Cryptography
&
Quantum Computers

Lorenz Panny

Technische Universität München

Antrittsvorlesung, Garching, 7 February 2024

This talk

Why cryptography?

The quantum threat

Post-quantum everything

Highlight: Isogenies

This talk

Why cryptography?

The quantum threat

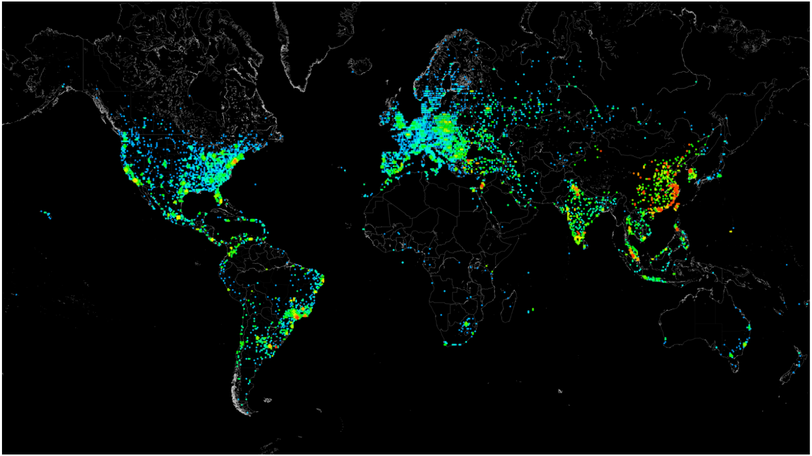
Post-quantum everything

Highlight: Isogenies

The internet

The ARPANET in December 1969

The internet



The internet

...is a giant computer network run by not necessarily trustworthy strangers.

Nothing to hide?

Cryptography is vital for much more than “just” privacy!

Nothing to hide?

Cryptography is vital for much more than “just” privacy!

- Consequences of insufficient communications security range from inconvenient to catastrophic, *in the real world.*

Nothing to hide?

Cryptography is vital for much more than “just” privacy!

- Consequences of insufficient communications security range from inconvenient to catastrophic, *in the real world.*
- Almost every bit of data gets routed over the internet at some point — including the software everyone runs.

Nothing to hide?

Cryptography is vital for much more than “just” privacy!

- Consequences of insufficient communications security range from inconvenient to catastrophic, *in the real world.*
- Almost every bit of data gets routed over the internet at some point — including the software everyone runs.

Nothing to hide?

Cryptography is vital for much more than “just” privacy!

- Consequences of insufficient communications security range from inconvenient to catastrophic, *in the real world.*
- Almost every bit of data gets routed over the internet at some point — including the software everyone runs.

- Existential threat: **CRITICAL INFRASTRUCTURE.**
  - Even airgapped systems are at risk: Firmware updates...

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.
- All users have the same capabilities.
  For instance: encrypting *and* decrypting.

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.
- All users have the same capabilities.
  For instance: encrypting *and* decrypting.
- Hence, symmetric.

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.
- All users have the same capabilities.
  For instance: encrypting *and* decrypting.
- Hence, symmetric.

Public-key cryptography (since ≈ 1976):

- Keys are now pairs: a private key and a public key.

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.
- All users have the same capabilities.
  For instance: encrypting *and* decrypting.
- Hence, symmetric.

Public-key cryptography (since $\approx$ 1976):

- Keys are now pairs: a private key and a public key.
- They give the respective owners different capabilities.

Two kinds of cryptography

"Classical" cryptography (for thousands of years):

- Secret keys exchanged in advance via a secure channel.
- All users have the same capabilities.
  For instance: encrypting *and* decrypting.
- Hence, symmetric.

Public-key cryptography (since $\approx$1976):

- Keys are now pairs: a private key and a public key.
- They give the respective owners different capabilities.
- Hence, asymmetric.

Example: Digital signatures

Example: Digital signatures

- Alice uses her private key to sign a (digital) document.

Example: Digital signatures

- Alice uses her private key to sign a (digital) document.
- Anyone can verify the signature using Alice's public key.

Example: Digital signatures

- Alice uses her private key to sign a (digital) document.
- Anyone can verify the signature using Alice's public key.

 This mimics the *intended* properties of a "real" signature.

Example: Public-key encryption

Example: Public-key encryption

- Anyone can use Bob's public key to encrypt a message

Example: Public-key encryption

- Anyone can use Bob's public key to encrypt a message such that only he can decrypt it using his private key.

Example: Public-key encryption

- Anyone can use Bob's public key to encrypt a message such that only he can decrypt it using his private key.

 Analogy: An open padlock for which Bob has the key.

Kerckhoffs' principle

Auguste Kerckhoffs, « La cryptographie militaire », Journal des sciences militaires, vol. IX, pp. 5–38, Janvier 1883, pp. 161–191, Février 1883.

> 2° Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;

Kerckhoffs' principle

Auguste Kerckhoffs, « La cryptographie militaire », Journal des sciences militaires, vol. IX, pp. 5–38, Janvier 1883, pp. 161–191, Février 1883.

> 2° Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;

- Security must rely exclusively on the secrecy of the keys! The method is assumed to be known to the public.

Kerckhoffs' principle

Auguste Kerckhoffs, « La cryptographie militaire », Journal des sciences militaires, vol. IX, pp. 5–38, Janvier 1883, pp. 161–191, Février 1883.

> 2° Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;

- Security must rely exclusively on the secrecy of the keys! The method is assumed to be known to the public.

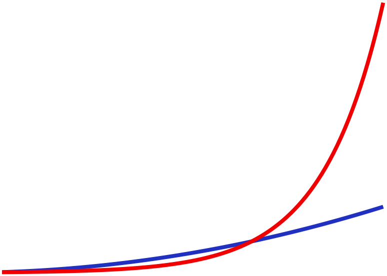
(Notice how this constitutes an important prerequisite for the development of cryptography as a science.)

Hard problems

- By design, asymmetric cryptography is always breakable
  — *at absurdly high costs*.

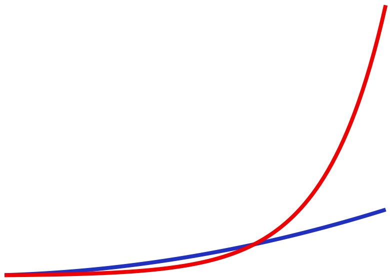
Hard problems

- By design, asymmetric cryptography is always breakable
  — *at absurdly high costs*.
- Security relies on computationally hard problems.



Hard problems

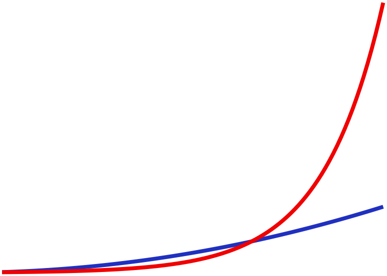
- By design, asymmetric cryptography is always breakable
  — *at absurdly high costs*.
- Security relies on computationally hard problems.



- Great source of hard problems: Algebra!
  Finite fields, elliptic curves, number fields, class groups, ...

Hard problems

- By design, asymmetric cryptography is always breakable
  — *at absurdly high costs*.
- Security relies on computationally hard problems.



- Great source of hard problems: Algebra!
  Finite fields, elliptic curves, number fields, class groups, ...
  - Key feature: These objects have a lot of useful structure.
  - Sweet spot: just enough to make things *functional* but *secure*.

On hardness (1)

- We almost never know *for certain* if cryptography is secure.

On hardness (1)

- We almost never know *for certain* if cryptography is secure.
- "Provable security" only reduces to a hardness *assumption*.
  Typical statement: "Breaking TLS is no easier than solving DLP or breaking AES."

On hardness (1)

- We almost never know *for certain* if cryptography is secure.
- "Provable security" only reduces to a hardness *assumption*.
  Typical statement: "Breaking TLS is no easier than solving DLP or breaking AES."
- Theory: If nontrivial cryptography is secure, then $\mathbf{P} \neq \mathbf{NP}$.

On hardness (1)

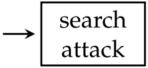
- We almost never know *for certain* if cryptography is secure.
- "Provable security" only reduces to a hardness *assumption*.
  Typical statement: "Breaking TLS is no easier than solving DLP or breaking AES."
- Theory: If nontrivial cryptography is secure, then $\mathbf{P} \neq \mathbf{NP}$.
  Reality: Does it matter? Is an $O(n^{666})$ algorithm really "tractable"?

