Quantum Algorithms for Attacking Post-Quantum Cryptography?

Mark Zhandry (Stanford University, Google Quantum AI)

Shor’s quantum algorithm breaks essentially all currently deployed public key cryptography

Breaking cryptography is the most significant consequence of quantum computing that we can predict with any reasonable confidence

Over the past decade, the prospect of quantum computers has arguably been the single greatest force shaping cryptographic research

And likely will remain as such for the decade to come

And yet, I’d argue we are still far behind where we need to be…

NIST and other organizations are (probably correctly) already encouraging the migration to post-quantum cryptosystems

But, our confidence in their quantum security is severely behind relative to the classical setting

Central Challenge: We ultimately gain confidence in the security of our cryptosystems by trying – and hopefully failing! – to break them

Problem 1: Order of magnitude (or worse) fewer people involved

#(people who have seriously tried to find a classical algorithm for factoring)?

According to ChatGPT: ≈ 2k- 10k

#(people who have seriously tried to find a quantum algorithm for lattices)?

According to ChatGPT: ≈ 100-150

Problem 2: Order of magnitude (or worse) fewer techniques

100s of truly distinct classical algorithmic techniques

Even just to factor integers classically:

- Trial division
- Fermat's method
- Pollard's rho
- Pollard's p−1
- Continued fraction methods
- Quadratic sieve
- Elliptic curve method
- Number field sieve
- Coppersmith's lattice-based techniques

But, maybe only a dozen or two truly distinct quantum algorithms

- QFT/hidden subgroup
- Amplitude amplification
- Quantum walks
- Quantum Singular Value Transform
- Quantum simulation
- Quantum linear algebra
- Random circuit sampling and friends
- …

Problem 3: Many orders of magnitude more important today

According to ChatGPT:

Encrypted communication in 1990 ≈ 10^{10}-10^{12} bytes/year

Encrypted communication in 2025 ≈ 10^{21} bytes/year

If there are any quantum attacks, we need to find them NOW, before widespread deployment

Compared to the development of (pre-quantum) public key cryptography, we are facing:

- Drastically fewer researchers
- Drastically fewer techniques
- A vastly more catastrophic scenario if we fail

My goal for today: get more people on board!

The Plan

10:30am-12:30pm: post-quantum cryptosystems and quantum algorithms for them

12:30pm – 2pm: lunch

2pm – 3pm: Kewen Wu on dequantizing certain quantum algorithms

3pm-5pm: Break + Discussions

This morning's outline:

1. Recap: Shor + Kuperberg
2. Cryptography from group actions
3. Cryptography from lattices
4. Cryptography from codes
5. Very brief summary of others

Please stop me for questions!

1. Recap: Shor + Kuperberg

Let $\mathbb{G}$ be a group

Def: A function $f : \mathbb{G} \to \mathcal{Y}$ is periodic if there exists a proper subgroup $\mathbb{H} \subseteq \mathbb{G}$ such that:

- $f(g + h) = f(g) \quad \forall g \in \mathbb{G}, h \in \mathbb{H}$
- $f(g) \neq f(g')$ if $g - g' \notin \mathbb{H}$

In this case, we call $\mathbb{H}$ the period of f

Thm [Simon'94, Shor'94,…]: There exists a QPT algorithm for computing the period of any periodic function in the case where $\mathbb{G}$ is *abelian*

Technically, the period will often be an exponential-sized set, which we cannot enumerate in polynomial time. So we actually only ask for a generating set

Factoring Integers

Given $N = pq$, let $a \leftarrow \mathbb{Z}_N^*$ and define

$$f_a : \mathbb{Z} \to \mathbb{Z}_N^*$$
$$f_a(x) = a^x \bmod N$$

Thm [Euler]: $a^{(p-1)(q-1)} \bmod N = 1$

Thus, period of f_a is* $(p-1)(q-1)$

$\Longrightarrow$ recover p, q with a little algebra

* Technically, it divides this number, and the actual factoring algorithm is a bit different. But good enough for our purposes

Solving Discrete Logarithms

Let $\mathbb{G}'$ be a group, g an element of prime order p

Discrete log problem: $h = g^a \mapsto a$ for $a \in \mathbb{Z}_p$

DLog as period-finding: define $f : \mathbb{Z}_p^2 \to \mathbb{G}'$

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

$f(x + a, y - 1) = f(x, y)$ ➡ Period of f is $(a, -1)$

Remark: seems like a crazy coincidence that all currently-deployed public key crypto can be cast as abelian period-finding

As we will see, while post-quantum candidates are not period-finding, there are still somewhat surprising coincidences among them

The Non-Abelian Case

For non-abelian groups, Shor's algorithm fails

Rough Reason: while QFT exists for non-abelian groups and is even efficient for many important cases, it behaves very differently

- Characters vs group elements
- Multi-dimensional vs phases

Important Special Case: Hidden shift / Dihedral

Let $\mathbb{G}$ be an abelian group

Given injective $f_0, f_1 : \mathbb{G} \to \mathcal{Y}$ with the promise

$$\exists a \in \mathbb{G} : f_1(g) = f_0(ag)$$

Goal: find a

Can view as hidden subgroup for $f(b, g) = f_b(g)$ where domain is interpreted as an appropriate dihedral group

Note that DLog is a special case of hidden shift

$$f_0(x) = g^x$$
$$f_1(x) = h^x \Rightarrow f_1(x) = f_0(ax)$$

BUT, Shor uses extra structure to compute $x, y \mapsto g^x h^y$

With general hidden shift, no way to combine outputs of f_0, f_1

Quantum Algorithms for Hidden Shift

Simon's: poly-time when $\mathbb{G} = \mathbb{Z}_2^n$

Observation: hidden shift when $\mathbb{G} = \mathbb{Z}_2^n$ is just hidden subgroup over $\mathbb{Z}_2^{n+1}$, $f(b, g) = f_b(g)$

$$f(1 \oplus 1, g \oplus a) = f(0, g \oplus a) = f(1, g)$$
$$f(0 \oplus 1, g \oplus a) = f(1, g \oplus a) = f(0, g)$$

Kuperberg's Algorithm

Time $2^{O(\sqrt{\log |\mathbb{G}|})}$ for hidden shift

For simplicity, switch to an additive notion for hidden shifts

Let $\mathbb{G}$ be an abelian group, written additively

Given injective $f_0, f_1 : \mathbb{G} \to \mathcal{Y}$ with the promise

$$\exists a \in \mathbb{G} : f_1(g) = f_0(a + g)$$

I'll also assume $\mathbb{G} = \mathbb{Z}_{2^n}$ for simplicity

Kuperberg's Algorithm

1. $$\frac{1}{\sqrt{2|\mathbb{G}|}} \sum_{b,g} |g, b, f_b(g)\rangle = \frac{1}{\sqrt{2|\mathbb{G}|}} \sum_{b,g} |g, b, f_0(g + ba)\rangle$$
$$= \frac{1}{\sqrt{2|\mathbb{G}|}} \sum_{b,g} |g - ba, b, f_0(g)\rangle$$

2. Measure and discard final register

$$\frac{1}{\sqrt{2}} \left(|g, 0\rangle + |g - a, 1\rangle\right)$$

For random (unknown) g

Kuperberg's Algorithm

$$\frac{1}{\sqrt{2}}\left(|g,0\rangle + |g-a,1\rangle\right)$$

3. QFT the first register

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{h,b}|h,b\rangle e^{i2\pi(g-ba)h/2^n} = \frac{1}{\sqrt{2^n}}\sum_{h}|h\rangle e^{i2\pi gh/2^n}\frac{1}{\sqrt{2}}\sum_{b}|b\rangle e^{-i2\pi bah/2^n}$$

4. Measure first register

$$h \quad , \quad \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle e^{-i2\pi ah/2^n}\right)$$

Kuperberg's Algorithm

$$h \ , \ \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle e^{-i2\pi ah/2^n}\right)$$

Now, suppose we got lucky and $h = 2^{n-1}$

$$|0\rangle + (-1)^a|1\rangle$$

Measure in Hadamard basis to learn LSB of a

→ Can recursively solve for $\lfloor a/2 \rfloor$

Kuperberg's Algorithm

$$h \ , \ \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle e^{-i2\pi ah/2^n}\right)$$

Now, suppose we got lucky and $h = 2^{n-1}$

$$|0\rangle + (-1)^a|1\rangle$$

But of course, such an h is exponentially unlikely

Kuperberg's Algorithm

$$h_0 \;,\; \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle e^{-i2\pi a h_0/2^n}\right) \qquad h_1 \;,\; \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle e^{-i2\pi a h_1/2^n}\right)$$

CNOT

$$\frac{1}{2}\sum_{b_0,b_1} |b_0, b_1 \oplus b_0\rangle e^{-i2\pi a(b_0 h_0 + b_1 h_1)/2^n}$$

$$= \frac{1}{2}\sum_{b_0} |b_0, 0\rangle e^{-i2\pi a b_0(h_0+h_1)/2^n} + |b_0, 1\rangle e^{-i2\pi a b_0(h_0-h_1)/2^n - i2\pi a h_1/2^n}$$

