Lattices, Isogenies, and Codes

Contrasting Post-Quantum Design Directions
for Applied Cryptography

Maher Mamah

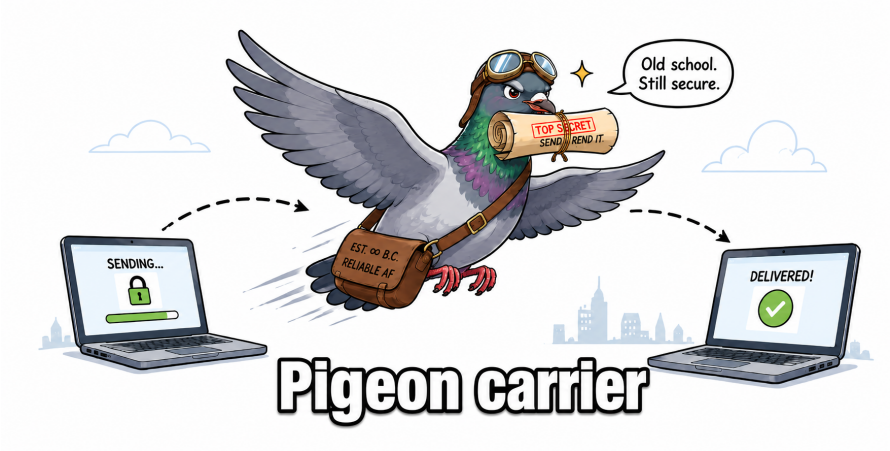
Cedarcrypt 2026

UNIVERSITY OF WATERLOO

What alternatives do we have when quantum computers break factoring and discrete logarithms?

And how do they differ in efficiency, security, and cryptographic functionality?

When RSA and ECC are broken



Design Directions

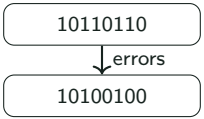
Lattices

Noise

structure + perturbation

Codes

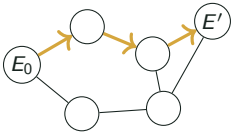
Decoding



recover original structure

Isogenies

Hidden paths



endpoints public, path hidden

Post-quantum cryptography is a design space, not a single replacement technology.

First assumption: trust noise

Lattice-based cryptography starts from a simple assumption:

Exact linear equations are easy.

Noisy linear equations can be hard.

This is the intuition behind **Learning With Errors (LWE)**.

LWE: exact equations become noisy equations

Choose a matrix $\mathbf{A}$
a secret $\mathbf{s}$
and small error $\mathbf{e}$

Linear algebra doesn't work here? ;(

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$$

Bob is given $\mathbf{A}$ and $\mathbf{b}$, and must recover $\mathbf{s}$.

Alice

Bob

example

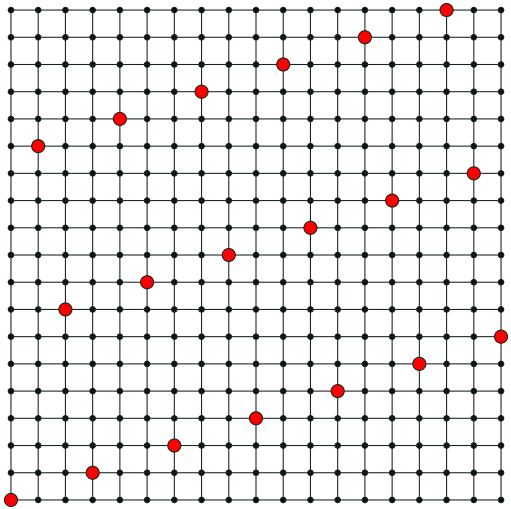


Figure: An illustration of the range of the tall matrix $A = \begin{pmatrix} 5 \\ 8 \end{pmatrix}$.

example

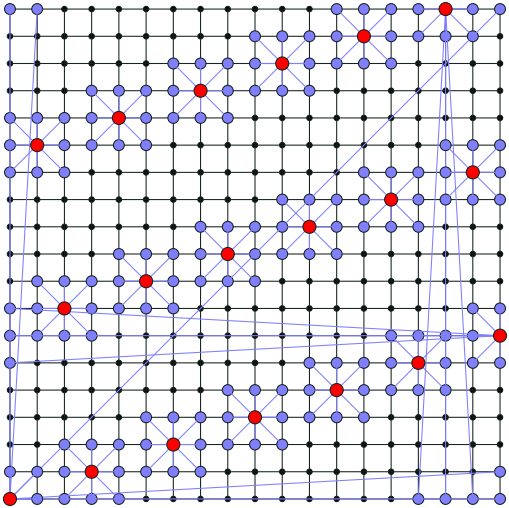
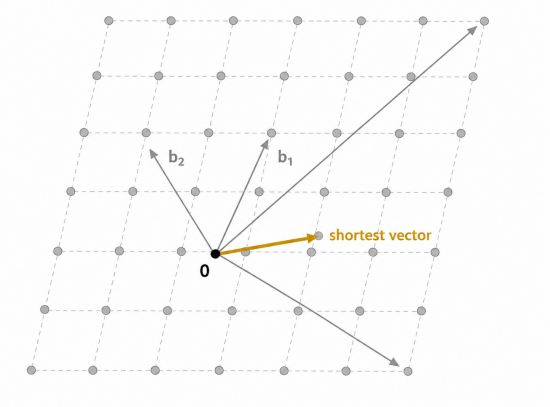


Figure: An illustration of the range of the tall matrix $A = \begin{pmatrix} 5 \\ 8 \end{pmatrix}$ with errors in $\{-1, 0, 1\}$.