A cryptanalyst's life

Have: Supposedly hard computational problem.

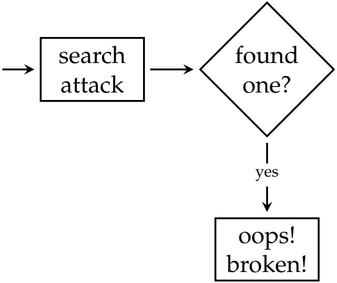
A cryptanalyst's life

Have: Supposedly hard computational problem.



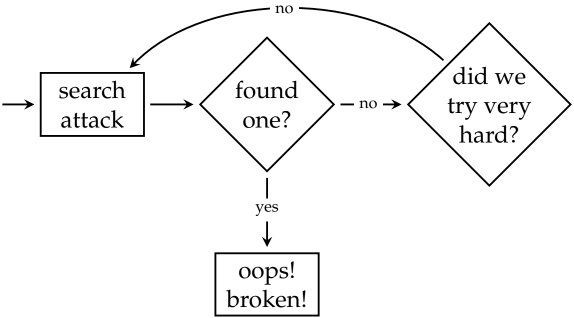
A cryptanalyst's life

Have: Supposedly hard computational problem.



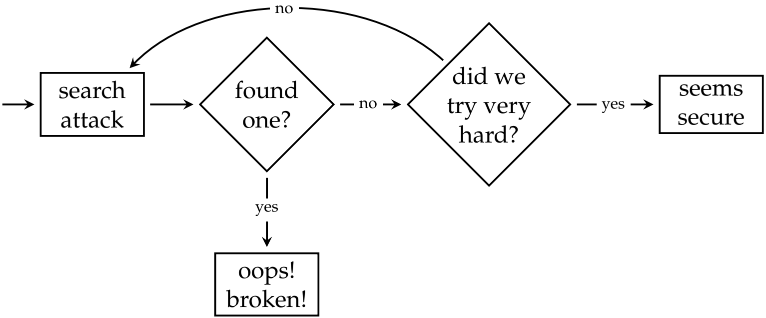
A cryptanalyst's life

Have: Supposedly hard computational problem.



A cryptanalyst's life

Have: Supposedly hard computational problem.



On hardness (2)

Common claim:
"**NP**-hard problems are needed/good for cryptography."

On hardness (2)

Common claim:

"**NP**-hard problems are needed/good for cryptography."

Reality:

- Cryptography *does not care* about worst-case hardness.

On hardness (2)

Common claim:

"**NP**-hard problems are needed/good for cryptography."

Reality:

- Cryptography *does not care* about worst-case hardness.
- Anything in **NP** can be viewed as an instance of some **NP**-complete problem, by definition.

On hardness (2)

Common claim:

"**NP**-hard problems are needed/good for cryptography."

Reality:

- Cryptography *does not care* about worst-case hardness.
- Anything in **NP** can be viewed as an instance of some **NP**-complete problem, by definition.
- Key question: Are we actually using hard instances?

  $\rightsquigarrow$ Theory of **average-case hardness**.

On hardness (2)

Common claim:

"**NP**-hard problems are needed/good for cryptography."

Reality:

- Cryptography *does not care* about worst-case hardness.
- Anything in **NP** can be viewed as an instance of some **NP**-complete problem, by definition.
- Key question: Are we actually using hard instances?

  $\rightsquigarrow$ Theory of **average-case hardness**.

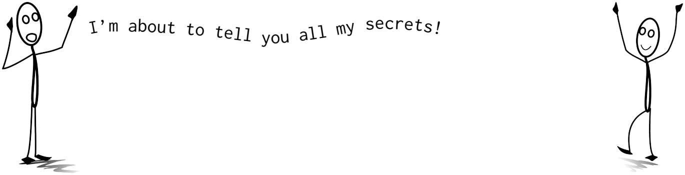
- The problems mainly used in contemporary public-key cryptography are in fact unlikely to be **NP**-hard!

Key agreement over an insecure channel

Key agreement over an insecure channel

Alice and Bob would like to establish a shared secret between them via an insecure channel.

(They can then use symmetric cryptography to communicate securely.)



Key agreement over an insecure channel

Alice and Bob would like to establish a shared secret between them via an insecure channel.

(They can then use symmetric cryptography to communicate securely.)

Key agreement over an insecure channel

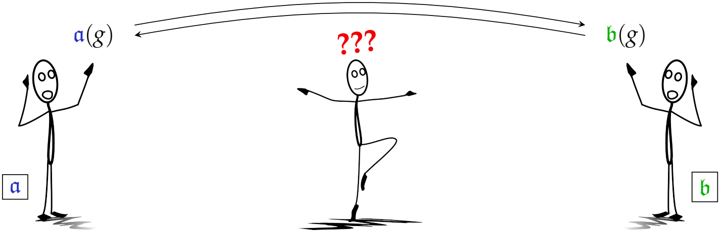
Alice and Bob would like to establish a shared secret between them via an insecure channel.

(They can then use symmetric cryptography to communicate securely.)

Key agreement over an insecure channel

Alice and Bob would like to establish a shared secret between them via an insecure channel.

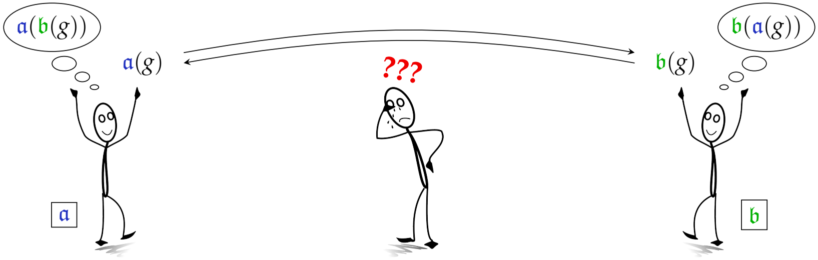
(They can then use symmetric cryptography to communicate securely.)



Key agreement over an insecure channel

Alice and Bob would like to establish a shared secret between them via an insecure channel.

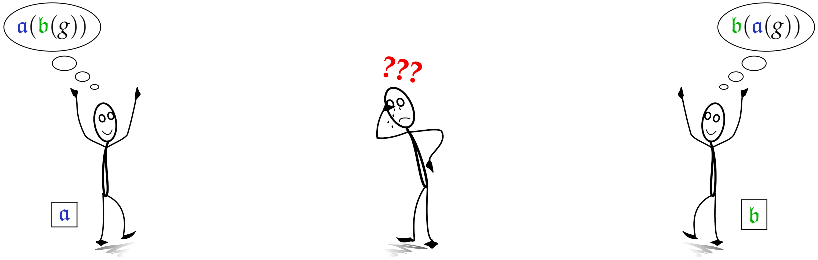
(They can then use symmetric cryptography to communicate securely.)



Key agreement over an insecure channel

Alice and Bob would like to establish a shared secret between them via an insecure channel.

(They can then use symmetric cryptography to communicate securely.)



Key agreement: First ideas

- Need: *Commuting* functions $\mathfrak{a}$ and $\mathfrak{b}$ that are hard to invert.

Key agreement: First ideas

- Need: *Commuting* functions $\mathfrak{a}$ and $\mathfrak{b}$ that are hard to invert.
- Idea: Simply multiply by a secret number?
  This "works" ($g \cdot a \cdot b = g \cdot b \cdot a$), but it's obviously insecure.
  (Attackers can simply compute $(g \cdot a)/g = a$.)

Key agreement: First ideas

- Need: *Commuting* functions $\mathfrak{a}$ and $\mathfrak{b}$ that are hard to invert.
- Idea: Simply multiply by a secret number?
  This "works" ($g \cdot a \cdot b = g \cdot b \cdot a$), but it's obviously insecure.
  (Attackers can simply compute $(g \cdot a)/g = a$.)
- Better idea: exponentiate to a secret power?
  This "works" ($(g^a)^b = (g^b)^a$), but with real numbers it's either clearly impossible to do this efficiently or insecure.

Key agreement: First ideas

- Need: *Commuting* functions $\mathfrak{a}$ and $\mathfrak{b}$ that are hard to invert.
- Idea: Simply multiply by a secret number?
  This "works" ($g \cdot a \cdot b = g \cdot b \cdot a$), but it's obviously insecure.
  (Attackers can simply compute $(g \cdot a)/g = a$.)
- Better idea: exponentiate to a secret power?
  This "works" ($(g^a)^b = (g^b)^a$), but with real numbers it's either clearly impossible to do this efficiently or insecure.
- Excellent idea: Do it in finite algebraic structures.
  This still "works", and can be secure and efficient.