Kuperberg's Algorithm

$$= \frac{1}{2} \sum_{b_0} |b_0, 0\rangle e^{-i2\pi a b_0 (h_0+h_1)/2^n} + |b_0, 1\rangle e^{-i2\pi a b_0 (h_0-h_1)/2^n - i2\pi a h_1/2^n}$$

Measure 2nd register, with probability ½ will obtain:

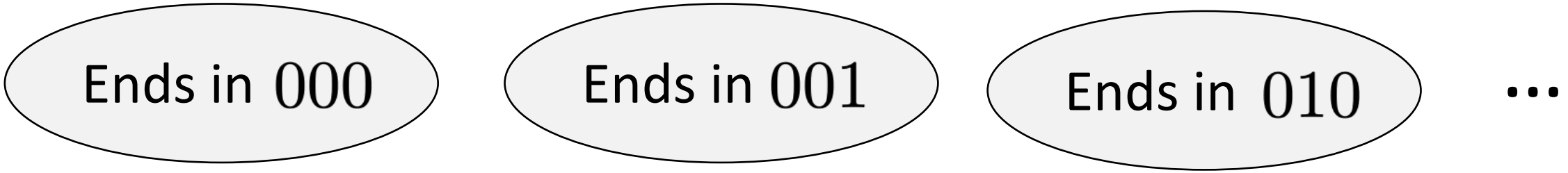
$$h_0 - h_1 \bmod 2^n \quad , \quad \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle e^{-i2\pi a (h_0-h_1)/2^n} \right)$$

Thus, we can subtract samples to obtain new ones

Kuperberg's Algorithm

Start with $2^{O(\sqrt{n})} = 2^{O(\sqrt{\log |\mathbb{G}|})}$ samples

Divide into buckets based on $\sqrt{n}$ least significant bits



Pair off within each bucket, subtract

➡ Loose ~ ¼ of samples, but now samples end in $0^{\sqrt{n}}$

Kuperberg's Algorithm

Repeat $\sqrt{n}$ times:

Every iteration looses ¼ of samples

But remaining samples end in $\sqrt{n}$ additional 0's

By the end, $2^{O(\sqrt{n})} \times 2^{-O(\sqrt{n})} = 1$ sample remains

Sample will end in 0^{n-1} ✓

Simon-meets-Kuperberg

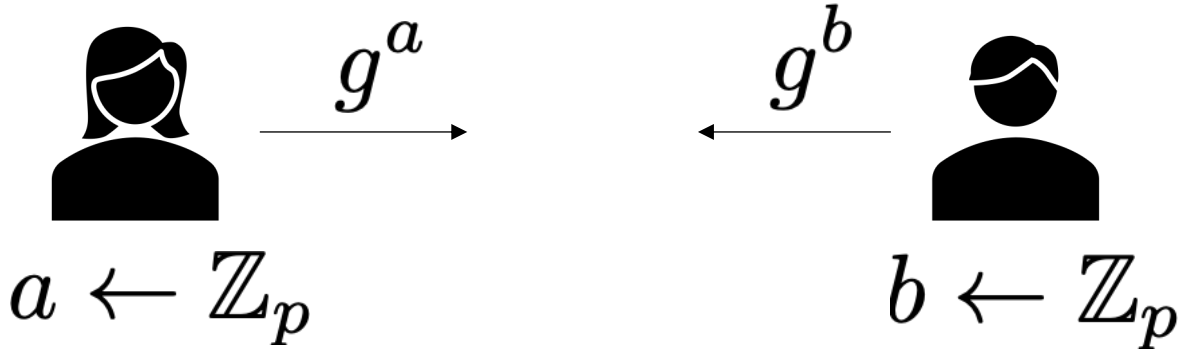
[Bonnetain-Naya-Plasencia'18]

If $\mathbb{G} = \mathbb{Z}_{2^a}^b$, can solve in time $b^{O(a)}$

Idea: in each iteration, lower-order bit of *all* coordinates set to zero, but at the cost of requiring b samples to create 1 sample

2. Cryptography from group actions

Recall Classical Diffie-Hellman:



$$k = g^{ab} = (g^a)^b = (g^b)^a$$

Discrete exponentiation efficient by repeated squaring

Observation

Diffie-Hellman only requires discrete exponentiation

- While obtained from multiplication, multiplication not explicitly needed

However, Shor's algorithm needs multiplication

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

Group Actions

[Brassard-Yung'90]

Group $\mathbb{G}$ acting on a set $\mathcal{X}$

$$a * (b * x) = (ab) * x$$

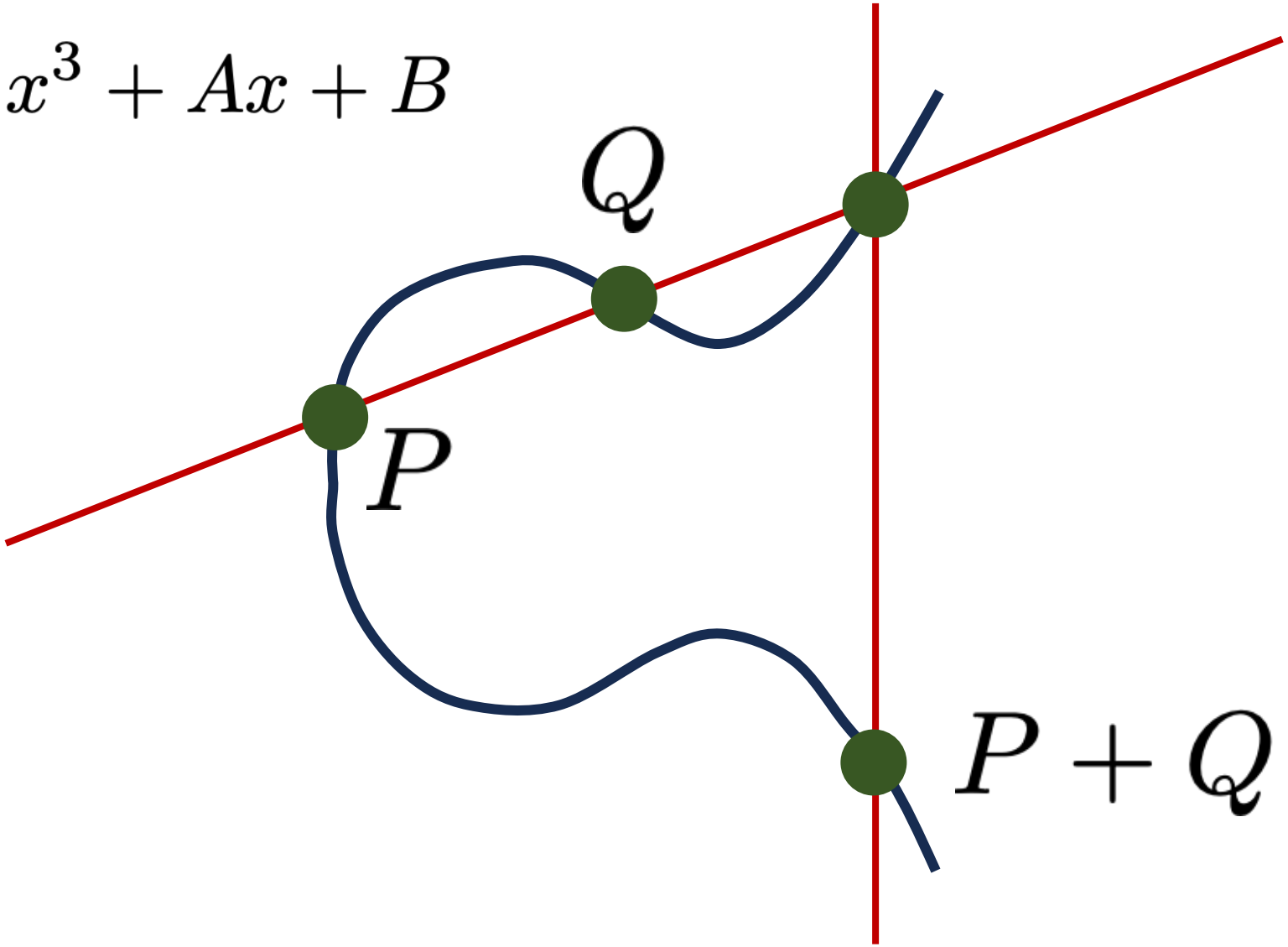
(Abelian) group action Diffie-Hellman:

$$k = (ab) * x = b * (a * x) = a * (b * x)$$

Where do (abelian) group actions come from?