The Shortest Vector Problem (SVP)

Problem

Given a lattice basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, find the non-zero shortest lattice vector in euclidean length $||v||$



A shortest nonzero vector in a 2D lattice.

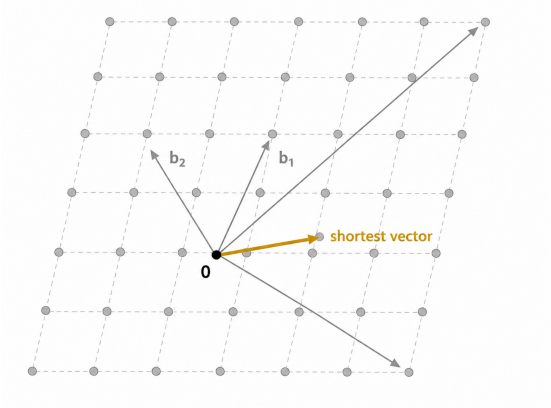
The Shortest Vector Problem (SVP)

Problem

Given a lattice basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, find the non-zero shortest lattice vector in euclidean length $||v||$

Why it matters

- Captures geometric hardness behind lattice cryptography.
- Short vectors become difficult to find in high dimension.
- Exact attacks are exponential; practical attacks use BKZ, enumeration, and sieving.



A shortest nonzero vector in a 2D lattice.

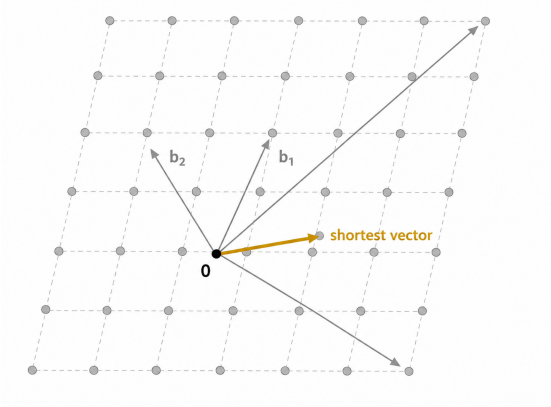
The Shortest Vector Problem (SVP)

Problem

Given a lattice basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, find the non-zero shortest lattice vector in euclidean length $||v||$

Why it matters

- Captures geometric hardness behind lattice cryptography.
- Short vectors become difficult to find in high dimension.
- Exact attacks are exponential; practical attacks use BKZ, enumeration, and sieving.



A shortest nonzero vector in a 2D lattice.

$$\boxed{\textit{Lattice sieving}: \ O(2^{0.292n+o(n)}) - \textit{time}}$$

Plain Kyber idea: Key generation

"I choose a secret s and add a little noise."

Alice

Key generation

Choose a public matrix $A \in \mathbb{Z}_q^{n \times m}$, a small secret s, and small noise e.

$$b = As + e \pmod{q}$$

$$\mathsf{pk} = (A, b), \qquad \mathsf{sk} = s$$

The pair (A, b) looks random to an attacker.

"All I see is the random stuff."

Bob

Plain Kyber idea: Encapsulation

Alice

Encapsulation

Sample a small random vector r, and compute

$$u = A^T r + e_1 \pmod q$$

$$v = b^T r + e_2 + \mu \left\lfloor \frac{q}{2} \right\rceil \pmod q$$

The bit $\mu \in \{0, 1\}$ is encoded by shifting v near either 0 or $q/2$.

Alice sends the noisy capsule (u, v) to Bob.

"I hide the bit μ inside a noisy capsule."

Bob

Plain Kyber idea: Decapsulation

"If you know the secret, you can clean it up."

Alice

Decapsulation

Using the secret s, compute

$$v - s^T u \approx \mu \left\lfloor \frac{q}{2} \right\rceil \pmod{q}$$

Since the remaining noise is small, rounding recovers μ, hence the shared key.

Shared key recovered!

"Subtract, round, done. Nice."