Diffie–Hellman key agreement (1976)

Forever fixed, public system parameters:

- A large prime p of the form $p = 2\ell + 1$ with ℓ prime.
- An element $g \in \mathbb{F}_p^\times$ of order ℓ.

Diffie–Hellman key agreement (1976)

Forever fixed, public system parameters:

- A large prime p of the form $p = 2\ell + 1$ with ℓ prime.
- An element $g \in \mathbb{F}_p^\times$ of order ℓ.

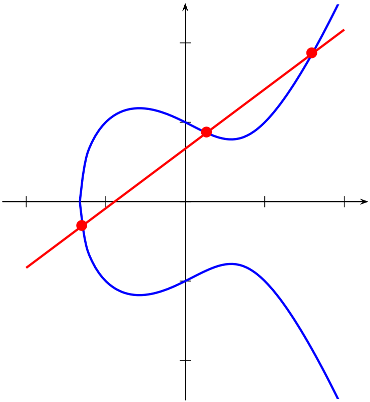
Alice	public	Bob
$a \leftarrow \{0, \ldots, \ell-1\}$.		$b \leftarrow \{0, \ldots, \ell-1\}$.
g^a	$\longrightarrow$ / $\longleftarrow$	g^b
Compute $(g^b)^a$.		Compute $(g^a)^b$.

Using elliptic curves instead of exponentiation

- Modern cryptographic reality: Elliptic curves are better.
- They are abelian groups $\rightsquigarrow$ everything works “the same”.

Using elliptic curves instead of exponentiation

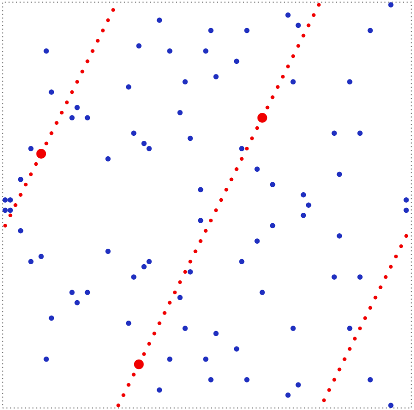
- Modern cryptographic reality: Elliptic curves are better.
- They are abelian groups $\rightsquigarrow$ everything works "the same".



The elliptic curve $y^2 = x^3 - x + 1$ over $\mathbb{R}$.

Using elliptic curves instead of exponentiation

- Modern cryptographic reality: Elliptic curves are better.
- They are abelian groups $\rightsquigarrow$ everything works "the same".



The elliptic curve $y^2 = x^3 - x + 1$ over the finite field $\mathbb{F}_{79}$.

This talk

Why cryptography?

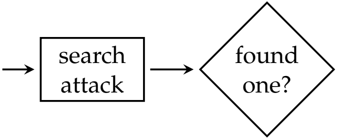
The quantum threat

Post-quantum everything

Highlight: Isogenies

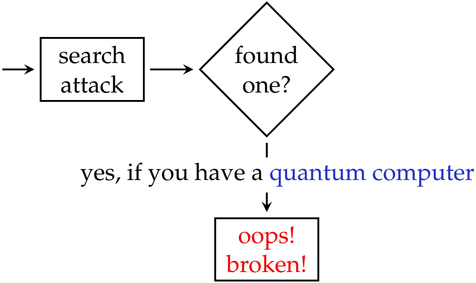
Enter quantum.

Have: Supposedly hard computational problem.



Enter quantum.

Have: Supposedly hard computational problem.



What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.

What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation "by hand": Interaction between brain and pen and paper.

What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation "by hand": Interaction between brain and pen and paper.
  - 

What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation “by hand”: Interaction between brain and pen and paper.
  - Mechanical calculation device: Classical mechanics — gears etc.

What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation “by hand”: Interaction between brain and pen and paper.
  - Mechanical calculation device: Classical mechanics — gears etc.
  - Pocket calculator/laptop: Electronics of silicon-based semiconductors.

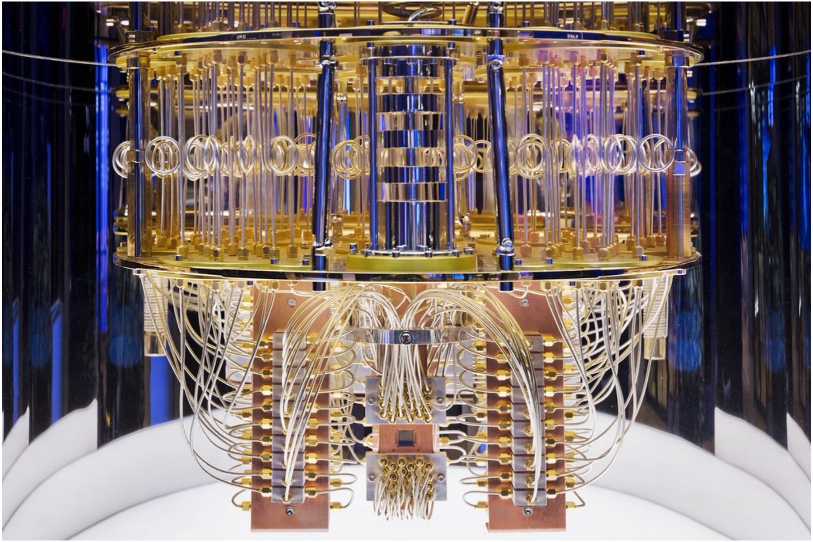
What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation "by hand": Interaction between brain and pen and paper.
  - Mechanical calculation device: Classical mechanics — gears etc.
  - Pocket calculator/laptop: Electronics of silicon-based semiconductors.
  - Quantum computer: Quantum-mechanical properties of particles.

What are computers, really?

- Computing essentially means manipulating and exploiting real-world physical processes to find some desired answer.
  - Calculation "by hand": Interaction between brain and pen and paper.
  - Mechanical calculation device: Classical mechanics — gears etc.
  - Pocket calculator/laptop: Electronics of silicon-based semiconductors.
  - Quantum computer: Quantum-mechanical properties of particles.

⇝ Quantum computers are just "the next evolution" of using an increasingly bigger share of physics to compute things.



Quantum computers: Core concepts

- "Classical" computer: Elementary unit is bit, can be 0 or 1.

Quantum computers: Core concepts

- "Classical" computer: Elementary unit is bit, can be 0 or 1.
- Quantum computer: Elementary unit is qubit, can store an infinite set of values "between" 0 and 1.

Quantum computers: Core concepts

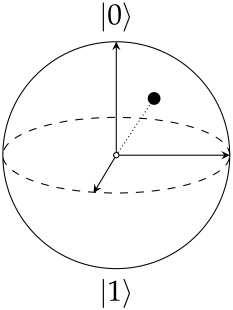
- "Classical" computer: Elementary unit is bit, can be 0 or 1.
- Quantum computer: Elementary unit is qubit, can store an infinite set of values "between" 0 and 1. They look like $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$.

  $\rightsquigarrow$ Single-qubit states are unit vectors in $\mathbb{C}^2$.

Quantum computers: Core concepts

- "Classical" computer: Elementary unit is bit, can be 0 or 1.
- Quantum computer: Elementary unit is qubit, can store an infinite set of values "between" 0 and 1. They look like $\alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ with $|\alpha|^2 + |\beta|^2 = 1$.

  $\rightsquigarrow$ Single-qubit states are unit vectors in $\mathbb{C}^2$.



Quantum computers: Core concepts

- Measuring the qubit $\alpha|0\rangle + \beta|1\rangle$ is probabilistic: We get
  - $|0\rangle$ with probability $|\alpha|^2$;
  - $|1\rangle$ with probability $|\beta|^2$.

Quantum computers: Core concepts

- Measuring the qubit $\alpha|0\rangle + \beta|1\rangle$ is probabilistic: We get
  - $|0\rangle$ with probability $|\alpha|^2$;
  - $|1\rangle$ with probability $|\beta|^2$.

  Afterwards, the qubit remains in that state: either $|0\rangle$ or $|1\rangle$.

Quantum computers: Core concepts

- Measuring the qubit $\alpha|0\rangle + \beta|1\rangle$ is probabilistic: We get
  - $|0\rangle$ with probability $|\alpha|^2$;
  - $|1\rangle$ with probability $|\beta|^2$.