Elliptic Curves for Pre-Quantum Crypto

$$y^2 = x^3 + Ax + B$$



Elliptic Curves for Pre-Quantum Crypto

Point addition is a rational function

Already have two solutions to cubic, can solve for third without taking roots

$$x^3 + ax^2 + bx + c = (x - r_1)(x - r_2)(x - r_3)$$

$$\Rightarrow \quad r_3 = -a - r_1 - r_2$$

Addition still works over finite fields, although geometric intuition lost

Elliptic Curves for Pre-Quantum Crypto

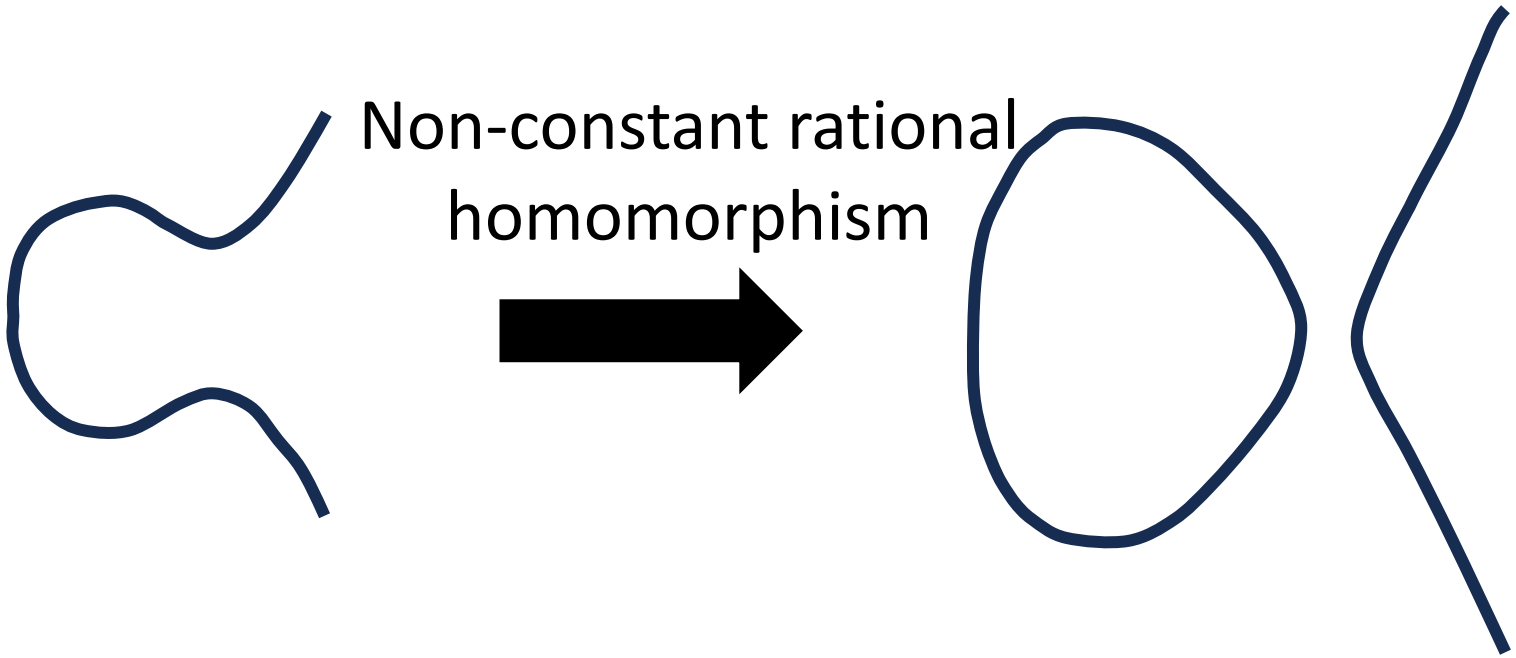
Once we have a group, we can talk
about discrete exponentiation/DLog

Exponentiation: high-degree rational function

Discrete log: taking roots of high-degree functions. Seems hard

To this day, if careful to avoid certain “bad” curves, only known practical classical attacks only make generic use of group multiplication. Such attacks provable take exponential time

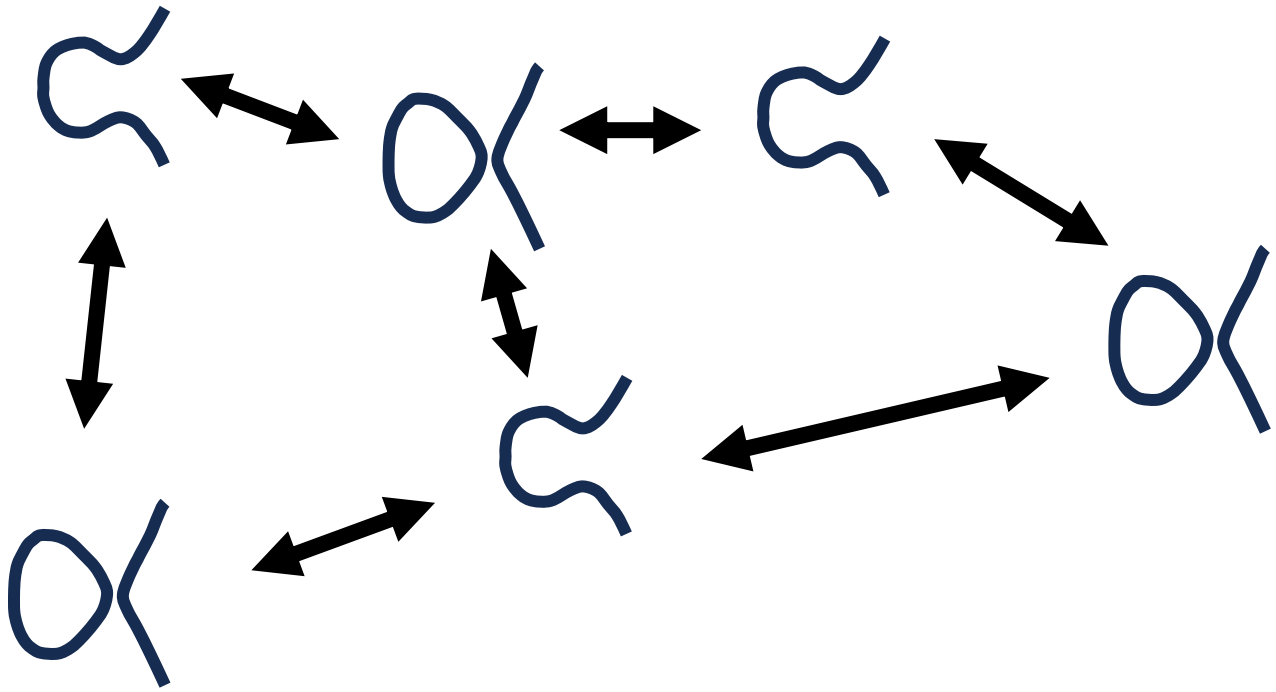
Isogenies for Post-Quantum Crypto



For cryptography, typically
use high-degree isogenies

Isogenies for Post-Quantum Crypto

Isogeny graph



For certain families of elliptic curves (including "ordinary" curves over finite fields) can essentially endow isogeny graph with an abelian group structure

$\mathcal{X} =$ Set of elliptic curves (modulo some equivalence classes)

$\mathbb{G} =$ "Ideal class group", which correspond to isogenies

Why do people think group actions from isogenies might be secure?

Intuition: even though Shor solves discrete logarithms on a quantum computer, it is still “generic” in the sense that it only uses the prescribed group operations in a black box way

Since exponentiation is a high-degree function, it seems hard to do anything non-generic

Since isogenies derive from similar areas of mathematics and are also high-degree functions, ***maybe*** the best quantum attacks remain generic

Currently, the only known generic attacks based on group actions simply solve hidden shift (Kuperberg)

$$y = a * x \quad \Rightarrow \quad \begin{aligned} f_0(a) &= a * x \\ f_1(a) &= a * y = (ag) * x \end{aligned}$$

Why might isogenies nevertheless be insecure?

Problem 1: group actions have much more structure than hidden shifts

$$f_0(a) = a * x$$
$$f_1(a) = a * y = (ag) * x$$

In general hidden shifts, once you have function output there's nothing else you can do

For group action hidden shifts, you can keep acting and acting and acting and..

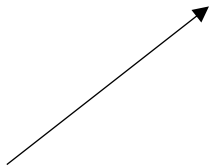
[Mutreja-Z'25]: some preliminary justification for generic group action hardness

Problem 2: maybe non-generic attacks?

In classical elliptic curve setting, some curves admit a “pairing”

MOV attack: DLog in EC ➡ DLog in Finite Field

[Menezes-Okamoto-Vanstone’93]



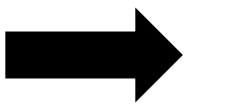
Sub-exponential classical attack. To not cause problems, need to ensure target field is very large

Problem 2: maybe non-generic attacks?

Could there by non-generic attacks on elliptic curves?