Bob

LARGE public keys and ciphertexts

Large matrices are costly

At Kyber-like security levels, an expanded LWE-style matrix lives in dimensions on the order of hundreds to about a thousand:

$$A \in \mathbb{Z}_q^{m \times n}, \qquad n, m \approx 500\text{–}1000.$$

Compactness through algebraic structure

Algebraic structure buys efficiency

Structure compresses

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{11}(x) & a_{12}(x) \\ a_{21}(x) & a_{22}(x) \end{pmatrix} \in R_q^{2\times 2}$$

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

Compactness through algebraic structure

Algebraic structure buys efficiency $\longleftrightarrow$ **but introduces symmetry**

Structure compresses

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{11}(x) & a_{12}(x) \\ a_{21}(x) & a_{22}(x) \end{pmatrix} \in R_q^{2\times 2}$$

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

- Replace large matrices with polynomial or module elements.

Compactness through algebraic structure

Algebraic structure buys efficiency $\longleftrightarrow$ **but introduces symmetry**

Structure compresses

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{11}(x) & a_{12}(x) \\ a_{21}(x) & a_{22}(x) \end{pmatrix} \in R_q^{2\times 2}$$

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

- Replace large matrices with polynomial or module elements.
- Use fast polynomial multiplication.

Compactness through algebraic structure

Algebraic structure buys efficiency $\longleftrightarrow$ **but introduces symmetry**

Structure compresses

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{11}(x) & a_{12}(x) \\ a_{21}(x) & a_{22}(x) \end{pmatrix} \in R_q^{2\times 2}$$

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

- Replace large matrices with polynomial or module elements.
- Use fast polynomial multiplication.
- Reduce public-key and ciphertext sizes.

Compactness through algebraic structure

Algebraic structure buys efficiency $\longleftrightarrow$ **but introduces symmetry**

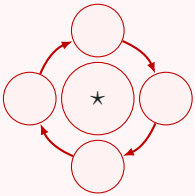
Structure compresses

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11}(x) & a_{12}(x) \\ a_{21}(x) & a_{22}(x) \end{pmatrix} \in$$

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

- Replace large matrices with polynomial or module elements.
- Use fast polynomial multiplication.
- Reduce public-key and ciphertext sizes.

Structure also reveals symmetry



?

- Extra structure may create new attack surfaces.
- Ideal and module lattices expose more symmetry than plain lattices.
- Security relies on hardness within a more structured family.

The design question

How much structure is safe?

Efficiency and size: lattices vs. classical public-key crypto

Key establishment

Scheme	Public data
X25519 / ECDH	32–65 B
RSA-2048/3072	256–384 B
ML-KEM-512	800 B pk, 768 B ct
ML-KEM-768	1184 B pk, 1088 B ct
ML-KEM-1024	1568 B pk, 1568 B ct

ML-KEM replaces quantum-vulnerable key exchange / key transport.

Efficiency and size: lattices vs. classical public-key crypto

Key establishment

Scheme	Public data
X25519 / ECDH	32–65 B
RSA-2048/3072	256–384 B
ML-KEM-512	800 B pk, 768 B ct
ML-KEM-768	1184 B pk, 1088 B ct
ML-KEM-1024	1568 B pk, 1568 B ct

ML-KEM replaces quantum-vulnerable key exchange / key transport.

Digital signatures

Scheme	Public key	Signature
ECDSA P-256	$\sim$ 64 B	$\sim$ 64 B
RSA-2048	256 B modulus	256 B
ML-DSA-44	1312 B	2420 B
ML-DSA-65	1952 B	3309 B
ML-DSA-87	2592 B	4627 B

ML-DSA replaces quantum-vulnerable signatures.

HAWK: flying high... and then under attack

Submitted to **NIST's additional call for digital signatures**.

Why HAWK looks great

- A **lattice-based signature** inspired by the **Lattice Isomorphism Problem**.
- Submitted to **NIST's additional signature call**.
- **Smaller public keys and signatures** than ML-DSA.
- **easier to deploy across hardware.**

Compact + elegant + hardware-friendly

HAWK: flying high... and then under attack

Submitted to **NIST's additional call for digital signatures**.

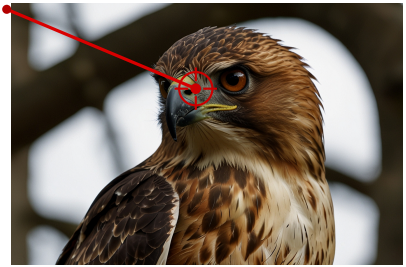
Why HAWK looks great

- A **lattice-based signature** inspired by the **Lattice Isomorphism Problem**.
- Submitted to **NIST's additional signature call**.
- **Smaller public keys and signatures** than ML-DSA.
- **easier to deploy across hardware.**