  Afterwards, the qubit remains in that state: either $|0\rangle$ or $|1\rangle$.

$\rightsquigarrow$ Measuring cannot tell us exactly what α, β were.

Quantum computers: Core concepts

- Measuring the qubit $\alpha|0\rangle + \beta|1\rangle$ is probabilistic: We get
  - $|0\rangle$ with probability $|\alpha|^2$;
  - $|1\rangle$ with probability $|\beta|^2$.

  Afterwards, the qubit remains in that state: either $|0\rangle$ or $|1\rangle$.

$\rightsquigarrow$ Measuring cannot tell us exactly what α, β were.

- But we can carefully manipulate a quantum state into something whose measurement outcome will be useful!

Quantum computers: Core concepts

- Measuring the qubit $\alpha|0\rangle + \beta|1\rangle$ is probabilistic: We get
  - $|0\rangle$ with probability $|\alpha|^2$;
  - $|1\rangle$ with probability $|\beta|^2$.

  Afterwards, the qubit remains in that state: either $|0\rangle$ or $|1\rangle$.

$\rightsquigarrow$ Measuring cannot tell us exactly what α, β were.

- But we can carefully manipulate a quantum state into something whose measurement outcome will be useful!

"Quantum states are like a box of chocolates.
You never know what you're gonna get."

— F. Gump, probably

Quantum computers: Core concepts

- Key fact: Combining multiple qubits allows entanglement.

Quantum computers: Core concepts

- Key fact: Combining multiple qubits allows entanglement.
- Naïve juxtaposition: State space would be $(\mathbb{C}^2)^n \cong \mathbb{C}^{2n}$.

Quantum computers: Core concepts

- Key fact: Combining multiple qubits allows entanglement.
- Naïve juxtaposition: State space would be $(\mathbb{C}^2)^n \cong \mathbb{C}^{2n}$.
- Physical reality: Combining qubits gives state space $\mathbb{C}^{2^n}$!
  Mathematically, this new state space is a tensor product $(\mathbb{C}^2)^{\otimes n}$.

Quantum computers: Core concepts

- Key fact: Combining multiple qubits allows entanglement.
- Naïve juxtaposition: State space would be $(\mathbb{C}^2)^n \cong \mathbb{C}^{2n}$.
- Physical reality: Combining qubits gives state space $\mathbb{C}^{2^n}$!
  Mathematically, this new state space is a tensor product $(\mathbb{C}^2)^{\otimes n}$.
- This allows for entangled states such as $\frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$.

Quantum computers: Core concepts

- Key fact: Combining multiple qubits allows entanglement.
- Naïve juxtaposition: State space would be $(\mathbb{C}^2)^n \cong \mathbb{C}^{2n}$.
- Physical reality: Combining qubits gives state space $\mathbb{C}^{2^n}$!
  Mathematically, this new state space is a tensor product $(\mathbb{C}^2)^{\otimes n}$.
- This allows for entangled states such as $\frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$.

This does **not** mean that "quantum computers can simply search through all secret keys simultaneously"!

Enter *post*-quantum. (1)

Common misconception:

«Quantum computers massively speed up ***all*** computations.

Enter *post*-quantum. (1)

Common misconception:

«Quantum computers massively speed up ***all*** computations.
Therefore, cryptography is doomed, and all hope is lost.»

Enter *post*-quantum. (1)

Common misconception:

«Quantum computers massively speed up ***all*** computations.
Therefore, cryptography is doomed, and all hope is lost.»

Not true at all!

Quantum computers struggle with plenty of tasks.

Enter *post*-quantum. (1)

Common misconception:

«Quantum computers massively speed up ***all*** computations.
Therefore, cryptography is doomed, and all hope is lost.»

Not true at all!

Quantum computers struggle with plenty of tasks.

Quantum cryptanalysis

Of primary relevance to cryptography are three algorithms:

Quantum cryptanalysis

Of primary relevance to cryptography are three algorithms:

- Grover's algorithm: Given a function $f : \{0,1\}^n \to \{0,1\}$ such that $\exists!\, x \in \{0,1\}^n$ with $f(x) = 1$, find that x.
  - !! Square-root complexity: from $O(2^n)$ to $O(2^{n/2})$.

Quantum cryptanalysis

Of primary relevance to cryptography are three algorithms:

- Grover's algorithm: Given a function $f : \{0,1\}^n \to \{0,1\}$ such that $\exists!\, x \in \{0,1\}^n$ with $f(x) = 1$, find that x.
  - !! Square-root complexity: from $O(2^n)$ to $O(2^{n/2})$.
- Shor's algorithm: Given a periodic function $f : \mathbb{Z}^r \to S$, find (a description of) the set of period vectors.
  - !! Polynomial-time complexity. (More on the next slide.)

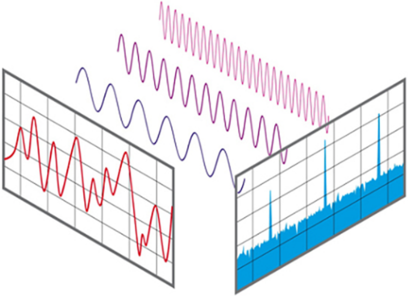
Quantum cryptanalysis

Of primary relevance to cryptography are three algorithms:

- Grover's algorithm: Given a function $f : \{0,1\}^n \to \{0,1\}$ such that $\exists!\, x \in \{0,1\}^n$ with $f(x) = 1$, find that x.
  - !! Square-root complexity: from $O(2^n)$ to $O(2^{n/2})$.
- Shor's algorithm: Given a periodic function $f : \mathbb{Z}^r \to S$, find (a description of) the set of period vectors.
  - !! Polynomial-time complexity. (More on the next slide.)
- Kuperberg's algorithm: Given two functions $f_1, f_2 : G \to S$ such that $\exists!\, s \in G$ with $f_2(x) = f_1(x+s)$ for all x, find that s.
  - !! Subexponential complexity: from $|G|^{O(1)}$ to $2^{O(\sqrt{\log|G|})}$.

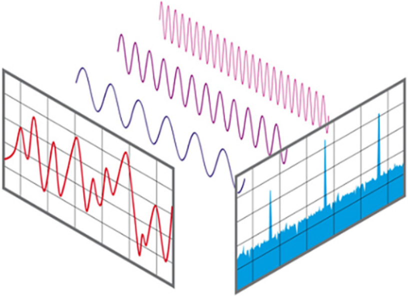
Shor's algorithm

Key idea: **Q**uantum **F**ourier **T**ransform.



Shor's algorithm

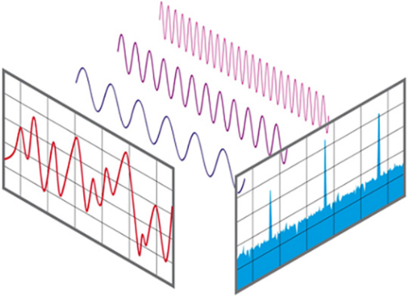
Key idea: **Q**uantum **F**ourier **T**ransform.



- Prepare a quantum state encoding a preimage set $f^{-1}(y)$.

Shor’s algorithm

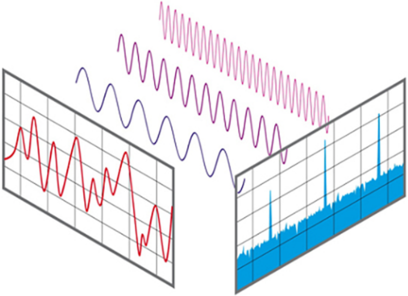
Key idea: **Quantum Fourier Transform**.



- Prepare a quantum state encoding a preimage set $f^{-1}(y)$.
- Apply QFT to obtain a quantum state encoding the period.

Shor's algorithm

Key idea: **Quantum Fourier Transform**.



- Prepare a quantum state encoding a preimage set $f^{-1}(y)$.
- Apply QFT to obtain a quantum state encoding the period.
- Then measure to get a random period vector. Repeat.

Shor vs. DLP

The **d**iscrete **l**ogarithm **p**roblem: Given g and $h = g^a$, find a.

(Here g, h are elements of a finite group G and a is an integer.)

Shor vs. DLP

The **d**iscrete **l**ogarithm **p**roblem: Given g and $h = g^a$, find a.

(Here g, h are elements of a finite group G and a is an integer.)

- Define the map $f\colon \ \mathbb{Z}^2 \to G, \ (x, y) \mapsto g^x \cdot h^y$.