Isogeny “pairing”?

$$a * x$$
$$b * x$$



Different group action

$$(ab) \star \tilde{x}$$

Such a pairing would allow for carrying our Shor’s algorithm

Notion investigated in [Boneh-Glass-Krashen-Lauter-Sharif-Silverberg-Tibouchi-Z’18], but no efficient candidates found

An aside about SIDH

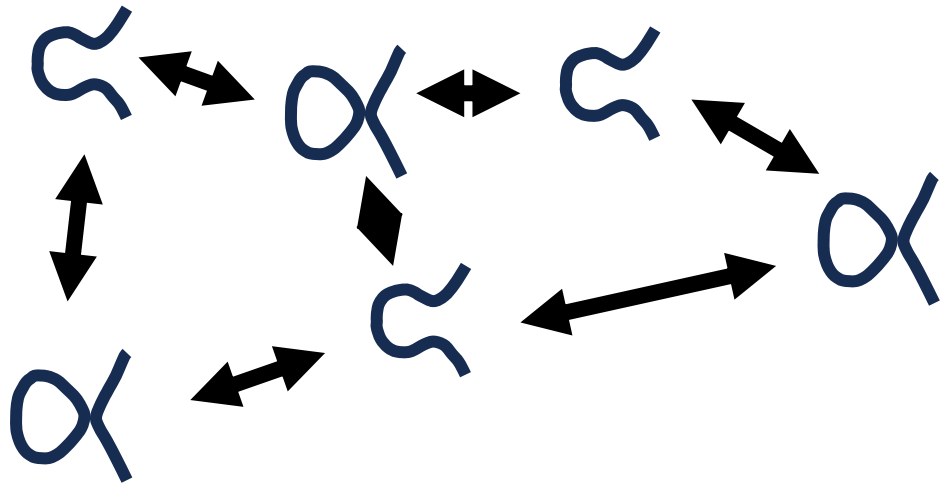
Due to Kuperberg's algorithm, need to set parameters much bigger to remain secure

128-bit security → $|\mathbb{G}| \approx 2^{128^2} \approx 2^{8000}$

Compare to classical groups, which only need size $\approx 2^{256}$

An aside about SIDH

For *super-singular curves*, no way to derive an abelian group action. Instead, often think of action as a walk on isogeny graph, an exponentially-large implicit expander



Nothing known better than birthday-attack → exponential security

An aside about SIDH

But, no easy way to do Diffie-Hellman without abelian group structure

SIDH: a mechanism to allow for key agreement by providing extra information/hints

> Unfortunately, [Castryck-Decru'22] showed that this extra info leads to classical efficient attacks

An aside about SIDH

Takeaway 1: attack only relevant to extra info given out. The core isogeny structure seems to remain secure even quantumly

Takeaway 2: beware of doomsday scenario where our post-quantum replacements are actually insecure today

The case of signatures

Obtaining efficient signatures from isogenies/group actions is hard!

In the case of generic group actions, partial explanation given in [Boneh-Guan-Z'23]

SQISign: uses super-singular isogenies, but departs from group action/graph abstraction and exploits deep mathematical facts to obtain an efficient signature

My opinion: we don’t have a good understanding of why super-singular isogenies should be exponentially-quantumly secure

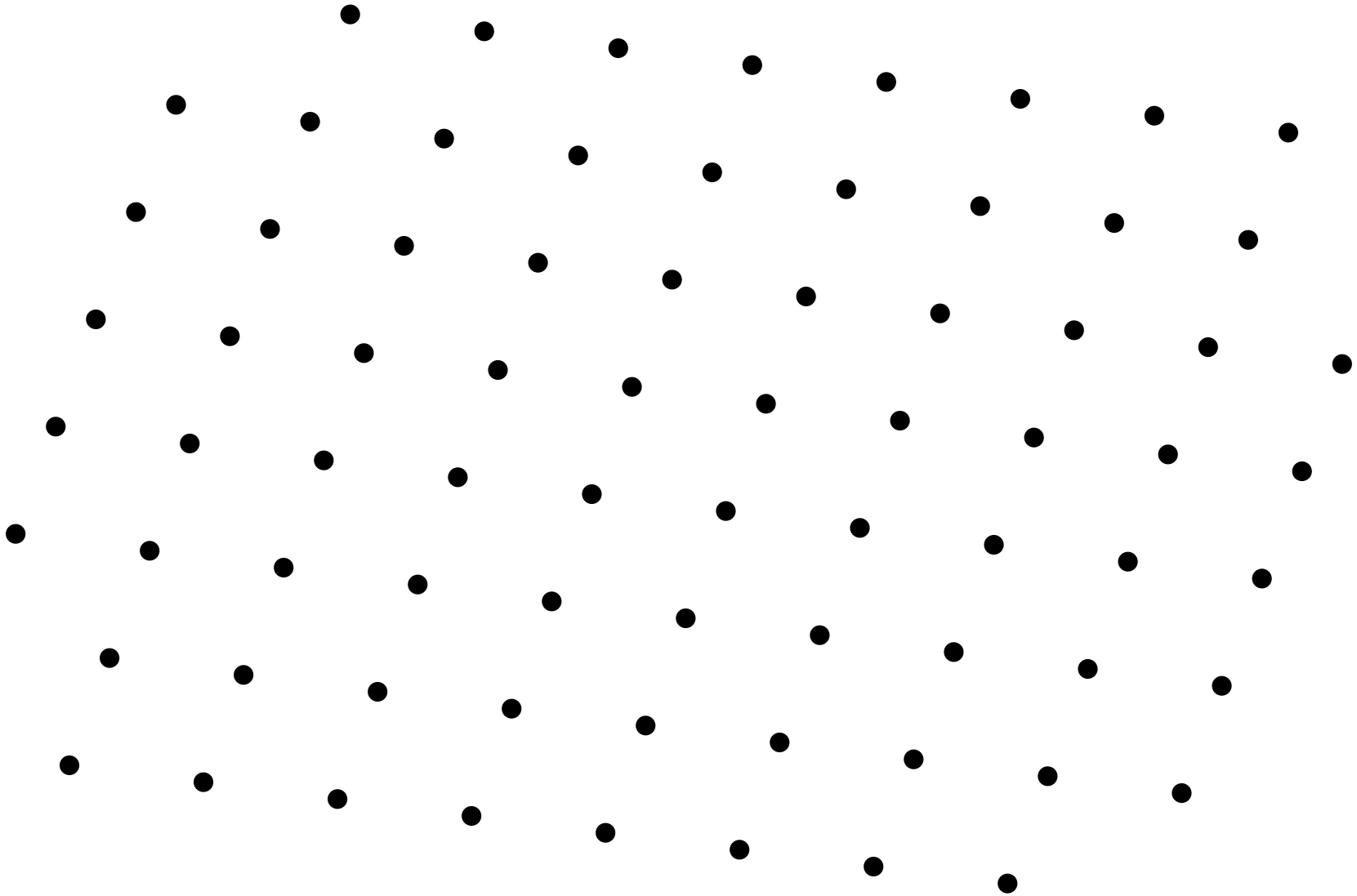
After all, quantum walks can accomplish non-trivial things!

[Childs-Cleve-Deotto-Farhi-Gutmann-Spielman’02]

Even a sub-exponential attack would be problematic and render current proposed parameter setting insecure