Compact + elegant + hardware-friendly

It was cool until everyone started shooting at the bird

cryptanalysis

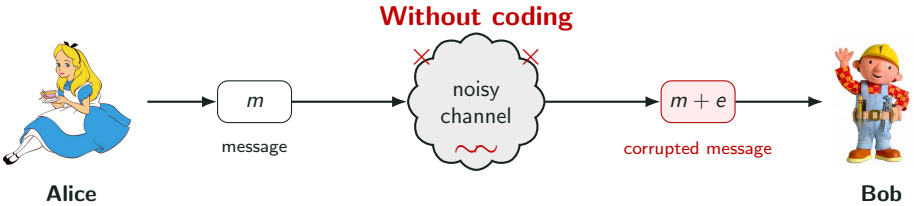


attacks

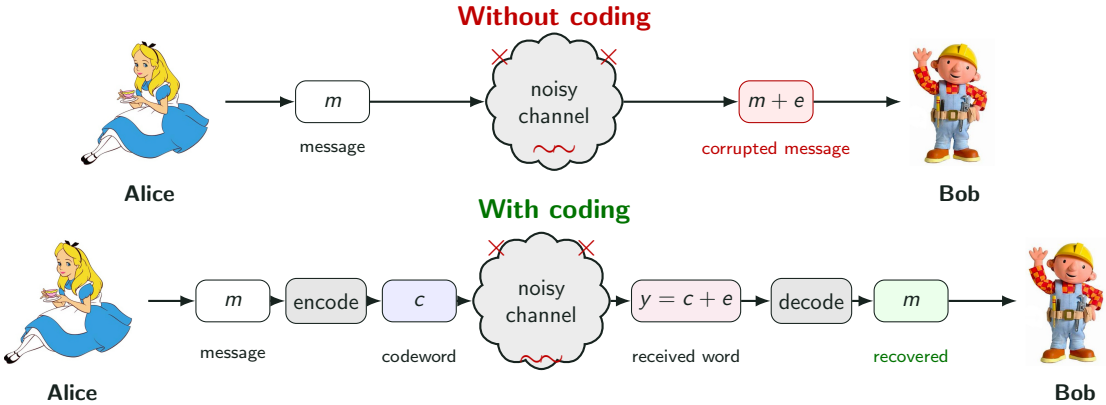
Codes

What if security comes from hiding information inside a noisy, redundant encoding?

Codes: Fancy names for vectors



Codes: Fancy names for vectors



Encoding allows Bob to recover the message even when it is transmitted over a noisy channel.

The decoding problem

$\mathbf{G} \in \mathbb{F}_q^{n \times k}$, $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{e}$ error vector with t non-zero entries,

Given $\mathbf{G}$ and $\mathbf{y}$, recover the unknown message $\mathbf{m}$ from

$$\mathbf{y} = \mathbf{G}\ \mathbf{m} + \mathbf{e} \qquad \text{(Decoding)}$$

The decoding problem

$\mathbf{G} \in \mathbb{F}_q^{n \times k}$, $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{e}$ error vector with t non-zero entries,

Given $\mathbf{G}$ and $\mathbf{y}$, recover the unknown message $\mathbf{m}$ from

$$\begin{array}{ccccccc} \mathbf{y} & = & \mathbf{G} & \mathbf{m} & + & \mathbf{e} \\ \downarrow & & \downarrow & \downarrow & & \downarrow \\ \mathbf{b} & = & \mathbf{A} & \mathbf{s} & + & \mathbf{e} \end{array} \qquad \text{(Decoding)}$$

The decoding problem

$\mathbf{G} \in \mathbb{F}_q^{n \times k}$, $\mathbf{m} \in \mathbb{F}_q^k$, $\mathbf{e}$ error vector with t non-zero entries,

Given $\mathbf{G}$ and $\mathbf{y}$, recover the unknown message $\mathbf{m}$ from

$$\begin{array}{ccccccc} \mathbf{y} & = & \mathbf{G} & \mathbf{m} & + & \mathbf{e} & \text{(Decoding)} \\ \downarrow & & \downarrow & \downarrow & & \downarrow & \\ \mathbf{b} & = & \mathbf{A} & \mathbf{s} & + & \mathbf{e} & \text{(LWE?)} \end{array}$$

The error $\mathbf{y} - \mathbf{G}\mathbf{m} = \mathbf{e}$ makes this a hard combinatorial problem

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).
- **Ciphertext:** a codeword plus random errors,

$$y = c + e.$$

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).
- **Ciphertext:** a codeword plus random errors,

$$y = c + e.$$

- **Private key:** an efficient algebraic decoder that removes e and recovers c.

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).
- **Ciphertext:** a codeword plus random errors,

$$y = c + e.$$

- **Private key:** an efficient algebraic decoder that removes e and recovers c.
- **Modern form:** Classic McEliece was submitted to NIST as a IND-CCA2-secure KEM.

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).
- **Ciphertext:** a codeword plus random errors,

$$y = c + e.$$

- **Private key:** an efficient algebraic decoder that removes e and recovers c.
- **Modern form:** Classic McEliece was submitted to NIST as a IND-CCA2-secure KEM.
- **Standardization:** McEliece has been recently standardized by ISO.

