Cryptography and quantum computers: Where do we stand?

Chloe Martindale

University of Bristol

CS Seminar, Bristol, 29 January 2020

What is this all about?

Cryptography

Sender — Channel with eavesdropper ‘Eve’ — Receiver

Cryptography

Sender　　Channel with eavesdropper 'Eve'　　Receiver

Problems:

- Communication channels store and spy on our data
- Communication channels are modifying our data

Cryptography

Sender Channel with eavesdropper ‘Eve’ Receiver

Problems:

- Communication channels 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?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.

Why does Eve need a quantum computer?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.
- 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?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.
- 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?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.
- 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?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.
- 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?

- In practise, crypto relies on a mix of asymmetric and symmetric cryptography.
- 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.

Case study: Diffie–Hellman key exchange '76

Public parameters:

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

Case study: Diffie–Hellman key exchange '76

Public parameters:

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

Harry	Eve	Meghan
$a \xleftarrow{\text{random}} \{0 \ldots p-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots p-1\}$
n^a →		← n^b
$s := (n^b)^a$		$s := (n^a)^b$

Case study: Diffie–Hellman key exchange '76

Public parameters:

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

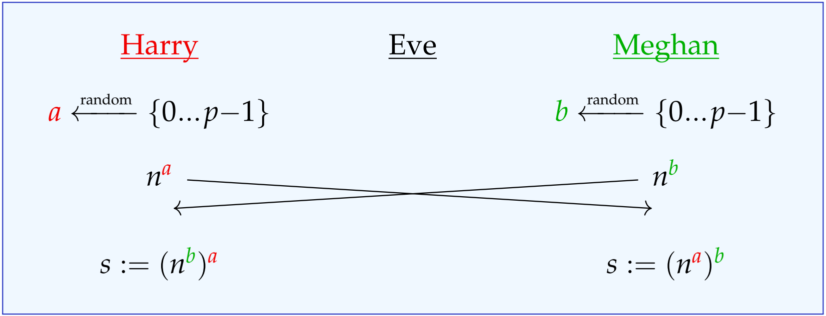
Harry	Eve	Meghan
$a \xleftarrow{\text{random}} \{0 \ldots p-1\}$		$b \xleftarrow{\text{random}} \{0 \ldots p-1\}$
n^a		n^b
$s := (n^b)^a$		$s := (n^a)^b$

- Harry and Meghan agree on a secret key s, then they can use that to encrypt their messages.

Case study: Diffie–Hellman key exchange '76

Public parameters:

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



- Harry and Meghan agree on a secret key s, then they can use that to encrypt their messages.
- Eve sees n^a and n^b, but can't find a, b, or s.

Case study: Diffie–Hellman key exchange ’76

Public parameters:

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

- H[illegible]eghan agree on a secret key s, then they can use [illegible]at to encrypt their messages.
- Eve sees n^a and n^b, but can’t find a, b, or s.

Alternatives

Ideas to replace the discrete logarithm problem:

Alternatives

Ideas to replace the discrete logarithm problem:

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

Alternatives

Ideas to replace the discrete logarithm problem:

- 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 the discrete logarithm problem:

- 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 the discrete logarithm problem:

- 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.

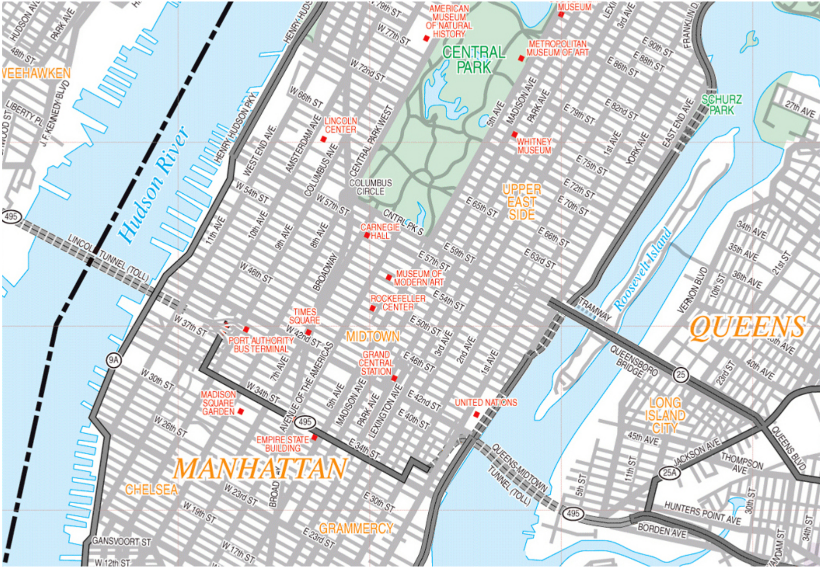
Alternatives

Ideas to replace the discrete logarithm problem:

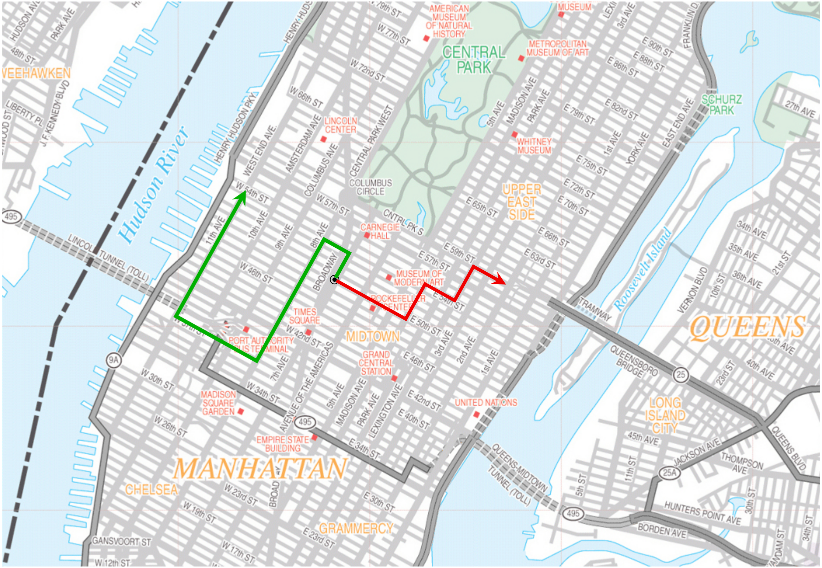
- Code-based encryption: uses error correcting codes.
  Short ciphertexts, large public keys.
- Hash-based signatures: uses hard-to-invert functions.
  Well-studied security, small public keys.
- Isogeny-based encryption and signatures: based on finding maps between (elliptic) curves.
  Smallest keys,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

(Slide mostly stolen from Tanja Lange)

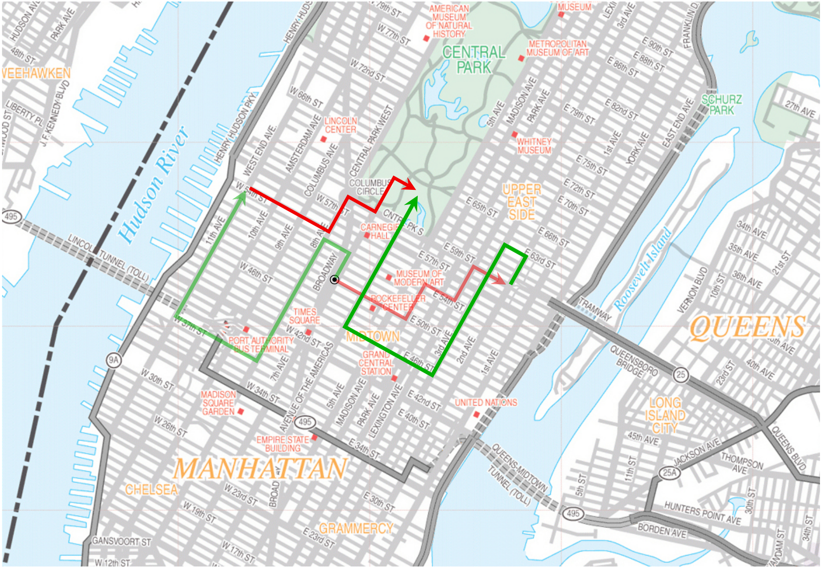
Case study: Isogenies. Graph walking Diffie-Hellman?