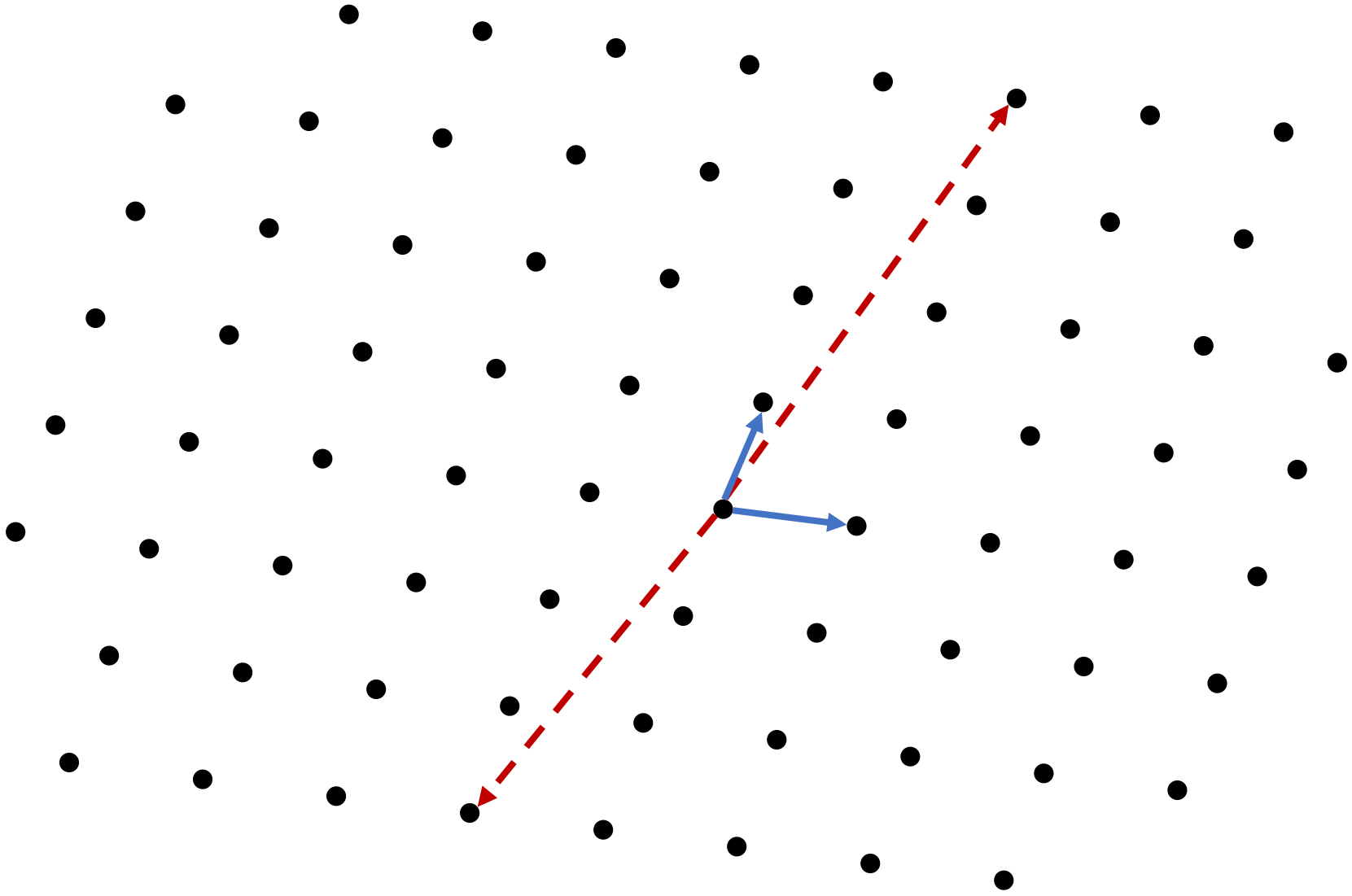
3. Cryptography from lattices

Lattices



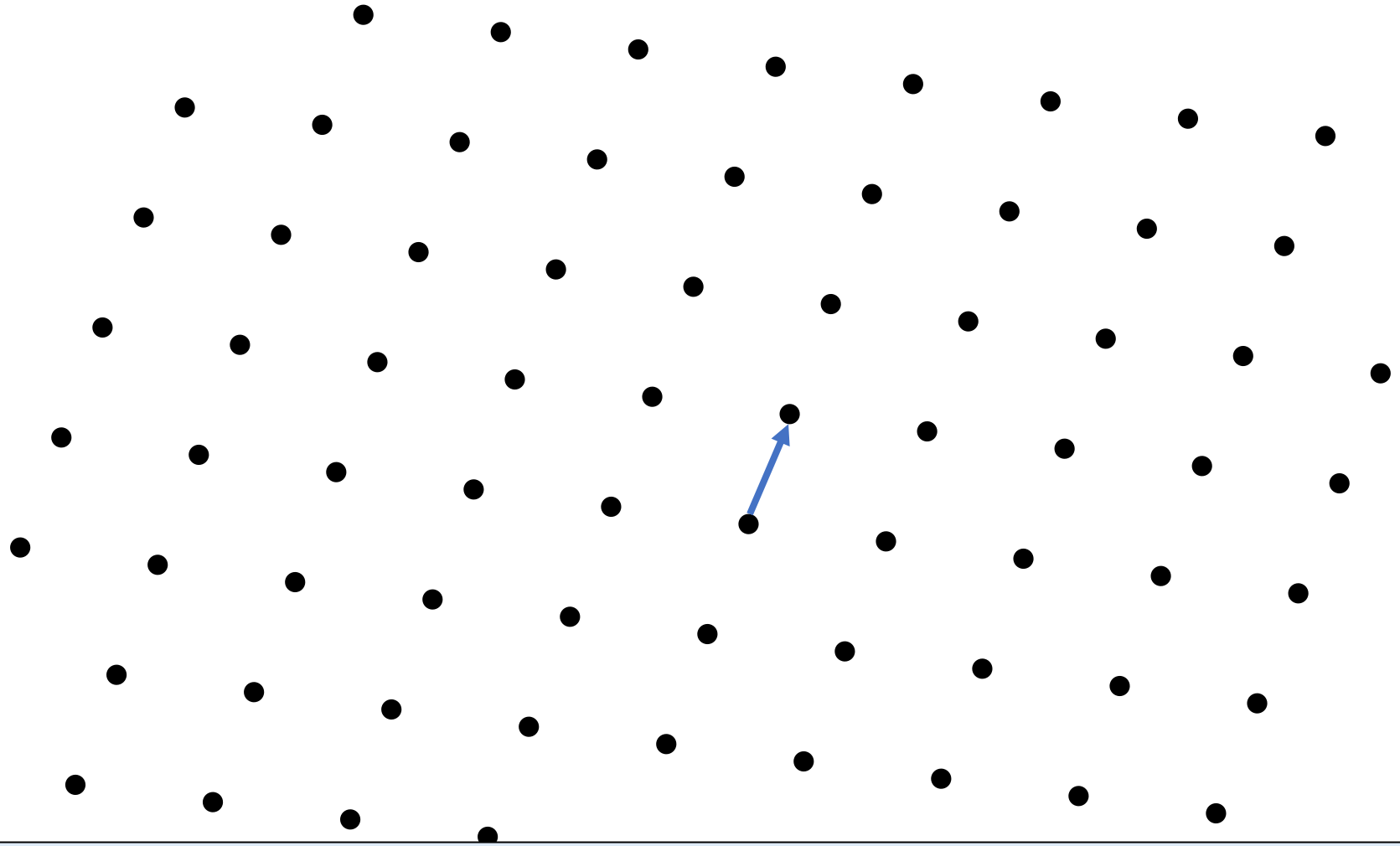
Imagine dimension in the 100s

Lattices



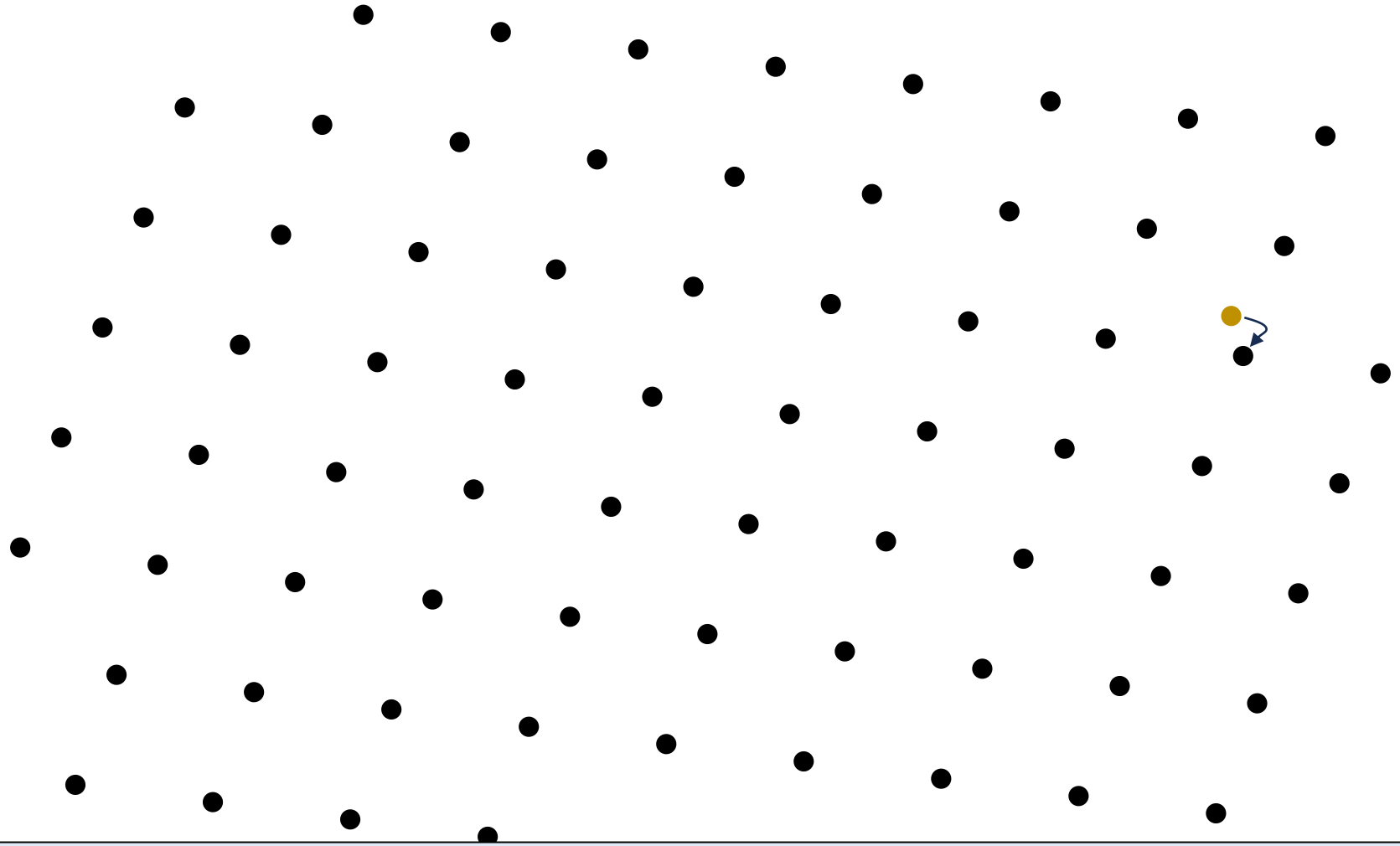
Basis: minimal set of vectors that generate lattice