Robert McEliece (1942–2019)

McEliece cryptosystem

- **1978:** first code-based public-key cryptosystem.
- **Public key:** a disguised random-looking binary Goppa code (the matrix H).
- **Ciphertext:** a codeword plus random errors,

$$y = c + e.$$

- **Private key:** an efficient algebraic decoder that removes e and recovers c.
- **Modern form:** Classic McEliece was submitted to NIST as a IND-CCA2-secure KEM.
- **Standardization:** McEliece has been recently standardized by ISO.

Security record

McEliece has withstood nearly half a century of cryptanalysis, making it one of the most conservative and secure public-key encryption/KEM.

Classic McEliece: conservative, but public-key heavy

Deployment profile

Huge public keys, tiny ciphertexts, and a long cryptanalytic record.

Instance	Public key	Private key	Ciphertext	Session key
mceliece348864	261,120 B	6,492 B	96 B	32 B
mceliece460896	524,160 B	13,608 B	156 B	32 B
mceliece6688128	1,044,992 B	13,932 B	208 B	32 B
mceliece6960119	1,047,319 B	13,948 B	194 B	32 B
mceliece8192128	1,357,824 B	14,120 B	208 B	32 B

HQC: NIST's code-based KEM

What is HQC?

HQC is a code-based key encapsulation mechanism built from quasi-cyclic codes.
NIST selected HQC for standardization as a backup/diversity KEM alongside ML-KEM.

Instance	Encap. key	Decap. key	Ciphertext	Session key
HQC-1	2,241 B	2,321 B	4,433 B	32 B
HQC-3	4,514 B	4,602 B	8,978 B	32 B
HQC-5	7,237 B	7,333 B	14,421 B	32 B

Isogenies

What if security comes from navigating a graph
where the endpoints are visible,
but the path is hidden?

Elliptic curves

An elliptic curve satisfies $y^2 = x^3 + ax + b$ over a field.

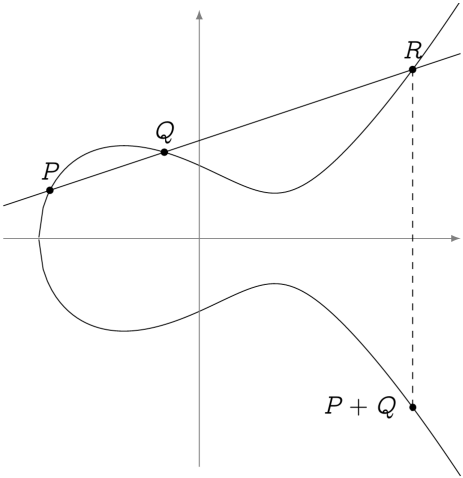
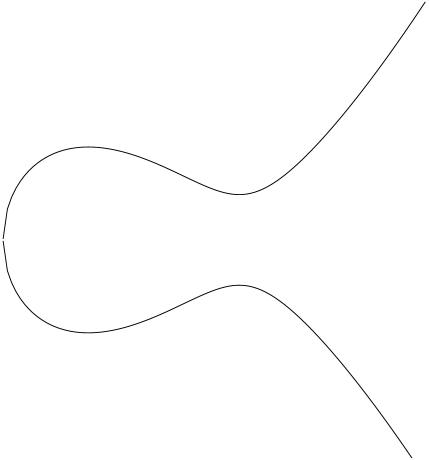


Figure: Elliptic curve rule

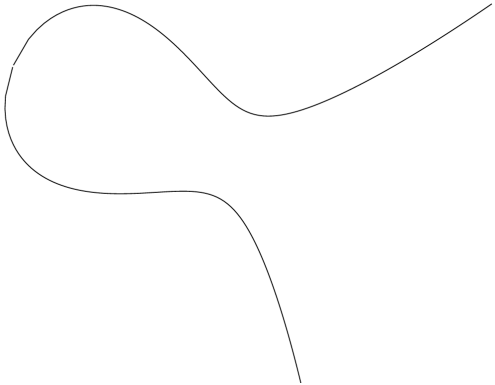
Elliptic curves

An elliptic curve satisfies $y^2 = x^3 + ax + b$ over a field.



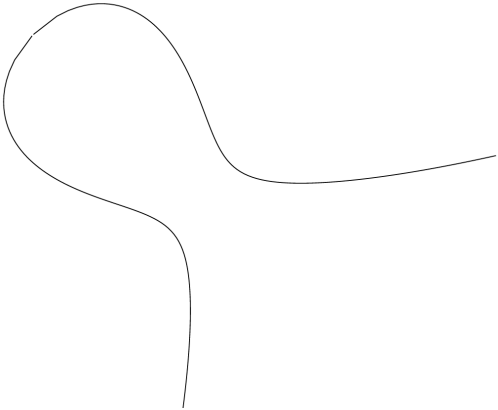
Elliptic curves

An elliptic curve satisfies $y^2 = x^3 + ax + b$ over a field.