Shor vs. DLP

The **d**iscrete **l**ogarithm **p**roblem: Given g and $h = g^a$, find a.

(Here g, h are elements of a finite group G and a is an integer.)

- Define the map $f\colon\ \mathbb{Z}^2 \to G,\ (x, y) \mapsto g^x \cdot h^y$.
- Find the period vector $(a, -1)$ using Shor's algorithm.

Shor vs. DLP

The **d**iscrete **l**ogarithm **p**roblem: Given g and $h = g^a$, find a.

(Here g, h are elements of a finite group G and a is an integer.)

- Define the map $f\colon\ \mathbb{Z}^2 \to G,\ (x, y) \mapsto g^x \cdot h^y$.
- Find the period vector $(a, -1)$ using Shor's algorithm.

Time complexity: $(\log|G|)^{O(1)}$. Generic classical algorithms: $\Omega(\sqrt{|G|})$.

Shor vs. DLP

The **d**iscrete **l**ogarithm **p**roblem: Given g and $h = g^a$, find a.

(Here g, h are elements of a finite group G and a is an integer.)

- Define the map $f\colon \mathbb{Z}^2 \to G,\ (x, y) \mapsto g^x \cdot h^y$.
- Find the period vector $(a, -1)$ using Shor's algorithm.

Time complexity: $(\log|G|)^{O(1)}$. Generic classical algorithms: $\Omega(\sqrt{|G|})$.

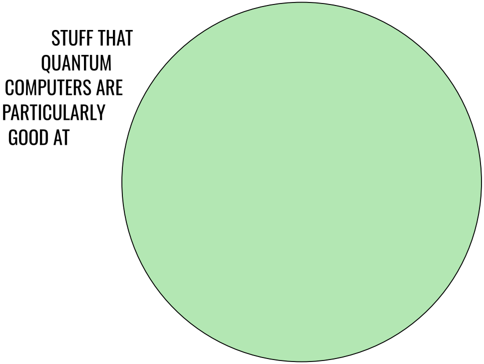
Exponential speedup.

Enter *post*-quantum. (2)

Unfortunate coincidence *(or is it?)*:

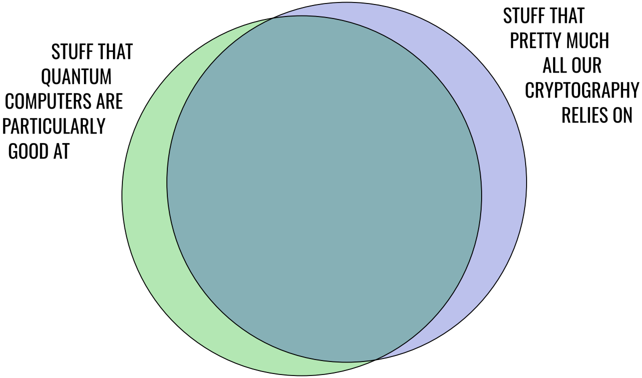
Enter *post*-quantum. (2)

<u>Unfortunate coincidence *(or is it?)*</u>:



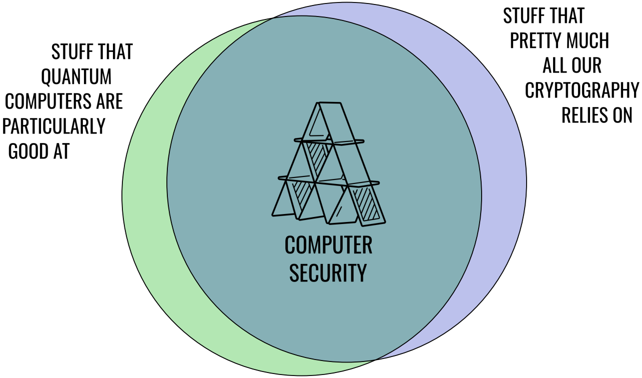
Enter *post*-quantum. (2)

Unfortunate coincidence *(or is it?)*:



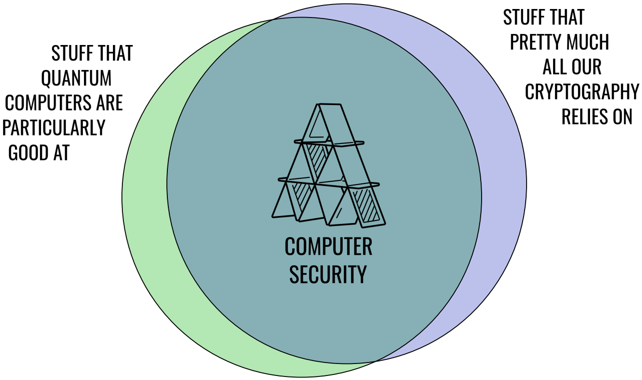
Enter *post*-quantum. (2)

Unfortunate coincidence *(or is it?)*:



Enter *post*-quantum. (2)

Unfortunate coincidence *(or is it?)*:



- Note: Public-key cryptography sustains much more damage from quantum attacks than symmetric cryptography.

This talk

Why cryptography?

The quantum threat

Post-quantum everything

Highlight: Isogenies

Post-quantum cryptography (PQC)

...substitutes quantum-weak building blocks by
quantum-resistant alternatives.

Note on "quantum cryptography"

- Post-quantum cryptography is not to be confused with "quantum cryptography".

Note on "quantum cryptography"

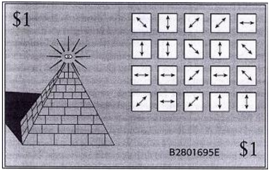
- Post-quantum cryptography is not to be confused with "quantum cryptography".
- PQC runs on classical computers. Only the attacker is assumed to have a quantum computer.

Note on "quantum cryptography"

- Post-quantum cryptography is not to be confused with "quantum cryptography".
- PQC runs on classical computers. Only the attacker is assumed to have a quantum computer.
- In quantum cryptography, all users need quantum devices!

Note on "quantum cryptography"

- Post-quantum cryptography is <u>not to be confused</u> with "quantum cryptography".
- PQC runs on classical computers.
  <u>Only the attacker</u> is assumed to have a quantum computer.
- In quantum cryptography, all users need quantum devices!



Note on “quantum cryptography”

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Federal Office for Information Security

General Intelligence and Security Service
Ministry of the Interior and Kingdom Relations

SWEDISH ARMED FORCES

Position Paper on Quantum Key Distribution

French Cybersecurity Agency (ANSSI)

Federal Office for Information Security (BSI)

Netherlands National Communications Security Agency (NLNCSA)

Swedish National Communications Security Authority, Swedish Armed Forces

Note on “quantum cryptography”

Executive summary

Quantum Key Distribution (QKD) seeks to leverage quantum effects in order for two remote parties to agree on a secret key via an insecure quantum channel. This technology has received significant attention, sometimes claiming unprecedented levels of security against attacks by both classical and quantum computers.

Due to current and inherent limitations, QKD can however currently only be used in practice in some niche use cases. For the vast majority of use cases where classical key agreement schemes are currently used it is not possible to use QKD in practice. Furthermore, QKD is not yet sufficiently mature from a security perspective. In light of the urgent need to stop relying only on quantum-vulnerable public-key cryptography for key establishment, the clear priorities should therefore be the migration to post-quantum cryptography and/or the adoption of symmetric keying.

This paper is aimed at a general audience. Technical details have therefore been left out to the extent possible. Technical terms that require a definition are printed in italics and are explained in a glossary at the end of the document.

The post-quantum zoo

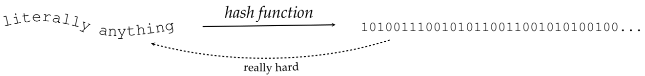
- PQC uses alternative hardness assumptions
  based on various (exciting!) types of mathematics.

The post-quantum zoo

- PQC uses alternative hardness assumptions
  based on various (exciting!) types of mathematics.

Hash-based signatures

Hash functions are random-looking functions that compress arbitrary data to short bitstrings. They should be hard to invert.



An individual can tie a hash value to their identity and later identify themself by revealing the corresponding input.
Selectively revealing inputs depending on a message leads to a signature scheme.

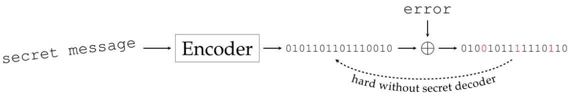
The post-quantum zoo

- PQC uses alternative hardness assumptions
  based on various (exciting!) types of mathematics.