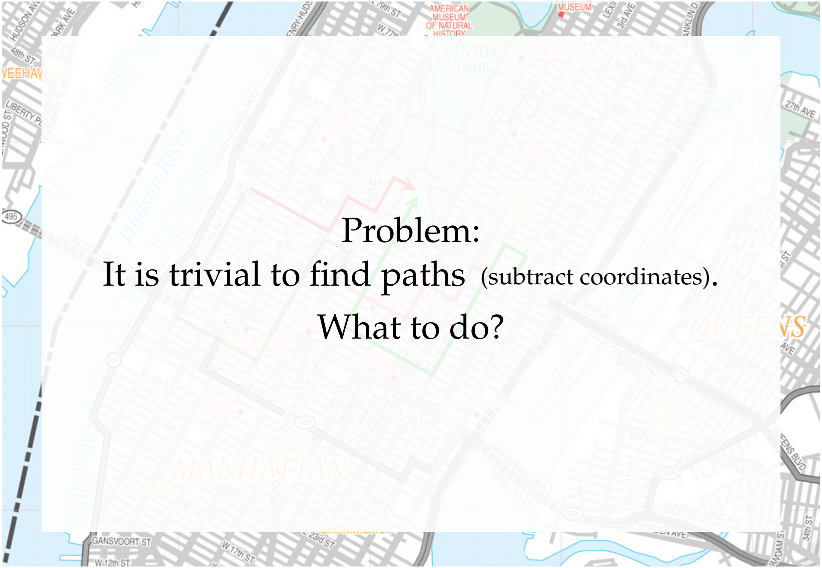
Case study: Isogenies. Graph walking Diffie-Hellman?



Case study: Isogenies. Graph walking Diffie-Hellman?



Case study: Isogenies. Graph walking Diffie-Hellman?



Problem:

It is trivial to find paths (subtract coordinates).

What to do?

Case study: Isogenies. Big picture

- Isogenies are a source of exponentially-sized graphs.

Case study: Isogenies. Big picture

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

Case study: Isogenies. 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.

Case study: Isogenies. 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.

Case study: Isogenies. Big picture

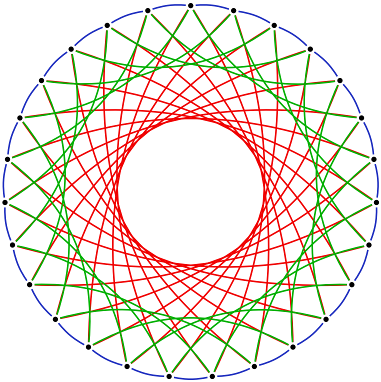
- Isogenies are a source of exponentially-sized graphs.
- We can walk efficiently on these graphs.
- Fast mixing: short paths to (almost) all nodes.
- No known efficient algorithms to recover paths from endpoints.
- Enough structure to navigate the graph meaningfully.
  That is: some well-behaved ‘directions’ to describe paths.

Case study: Isogenies

Components of the isogeny graphs look like this:

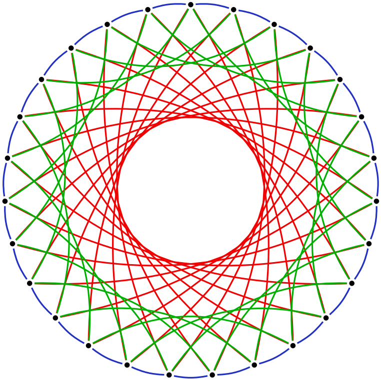
Case study: Isogenies

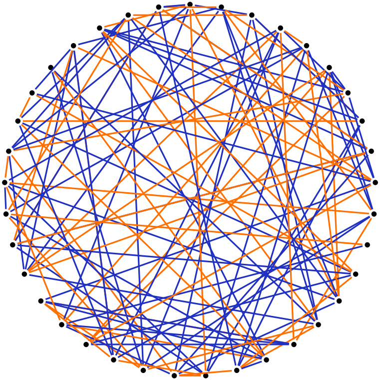
Components of the isogeny graphs look like this:



Case study: Isogenies

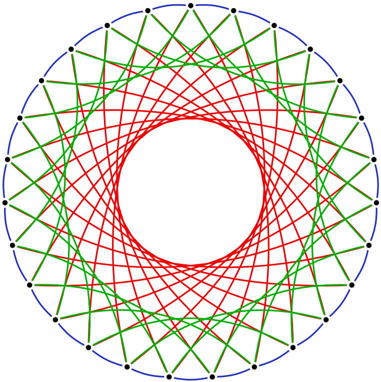
Components of the isogeny graphs look like this:



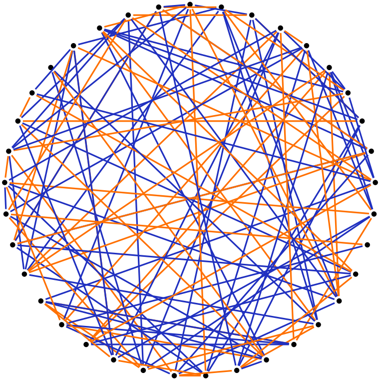


Case study: Isogenies

At this time, there are two distinct families of systems:

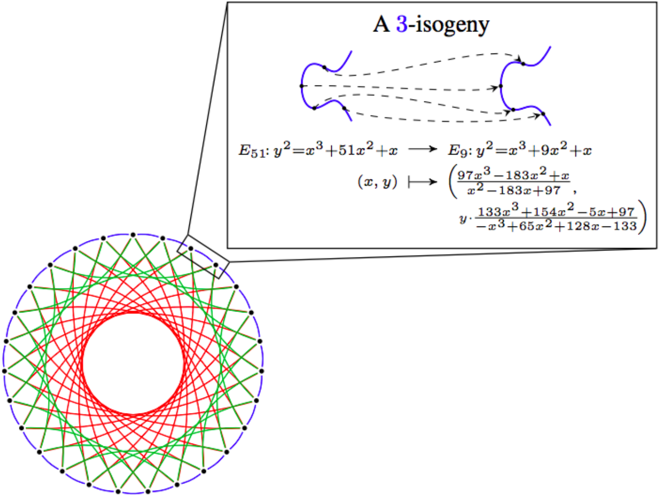


CSIDH [ˈsiːˌsaɪd]
https://csidh.isogeny.org

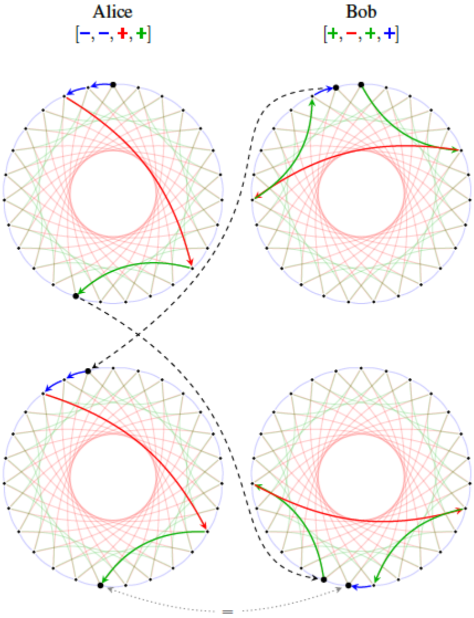


SIKE
https://sike.org

Case study: Isogenies



Case study: Isogenies. Key exchange at the CSIDH



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.

Where are we now (according to NIST)?

The NIST not-a-competition:

- Had 82 submissions in 2017
- 69 were accepted
- 26 submissions currently in 2nd round, aiming for a total of 3 rounds
- Aiming for standardization in 2022.

Where are we now (according to NIST)?

Stolen from NIST's/Dustin Moody's Round 1 summary from 2019:

	Signatures	Encryption
Code-based	2	17
Hased-based	3	0
Isogeny-based	1	1
Lattice-based	5	21
Multivariate	7	2
Others	2	4

Where are we now (according to NIST)?

Stolen from NIST's/Dustin Moody's Round 1 summary from 2019:

	Signatures	Encryption
Code-based	2	17
Hased-based	3	0
Isogeny-based	1	1
Lattice-based	5	21
Multivariate	7	2
Others	2	4

- Doesn't include CSIDH!
  (It is newer than the NIST competition).

What can we do?

We have:

- KEM/Encryption and signatures
  (many options from NIST competition).
- Diffie-Hellman-style / non-interactive key exchange
  (only option is with CSIDH).

We don't have:

- Anything else! For example, privacy-preserving protocols.

Important open problems/research directions

Needed for all post-quantum candidates:

- Thorough cryptanalysis – classical and quantum.
- Secure and efficient implementation (especially considering hardware limitations).
- Meaningful comparison between candidates (must come from comparable implementations).
- More advanced protocols.

Thank you!