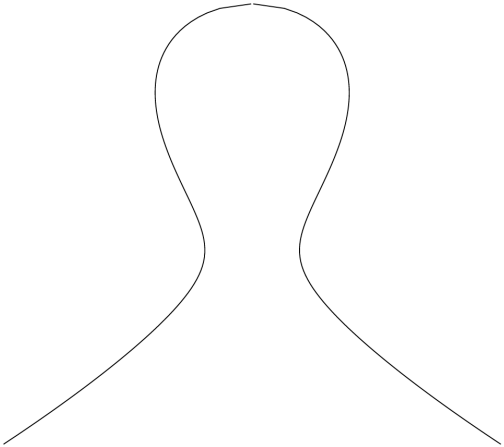
Elliptic curves

An elliptic curve satisfies $y^2 = x^3 + ax + b$ over a field.



Elliptic curves

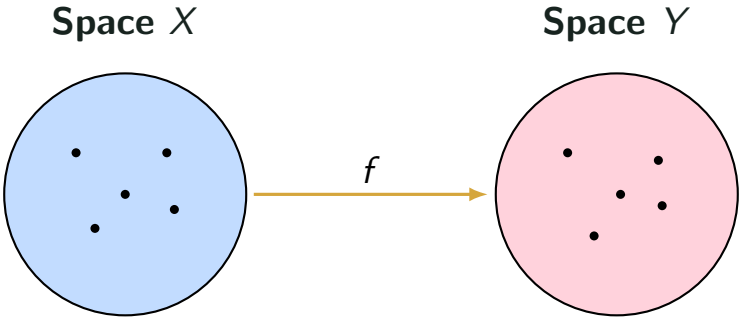
An elliptic curve satisfies $y^2 = x^3 + ax + b$ over a field.



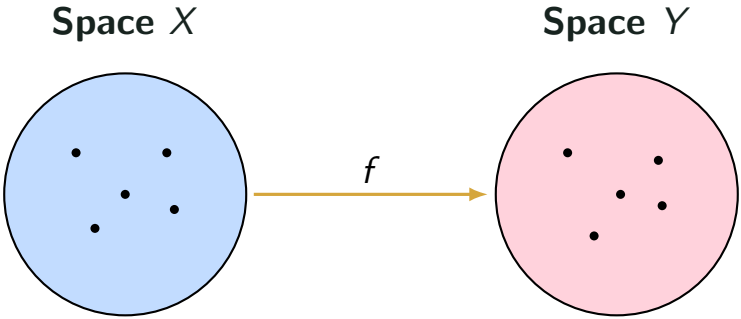
Elliptic curves

Credit: Rachel Deyts

A familiar idea



A familiar idea



$$f(x, y) = (g(x), h(y))$$

An isogeny is a function between elliptic curves

$$E \xrightarrow{\phi} E'$$

An isogeny is a function between elliptic curves

$$E \xrightarrow{\phi} E'$$

So an isogeny is still a function:

$$\phi : E \longrightarrow E'$$

An isogeny is a function between elliptic curves

$$E \xrightarrow{\phi} E'$$

So an isogeny is still a function:

$$\phi : E \longrightarrow E'$$

$$\phi(x, y) = (f(x), g(x, y))$$

How do we describe an isogeny?

- An isogeny is identified by its kernel:

$$\ker \phi \subset E.$$

Alice **Bob**

$$E \xrightarrow{\phi} E'$$

$$\ker \phi = \langle g \rangle$$

A cyclic kernel generator determines the isogeny.

How do we describe an isogeny?

- An isogeny is identified by its kernel:
$$\ker \phi \subset E.$$
- In cryptography, we usually work with **cyclic** kernels.

Alice **Bob**

$$E \xrightarrow{\phi} E'$$

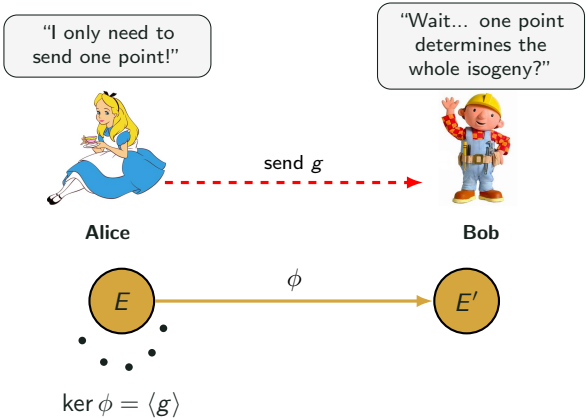
$$\ker \phi = \langle g \rangle$$

A cyclic kernel generator determines the isogeny.

How do we describe an isogeny?

- An isogeny is identified by its kernel:
$$\ker \phi \subset E.$$
- In cryptography, we usually work with **cyclic** kernels.
- So if
$$\ker \phi = \langle g \rangle,$$
then giving the generator g is enough to determine the isogeny.

just send $g = (x, y) \in E$
so 32-64B only!