Code-based crypto

Main application: **Encryption**.

Underlying problem: Correct errors in a codeword of a random-looking code.



Oldest proposal: McEliece 1978. Still *essentially unbroken* [2].

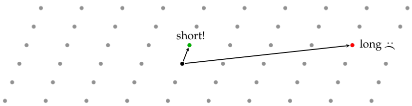
The post-quantum zoo

- PQC uses alternative hardness assumptions based on various (exciting!) types of mathematics.

Lattice-based crypto

Main applications: **Encryption**, **signatures**, and beyond.

Underlying problem: Find short vectors in a discrete additive subgroup of $\mathbb{R}^n$.



The post-quantum zoo

- PQC uses alternative hardness assumptions based on various (exciting!) types of mathematics.

Multivariate crypto

Main application: **Signatures**.

Underlying problem: Solve systems of quadratic equations over a finite field.

$$
\begin{aligned}
10x^2 + 15z^2 + 19xy + 7xz + 27yz + 20x + y &\equiv 14 \pmod{31}\\
25x^2 + 30y^2 + 17z^2 + 30xy + 23xz + 27yz + 15x + 4y + 16z &\equiv 5 \pmod{31}\\
15x^2 + 9y^2 + 11z^2 + 18xy + 24xz + 16yz + 28x + 9y + 3z &\equiv 6 \pmod{31}\\
27x^2 + 10y^2 + 17z^2 + 7xz + 28yz + 4x + 13y + 27z &\equiv 12 \pmod{31}
\end{aligned}
$$

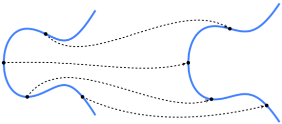
The post-quantum zoo

- PQC uses alternative hardness assumptions
  based on various (exciting!) types of mathematics.

Isogeny-based crypto

Main application: **Key exchange**, **signatures**.

Underlying problem: Find an isogeny between two elliptic curves.
An *isogeny* is a surjective group homomorphism given by rational functions.



Shortcomings of PQC

Shortcomings of PQC

The good news:
There *are* plausible PQC replacements for most cryptography.

Shortcomings of PQC

The good news:
There *are* plausible PQC replacements for most cryptography.

The bad news: PQC is typically slower, bigger, or less flexible.

Shortcomings of PQC

The <u>good news</u>:
There *are* plausible PQC replacements for most cryptography.

The <u>bad news</u>: PQC is typically slower, bigger, or less flexible.

pre-quantum	post-quantum
6ee2da7b68b7a997e062d09d94c1c76de61b5c260a35273713ddcc29e09ac840	45c83435071624067d69587335b97bf564929709c8825a004b028ae09c40980a 07e8d4bd604527ee221e8bac67d34cbe762c26df8453aae8b8c82b59c51a8552 6aba8ddc4b5f63cf69a5b367d3153e460f497a209c495fca318862d6a5780086 5479a006012d82f7212b40284d310e01bcb11e122c1fd303e441807849a7ea47 976a99abb7ccc4b674ad66f68eca195789b277d23c3d67bc418ca7c908b21e53 984983ba0205e4689000ace97238b3699016fa95e7a3a59cec0be81363852756 2fa9bf10d715e7505f6e1c1433521a918a7df52760a0d8a9549569f10827c423 cddff82aae01a90111395487b9c82b7b5a7978d789679e66b75087bfbff0569f c94e94f93531b721315926388431f2a36ae0f701bac254befb437c58641d4560 c8738a98f30918945db0a6900ad2c2abfc3e0f4786a4555639d84dcdd031d8f0 508d8c774d68298bcac4f42c6a7ff585af491fa7d7c3bbb41727699ebb315c43 7b210d42626ebc66c916af1f3515374314e4f40309ca7289c7bc51c301d8180e dc792d4dd44c41b77bd47a972d8434a9f03bb3954236ec422be0c8e991a79af2 86b6a7c459a95ed44868ed8052f2db0f3741710228979507cff961564882b5ea 19515ee00d657c7141e9b05f9a24136a2f915620b664404b5397cc7842748973 d0716cc273b528d51383a63fc8a3c4a3b1a8bc965775d750add6996c929e29f4 1e42362a759baa76f5a3dc0552f1d83195960e45837901494a87f2a6dc3b5d8b 73a9695c1229a0c9bddb0b2d99aa350c6cac657745c1308af354e10595f3682a 34dc26d9d28e2e2c4634aca75e94384700c9c06b1bca348330ac1791fab14190 99cf1288283bab03dca09ab3593cf3b12739cb44c0c04c6b93d1ea831df6bcb8 807aa6aa8cbec64d749a9e47f851c47c6537e196f1fcc4d63b67d29a58e86b9a 72a199cbb793c5084e5bab20bd02289b4aaa64e4c119488531e8a651a3175014 8e1742c5390bb9995c123f3056ad44c476468ded4b88a49130e35b4b00803dd2 4718674ca708e436d5c15ee1d95367c623512653c83b27b41cb308f8c2929b19 3b5487a4ce6401ec27a1605f879e2d9c53bf27e165246401cad7840a077934b8

This talk

Why cryptography?

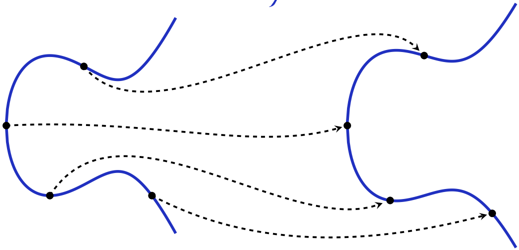
The quantum threat

Post-quantum everything

Highlight: Isogenies

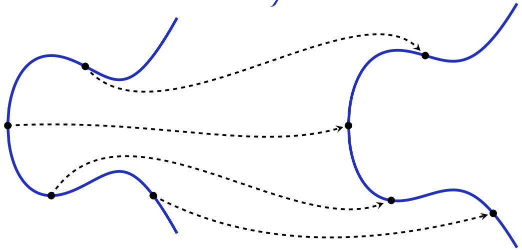
Isogenies of elliptic curves

- ...are essentially just *nice maps* between elliptic curves.

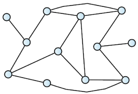


Isogenies of elliptic curves

- ...are essentially just *nice maps* between elliptic curves.

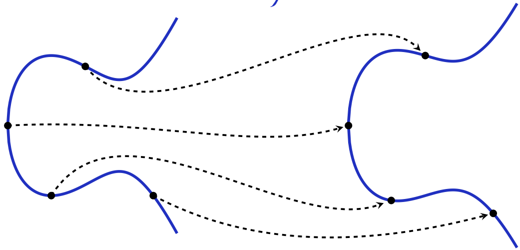


- They are a source of exponentially large graphs.

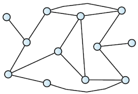


Isogenies of elliptic curves

- ...are essentially just *nice maps* between elliptic curves.

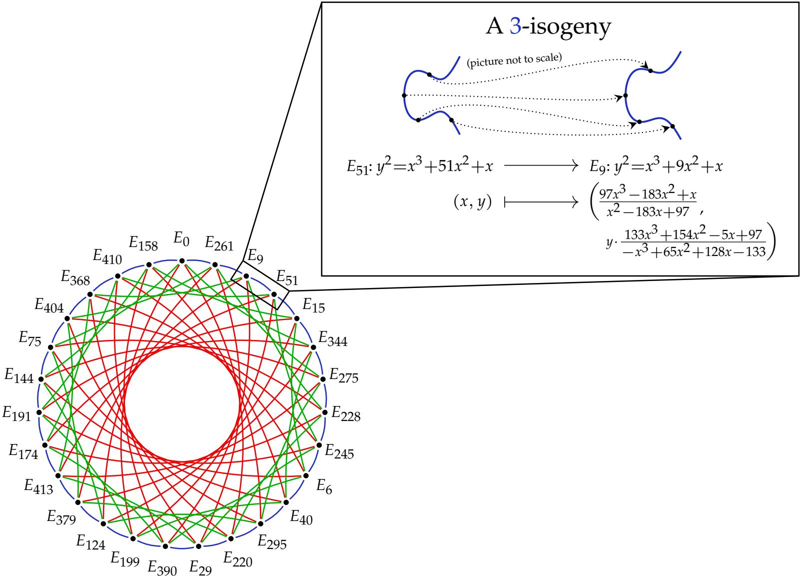


- They are a source of exponentially large graphs.