Lattices



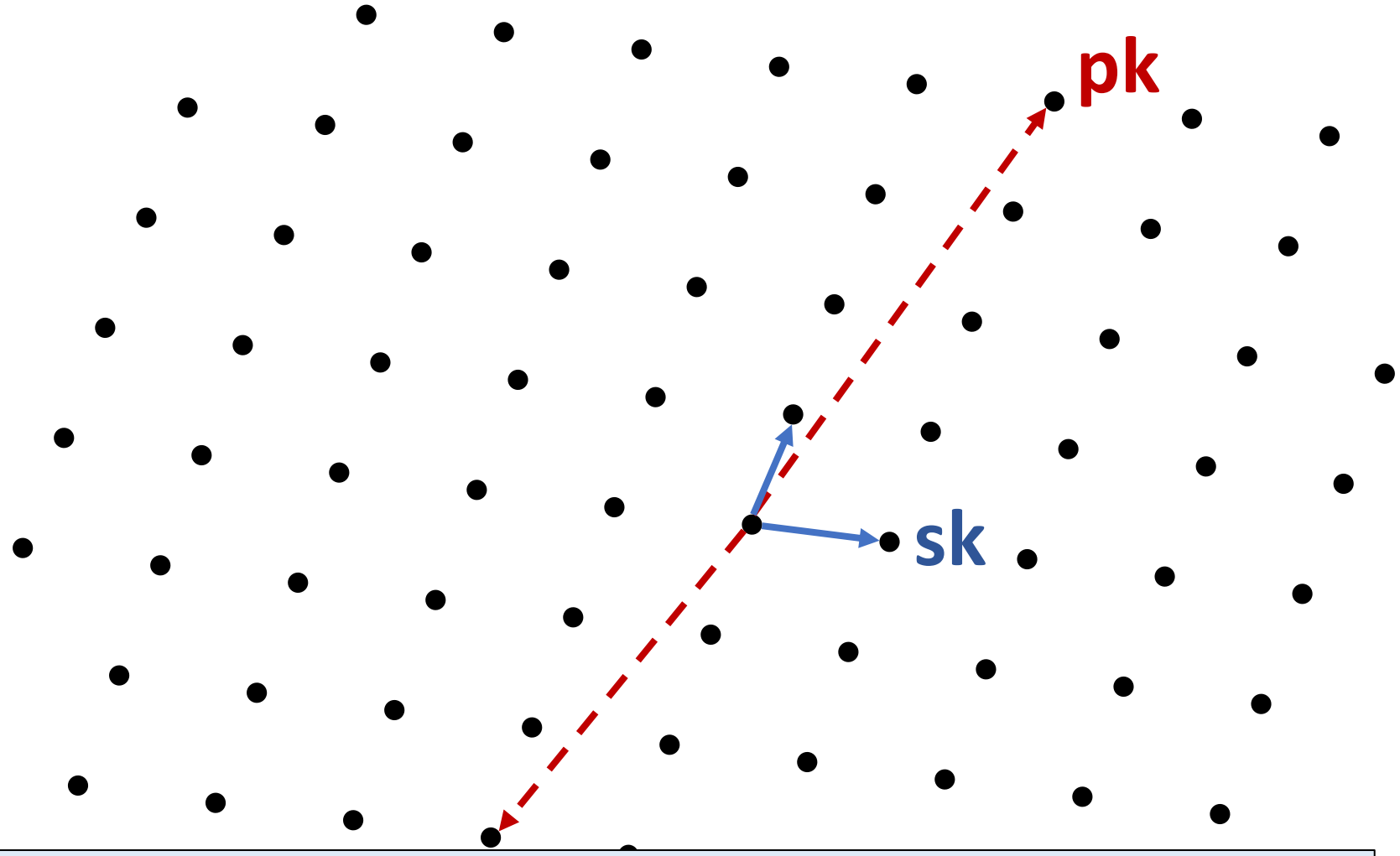
(Approx.) shortest vector problem (SVP): given lattice (described by some basis), find (approx.) shortest vector

Lattices



(Approx.) closest vector problem (CVP): given lattice and point off lattice, find (approx.) closest lattice point

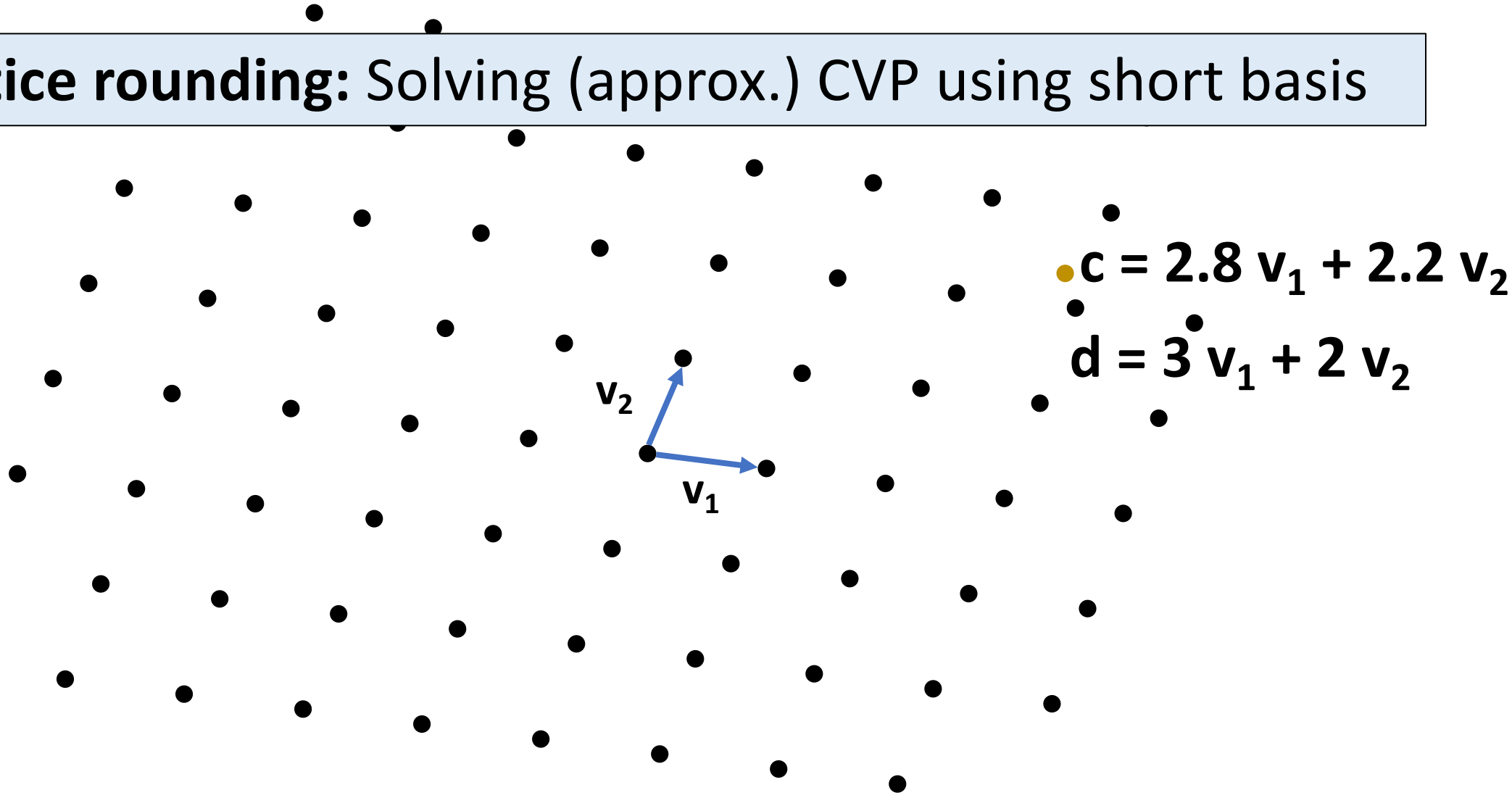
Lattices



Trapdoor: Give out large basis as public key, keep short basis as secret key / trapdoor