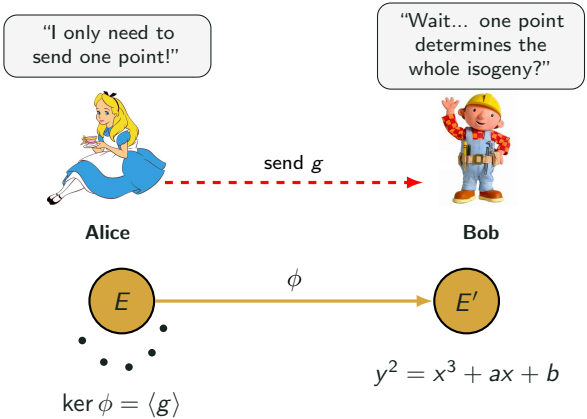


A cyclic kernel generator determines the isogeny.

How do we describe an isogeny?

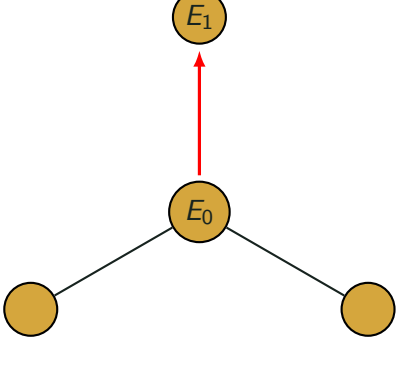
- An isogeny is identified by its kernel:
$$\ker \phi \subset E.$$
- In cryptography, we usually work with **cyclic** kernels.
- So if
$$\ker \phi = \langle g \rangle,$$
then giving the generator g is enough to determine the isogeny.

just send $g = (x, y) \in E$
so 32-64B only!

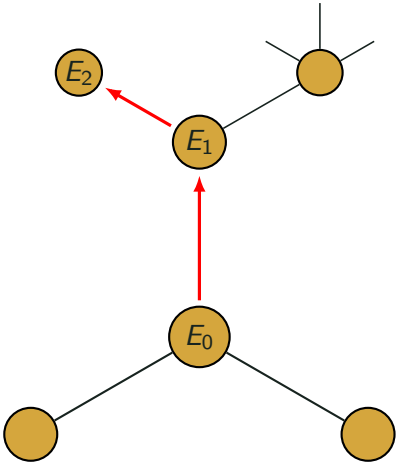


A cyclic kernel generator determines the isogeny.

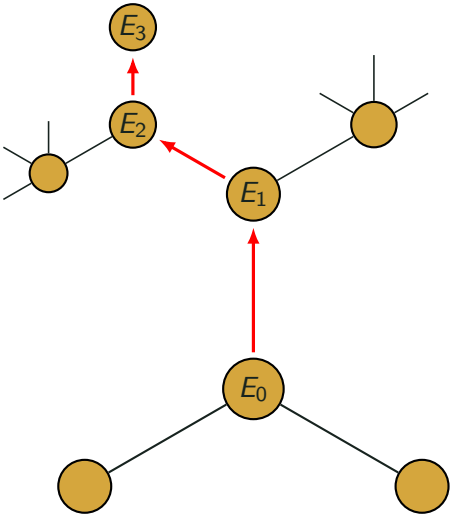
Degree-2 Isogenies



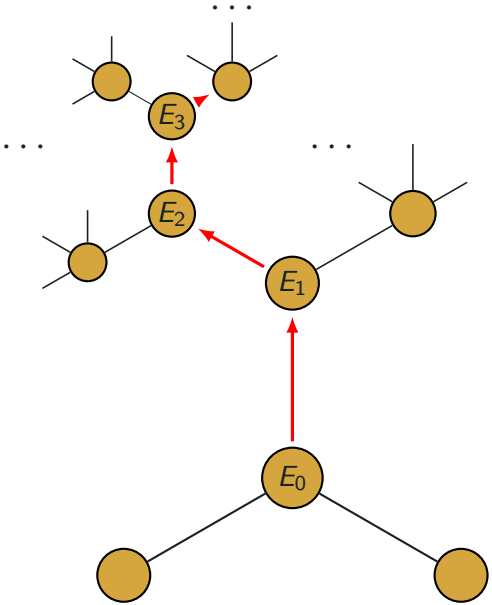
Degree-2 Isogenies



Degree-2 Isogenies



Degree-2 Isogenies



Small example: Isogeny graphs

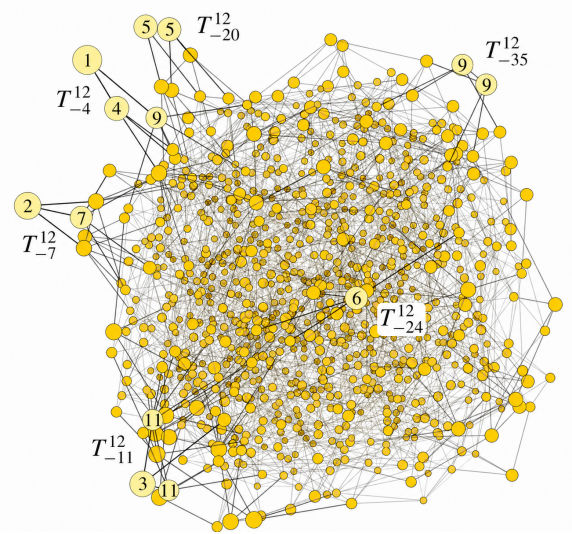


Figure: The supersingular 2-isogeny graph over $\mathbb{F}_{20011}$ [LB20].