- ...with enough structure to navigate meaningfully!

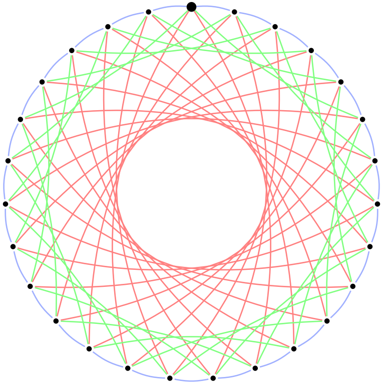
Graphs of elliptic curves



CSIDH [ˈsiːˌsaɪd] key exchange

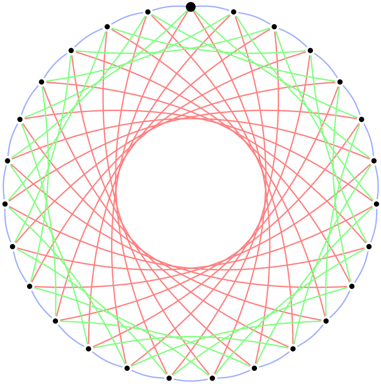
Alice

[+, +, −, −]



Bob

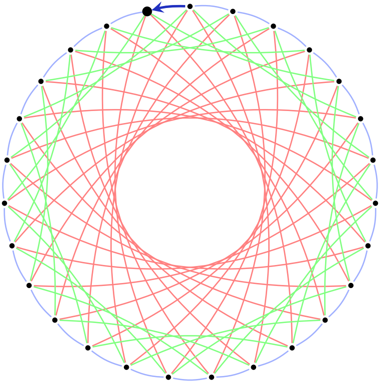
[−, +, −, −]



CSIDH [ˈsiːˌsaɪd] key exchange

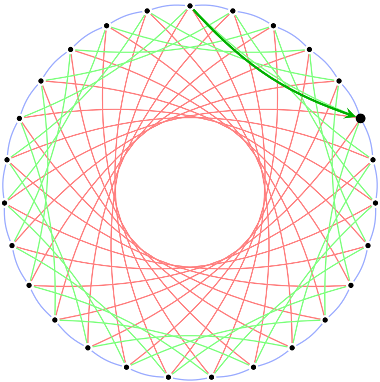
Alice

[+, +, −, −]
↑



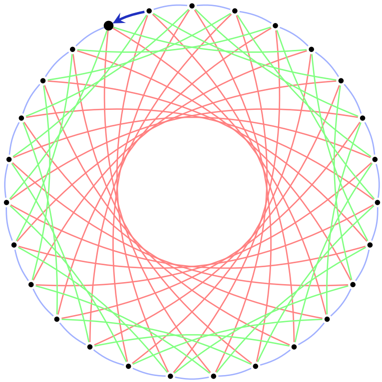
Bob

[−, +, −, −]
↑

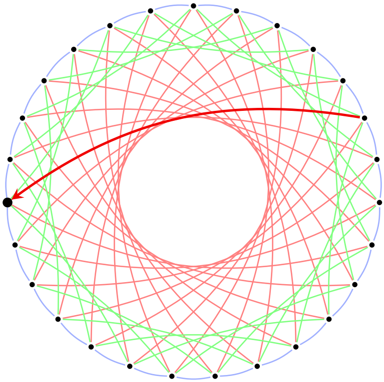


CSIDH [ˈsiːˌsaɪd] key exchange

Alice
[+, +, −, −]
↑



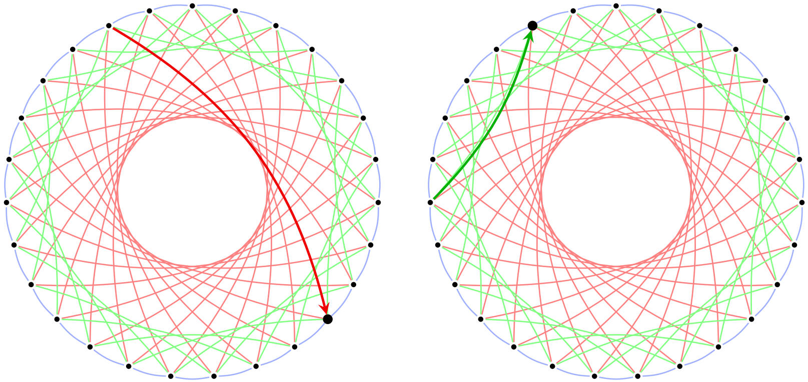
Bob
[−, +, −, −]
↑



CSIDH [ˈsiːˌsaɪd] key exchange

Alice
[+, +, −, −]
↑

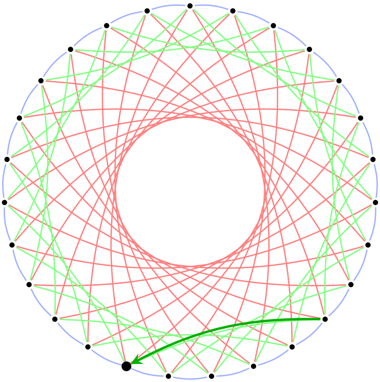
Bob
[−, +, −, −]
↑



CSIDH [ˈsiːˌsaɪd] key exchange

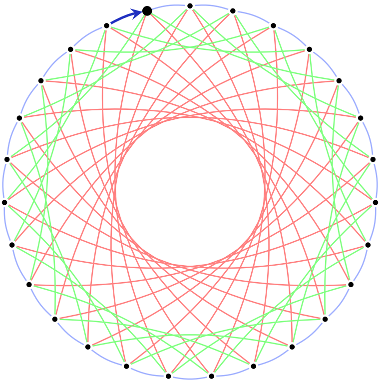
Alice

[+, +, −, −]
↑



Bob

[−, +, −, −]
↑



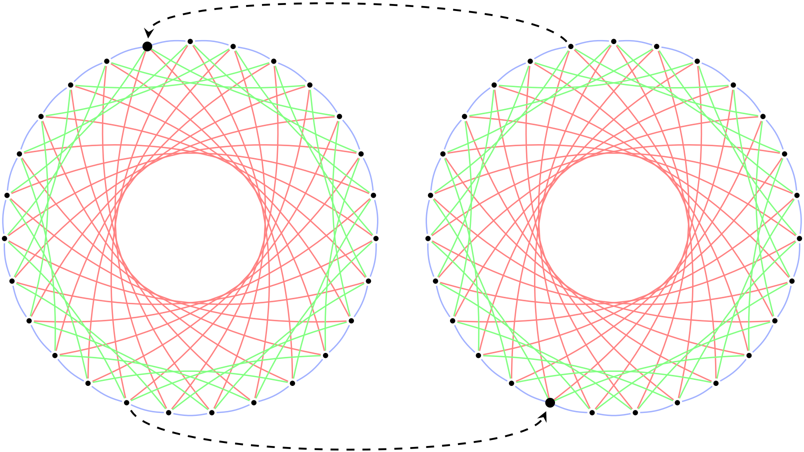
CSIDH [ˈsiːˌsaɪd] key exchange

Alice

[+, +, −, −]

Bob

[−, +, −, −]

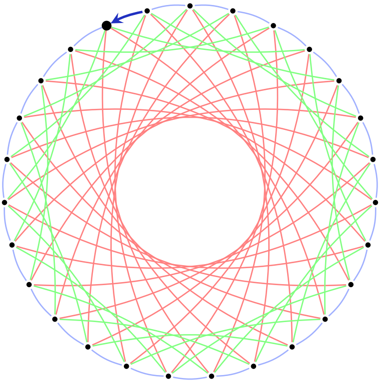


CSIDH [ˈsiːˌsaɪd] key exchange

Alice

[+, +, −, −]