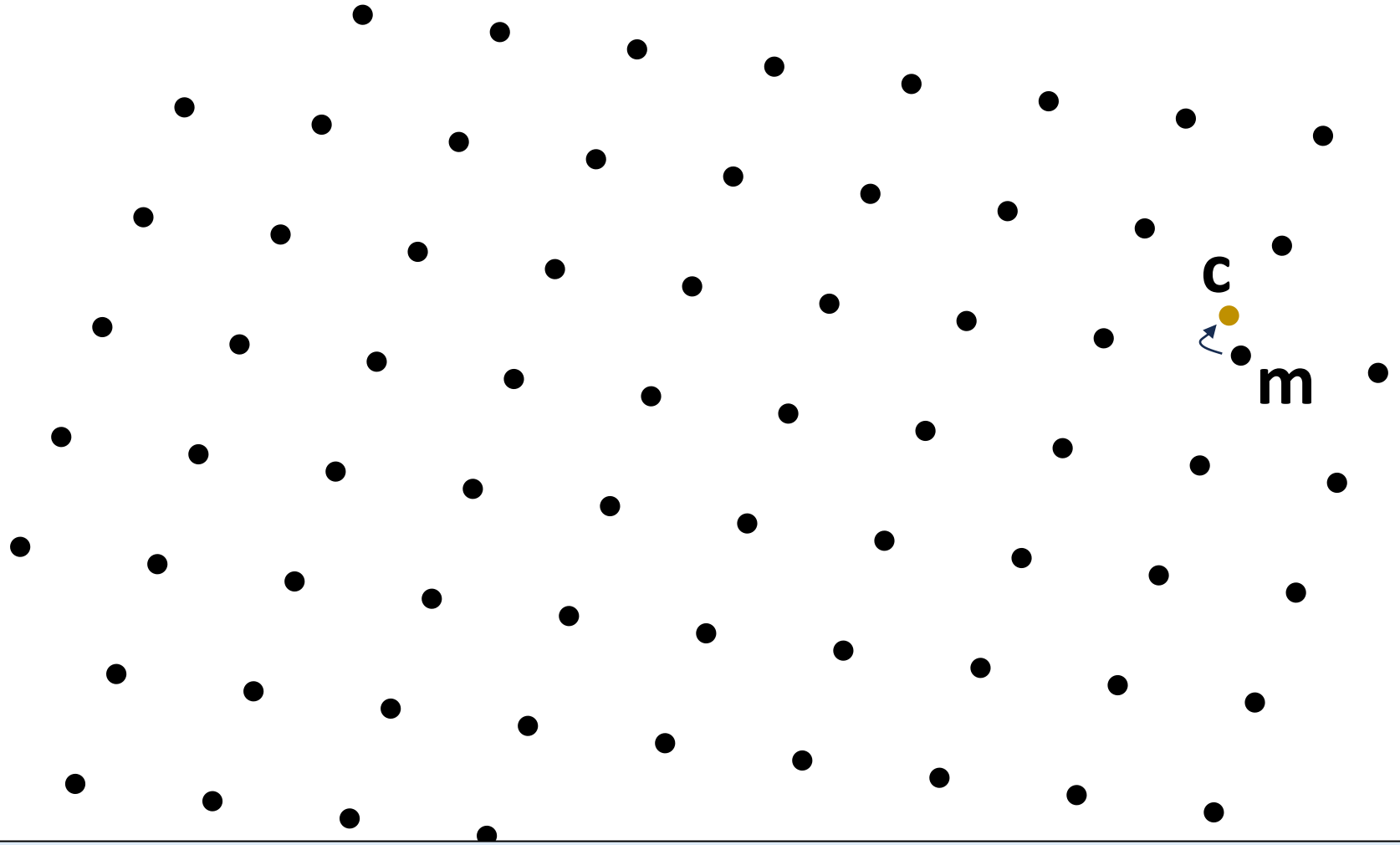
Lattices

Lattice rounding: Solving (approx.) CVP using short basis



Shorter bases give closer rounding

Lattices



Encrypt **m**: (1) Map **m** to lattice point
(2) Output close non-lattice point

SIS and LWE

SIS:

Given random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ $(m \gg n)$

Find short $\mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} \bmod q = 0$

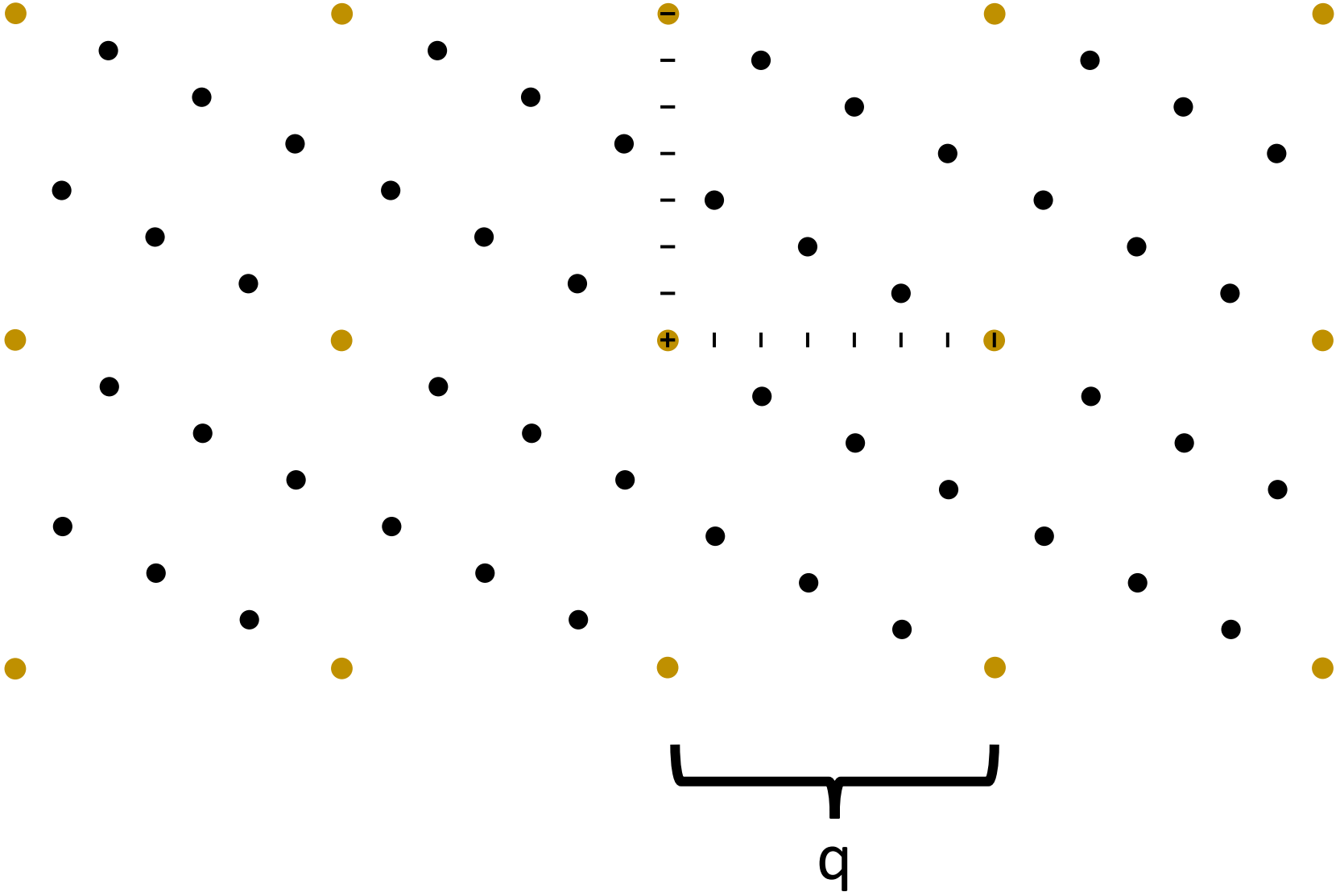
LWE:

Given random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ $(m \gg n)$

$$\mathbf{u} = \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \bmod q$$

Find $\mathbf{s}$

SIS and LWE: SVP/CVP on Random **q**-ary Lattices



SIS and LWE: SVP/CVP on Random **q**-ary Lattices

SIS = **Approx SVP** in random **q**-ary lattices

LWE = **Approx CVP** in random **q**-ary lattices

For both, lattice described by $\mathbf{A} \in \mathbb{Z}_q^{n \times m}, m \gg n$

Corresponds to lattice

$$\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{x} : \mathbf{A} \cdot \mathbf{x} \bmod q = 0\}$$

Corresponds to lattice

$$\Lambda_q(\mathbf{A}) := \{\mathbf{A}^T \cdot \mathbf{s} + q\mathbf{k}\}$$

q-ary Lattices

Thm [Ajtai'96]: SIS is at least as hard **Approx SIVP** in general lattices

Thm [Regev'05]: LWE is *quantumly* at least as hard **Approx SIVP** in general lattices

Thm [Peikert'09, Brakerski-Langlois-Peikert-Regev-Stehlé'13]: LWE is *classically* at least as hard **Approx SIVP** in general lattices

Why might lattices be hard?

Extremely small approx. ratios are NP-hard (but crypto-relevant ones cannot be)

Worst-case-to-average-case reduction for SIS/LWE → good distribution of hard instances

While lattices are periodic, the period is already known. Instead the problem is to find a short representation of the period

Why might lattices be easy?

