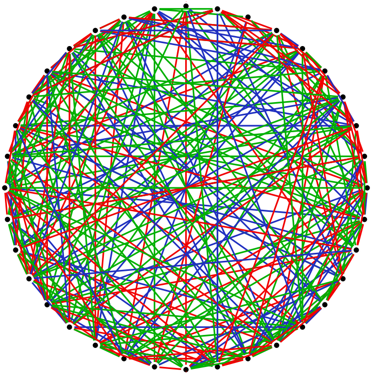
Nodes: Supersingular elliptic curves defined over k up to $\cong_k$.
Edges: 3-, 5-, and 7-isogenies defined over k up to $\cong_k$.



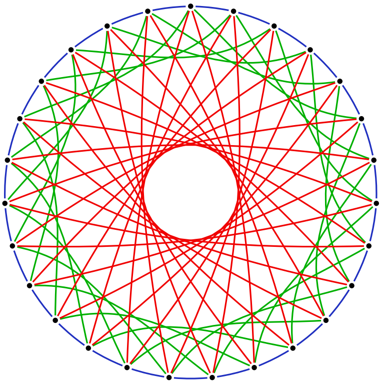
$k = \mathbf{F}_{419^2}$ (same as $\overline{\mathbf{F}}_{419}$)
SIDH case

SIDH vs. CSIDH

Nodes: Supersingular elliptic curves defined over k up to $\cong_k$.
Edges: 3-, 5-, and 7-isogenies defined over k up to $\cong_k$.



$k = \mathbf{F}_{419^2}$ (same as $\overline{\mathbf{F}}_{419}$)
SIDH case



$k = \mathbf{F}_{419}$
CSIDH case

Pretty pictures by Chloe Martindale and Lorenz Panny.


Questions?