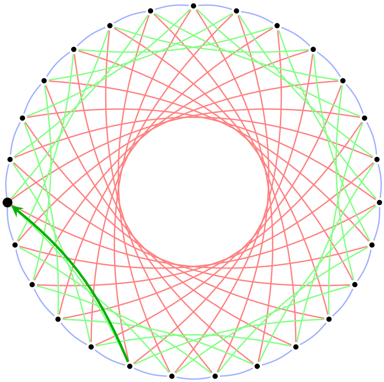
↑



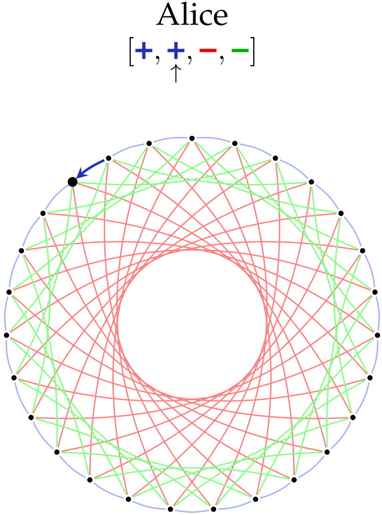
Bob

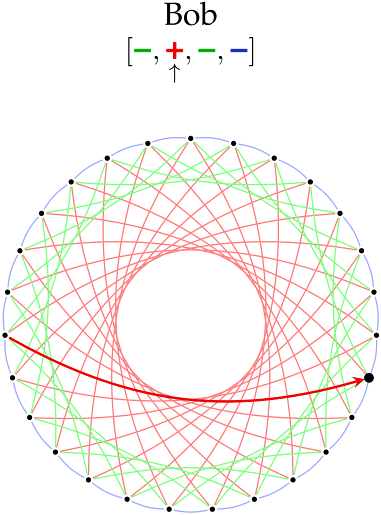
[−, +, −, −]

↑



CSIDH [ˈsiːˌsaɪd] key exchange



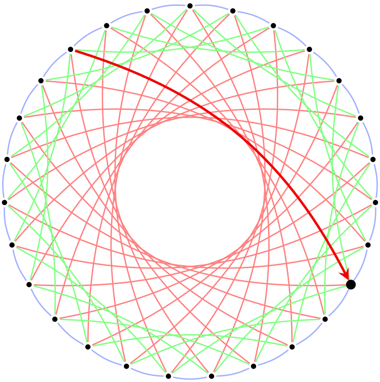


CSIDH [ˈsiːˌsaɪd] key exchange

Alice

[+, +, −, −]

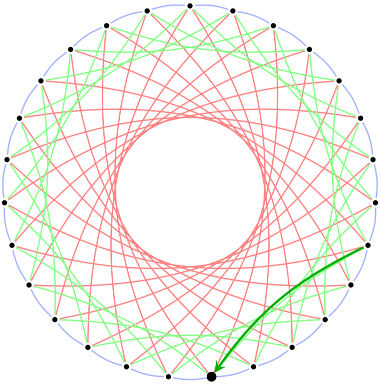
↑



Bob

[−, +, −, −]

↑



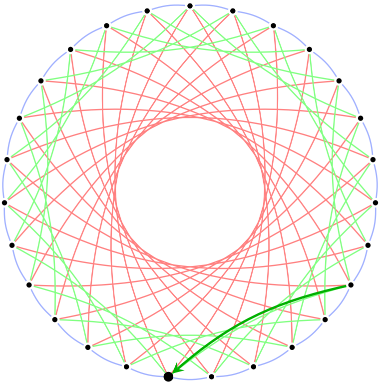
CSIDH [ˈsiːˌsaɪd] key exchange

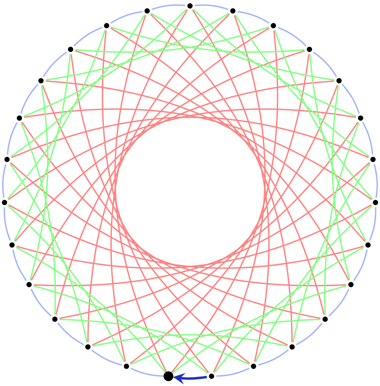
Alice

[**+**, **+**, **−**, **−**]
↑

Bob

[**−**, **+**, **−**, **−**]
↑

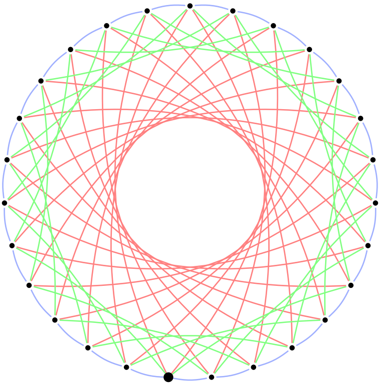




CSIDH [ˈsiːˌsaɪd] key exchange

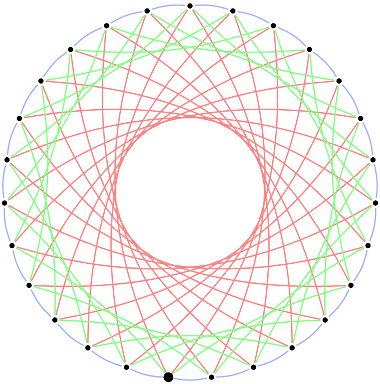
Alice

[+, +, −, −]

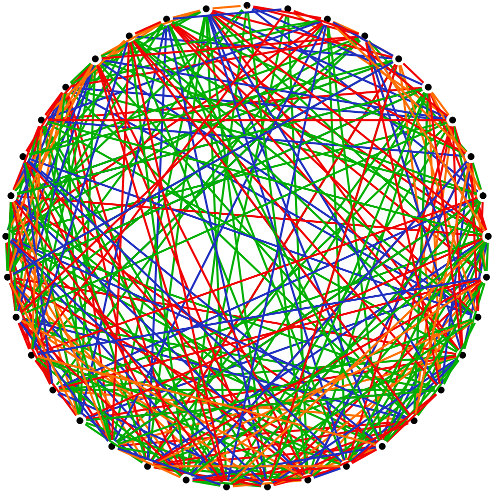


Bob

[−, +, −, −]



A much more random-looking isogeny graph



SQIsign

Isogeny graphs are not random graphs.

Lots of useful structure looming in the background.

SQIsign

Isogeny graphs are not random graphs.
Lots of useful structure looming in the background.

Deuring correspondence:

Almost exact equivalence between the worlds of maximal orders in certain quaternion algebras and of supersingular elliptic curves.

SQIsign

Isogeny graphs are not random graphs.
Lots of useful structure looming in the background.

Deuring correspondence:

> Almost exact equivalence between the worlds of maximal orders in certain quaternion algebras and of supersingular elliptic curves.

The correspondence is polynomial-time in the $\Rightarrow$ direction, but exponential-time in the $\Leftarrow$ direction. *Perfect for cryptography!*

SQIsign

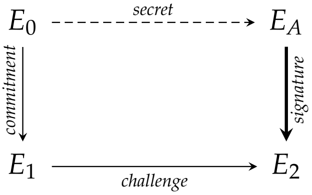
Isogeny graphs are not random graphs.
Lots of useful structure looming in the background.

Deuring correspondence:

Almost exact equivalence between the worlds of maximal orders in certain quaternion algebras and of supersingular elliptic curves.

The correspondence is polynomial-time in the $\Rightarrow$ direction, but exponential-time in the $\Leftarrow$ direction. *Perfect for cryptography!*

SQIsign is a signature scheme based on this one-wayness.



This talk

Why cryptography?

The quantum threat

Post-quantum everything

Highlight: Isogenies

Main goal: Solid foundations for computer security

- **Cryptanalysis!**
  - Algorithmic advances
  - Low-level programming (CPUs, GPUs)
  - Quantum algorithms

Main goal: Solid foundations for computer security

- **Cryptanalysis!**
  - Algorithmic advances
  - Low-level programming (CPUs, GPUs)
  - Quantum algorithms
- Number theory & algebraic geometry

Main goal: Solid foundations for computer security

- **Cryptanalysis!**
  - Algorithmic advances
  - Low-level programming (CPUs, GPUs)
  - Quantum algorithms
- Number theory & algebraic geometry
- Fast algorithms, computer algebra

  $\longrightarrow$ Open-source software

Main goal: Solid foundations for computer security

- **Cryptanalysis!**
  - Algorithmic advances
  - Low-level programming (CPUs, GPUs)
  - Quantum algorithms

- Number theory & algebraic geometry

- Fast algorithms, computer algebra

  $\longrightarrow$ Open-source software

- Implementations and side channels
  - Low-level programming (again)
  - High-assurance cryptography

Main goal: Solid foundations for computer security

- **Cryptanalysis!**
  - Algorithmic advances
  - Low-level programming (CPUs, GPUs)
  - Quantum algorithms

- Number theory & algebraic geometry

- Fast algorithms, computer algebra
  $\longrightarrow$ Open-source software

- Implementations and side channels
  - Low-level programming (again)
  - High-assurance cryptography

- Information security in general
  - Memory corruptions, reverse engineering, web hacking, ...