Lattices have tons of geometric structure that can potentially be exploited by quantum algorithms

For efficiency, only very special lattices (ideal/module, etc) are used. Understanding of quantum hardness is very limited

Parameters are set based on state-of-the-art attacks. Even a sub-exponential attack would break current standards

Existing attacks on lattices

Classical Attack: Lattice Sieving

[Ajtai-Kumar-Sivakumar'01, …]

Run time $2^{O(n)}$

Compare to brute-for search: $2^{O(n \log q)}$

Quantum Attack: Grover

Naïve Grover still gives $2^{O(n \log q)}$, worse than classical

Grover + Sieving: slightly better exponent in $2^{O(n)}$

[Laarhoven-Mosca-van de Pol'15,…]

Lattices as hidden shifts

[Regev'02]

LWE sample: $\mathbf{u} = \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \bmod q$

$$f_0(\mathbf{r}) = \left\lfloor \frac{\mathbf{A}^T\mathbf{r} \bmod q}{q} \right\rceil$$

$$f_1(\mathbf{r}) = \left\lfloor \frac{\mathbf{A}^T\mathbf{r} + \mathbf{u} \bmod q}{q} \right\rceil$$

$$= \left\lfloor \frac{\mathbf{A}^T(\mathbf{r}+\mathbf{s}) + \mathbf{e} \bmod q}{q} \right\rceil \approx \left\lfloor \frac{\mathbf{A}^T(\mathbf{r}+\mathbf{s}) \bmod q}{q} \right\rceil = f_0(\mathbf{r}+\mathbf{s})$$

Lattices as hidden shifts
[Regev'02]

Can we apply Kuperberg/Simon-meets-K?

Probably not: in usual settings $\mathbf{e}/q$ is inverse poly

➡ Rounding has an inverse-poly chance of changing answer

Kuperberg requires super-poly samples → almost certain some sample bad. In such bad cases, superposition collapses to classical value, which completely poisons entire algorithm

Lattices as hidden shifts
[Regev'02]

What if error rate $\mathbf{e}/q$ is $2^{-O(\sqrt{\log|\mathbb{G}|})} = 2^{-O(\sqrt{n \log q})}$?

Then with good probability no rounding errors → algorithm works!

But in this regime, already have good classical algorithms

Lattices as hidden shifts
[Regev'02]

What about Simon-meets-K? Needs $2^{-O(\log n \log q)}$ samples

Always >> q, so no regime where error rate is small enough

Lattices as hidden shifts
[Regev'02]

Kuperberg requires $2^{-O(\sqrt{\log |\mathbb{G}|})} = 2^{-O(\sqrt{n \log q})}$ samples

Simon-meets-K requires $2^{-O(\log n \log q)}$

Wishful thinking: maybe $2^{-O(\ (\log n)\sqrt{\log q}\ )}$ suffices?

➡ quasi-poly algorithm for quasi-poly noise-modulus ratio

Regev's Reduction (idea)

Goal: solve SIS (which we know allows for solving LWE)

Regev's Reduction (idea)

First prepare

$$|\psi_1\rangle := \sum_{\mathbf{s}} |\mathbf{A}^T \cdot \mathbf{s} \bmod q\rangle \qquad |\psi_2\rangle := \sum_{\mathbf{s},\mathbf{e}} \sqrt{N_{\frac{q}{2\sigma}}(\mathbf{e})}\, |\mathbf{e} \bmod q\rangle$$

$$1 \ll \sigma \ll q$$

Controlled sum:

$$|\psi_3\rangle := \sum_{\mathbf{s},\mathbf{e}} \sqrt{N_{\frac{q}{2\sigma}}(\mathbf{e})}\, |\mathbf{A} \cdot \mathbf{s} \bmod q, \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \bmod q\rangle$$

Regev's Reduction (idea)

$$|\psi_3\rangle := \sum_{\mathbf{s},\mathbf{e}} \sqrt{N_{\frac{q}{2\sigma}}(\mathbf{e})}\, |\mathbf{A}\cdot\mathbf{s} \bmod q, \mathbf{A}^T\cdot\mathbf{s}+\mathbf{e} \bmod q\rangle$$

Now pretend that LWE is easy, and compute:

$$|\psi_4\rangle := \sum_{\mathbf{s},\mathbf{e}} \sqrt{N_{\frac{q}{2\sigma}}(\mathbf{e})}\, |\mathbf{A}^T\cdot\mathbf{s}+\mathbf{e} \bmod q\rangle$$

Regev's Reduction (idea)

$$|\psi_4\rangle := \sum_{\mathbf{s},\mathbf{e}} \sqrt{N_{\frac{q}{2\sigma}}(\mathbf{e})}\, |\mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \bmod q\rangle$$

Next, apply QFT. By convolution theorem:

$$|\psi_5\rangle := \sum_{\mathbf{x}:\mathbf{A}\cdot\mathbf{x} \bmod q=0} \sqrt{N_\sigma(\mathbf{x})}\, |\mathbf{x}\rangle$$

Finally, measure to get a short vector in $\Lambda_q^\perp(\mathbf{A})$

Question: Can Regev's reduction be modified to eliminate the need to solve LWE?

Easier Q: Can we modify the primal problem (SIS) to make the dual problem (LWE) easy?

[Chen-Liu-Z'22]: Quantum attack on certain instances of SIS under infinity norm

[Yamakawa-Z'22]: Quantum advantage w/o "structure"

[Jordan-Shutty-Wootters-Zalcman-Schmidhuber-KingiOsakov-Khattar-Babbush'25,…]: Advantage for optimization

4. Cryptography from codes

Rough idea of McEliece

Choose linear ECC with efficient decoding $\mathbf{C} \in \mathbb{F}_q^{n \times m}$

Obfuscate code as $\mathbf{C}' = \mathbf{LCR}$

$\mathbf{L} =$ Random invertible matrix

$\mathbf{R} =$ Random permutation matrix

Secret key: $\mathbf{L}, \mathbf{C}, \mathbf{R}$, public key: $\mathbf{C}' = \mathbf{LCR}$

Rough idea of McEliece

Encrypt 0: $\mathbf{s}^T\mathbf{C}' + \mathbf{e}^T$ where $\mathbf{e}$ has small Hamming weight

Encrypt 1: random word

Decrypt $\mathbf{u}$: $\mathbf{u}^T\mathbf{R}^{-1} = (\mathbf{s}^T\mathbf{L})\mathbf{C} + \mathbf{e}^T\mathbf{R}^{-1}$

small Hamming weight

➡ Can decode

As far as I know, best attacks treat public key as an arbitrary linear code, ignoring all structure

Best classical attacks: based on ISD, $2^{O(n)}$

Best quantum attacks: ISD+Grover, $2^{O(n)}$

Improvement much less than quadratic

Code-based crypto as lattice with different noise model:

Small Hamming vs small L2 norm

Regev+Kuperberg?

Unfortunately, no analog of rounding to eliminate Hamming error (as far as I know)

Code-based as group action: public key obtained by acting on $\mathbf{C}$ by symmetric group. Can always canonicalize away $\mathbf{L}$ by row-reduction

If $\mathbf{C}$ known, reduces to hidden shift for symmetric group

In typical proposals $\mathbf{C}$ unknown, and symmetric group hidden subgroup may be hard anyway

Learning Parity with Noise

$$\mathbf{u} = \mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \bmod 2$$

Low Hamming weight

Benefit: no structure to $\mathbf{A}$, so maybe "more secure"

[Alekhnovich'03]

Downside: no known trapdoor. Known PKE schemes require very small noise rate; maybe less secure

5. Other

Multivariate Quadratics

Let $P : \mathbb{F}^m \to \mathbb{F}^n$ be a surjective, easily invertible degree-2 polynomial function

Public key: $A_1 \circ P \circ A_2$ for secret affine A_1, A_2

Secret key: A_1, A_2, P allows for inverting

Good enough for hash-and-sign signatures

Multivariate Quadratics

Attacks typically try to recover A_1, A_2 from $A_1 \circ P \circ A_2$

Classical: Groebner basis methods
Quantum: Maybe small speedups using Grover

Hash-Based Cryptography

Just use standard hash-functions for crypto. Best known attacks are Grover/BHT

Yields:

- Secret key crypto
- Digital signatures

Unlikely to give public key encryption

Non-abelian Cryptography

Goal: use a non-abelian group for key exchange/public key encryption, to avoid Shor

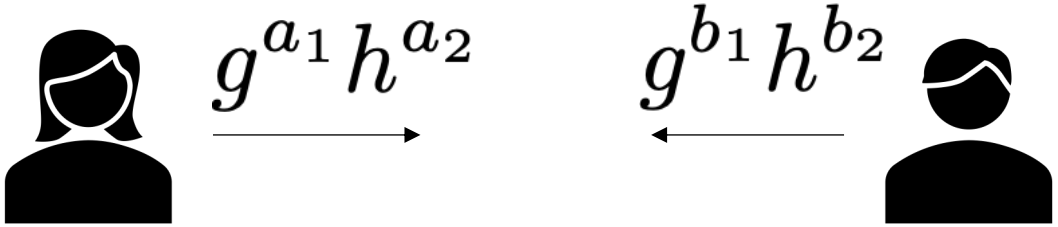
Challenge: typically, Alice and Bob generate the shared key using different operations

In group based settings, this usually corresponds to group being abelian. Non-abelian cryptosystems often make assumptions about group structure

Non-abelian Cryptography

Only fully-generic example I'm aware of: Stickle key exchange (though no known "platform groups")

Let g, h be high-order non-commuting group elements



$$k = g^{a_1 b_1} h^{a_2 b_2}$$

Easy to view as group action with $\mathbb{Z}^2$ acting on $\mathbb{G}$

Thanks!