Supersingular ℓ-isogeny graphs

- The graph has

$$\mathcal{O}(p) = \mathcal{O}(2^n)$$

  vertices, so for cryptographic p, it is exponentially large.
- It is $(\ell + 1)$-regular.
- It is a **Ramanujan graph**: an essentially optimal expander.
- Random walks rapidly approach a near-uniform distribution.

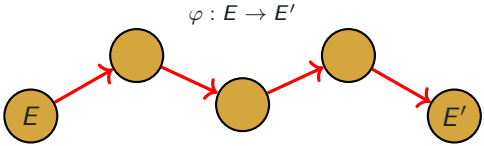
Supersingular isogeny path problem

Problem

Given two supersingular elliptic curves E and E', viewed as vertices in the supersingular isogeny graph, find an isogeny

$$\varphi : E \longrightarrow E',$$

equivalently, find a walk in the graph connecting E to E'.



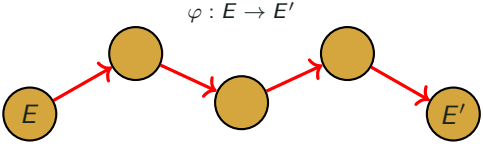
Supersingular isogeny path problem

Problem

Given two supersingular elliptic curves E and E', viewed as vertices in the supersingular isogeny graph, find an isogeny

$$\varphi : E \longrightarrow E',$$

equivalently, find a walk in the graph connecting E to E'.



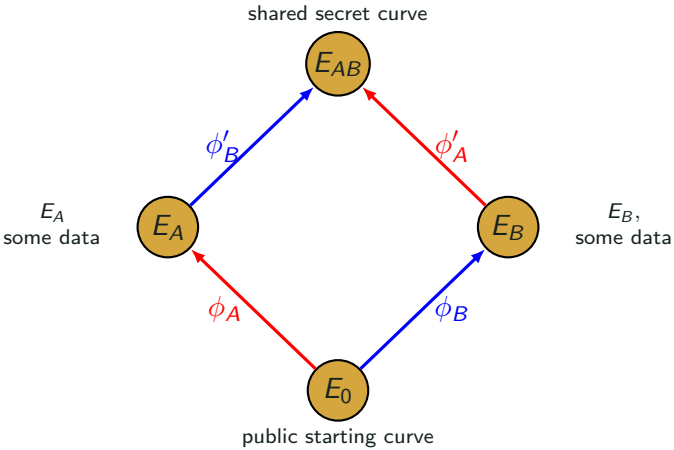
Best known generic attack

The best generic classical approach is essentially meet-in-the-middle search:

$$T_{\text{classical}} \approx O(p^{1/2}).$$

This square-root barrier has been the baseline for roughly 30 years.

A cautionary lesson: SIDH/SIKE



SQIsign: compact isogeny-based signatures

- Isogeny-based FS-digital signature scheme.
- Based on hidden-path hardness in supersingular isogeny graphs.
- Advanced to Round 3 of NIST's additional signature call.
- Smallest sig + pk sizes of all post-quantum schemes.

Set	**pk**	**sig**
ECDSA P-256	$\sim$ 64 B	$\sim$ 64 B
SQIsign-I	65 B	148 B
SQIsign-III	97 B	224 B
SQIsign-V	129 B	292 B

SQIsign performance

Set	NIST level	Signing time	Verification time
SQIsign-I	1	49.39 ms	3.44 ms
SQIsign-III	3	130.16 ms	9.16 ms
SQIsign-V	5	215.70 ms	17.82 ms

Benchmarked on a 12th Gen Intel Core i7-12650H, 8 cores.

Takeaway

SQIsign trades efficiency for extreme compactness.
It is especially attractive in bandwidth-constrained settings.

Design space

- Lattice have linearity making them desirable for **FHE**.
- Codes provide one of the most **durable** and **secure** alternatives
- Isogenies are cool in **randomness**-related schemes like VRF's and Key-exchange.

Final message

Post-quantum cryptography is not just about replacing RSA and ECC.

It is a rich design space.

Different mathematical worlds support different cryptographic primitives:

Lattices Codes Isogenies

and each one offers a different way to meet applied cryptographic needs.