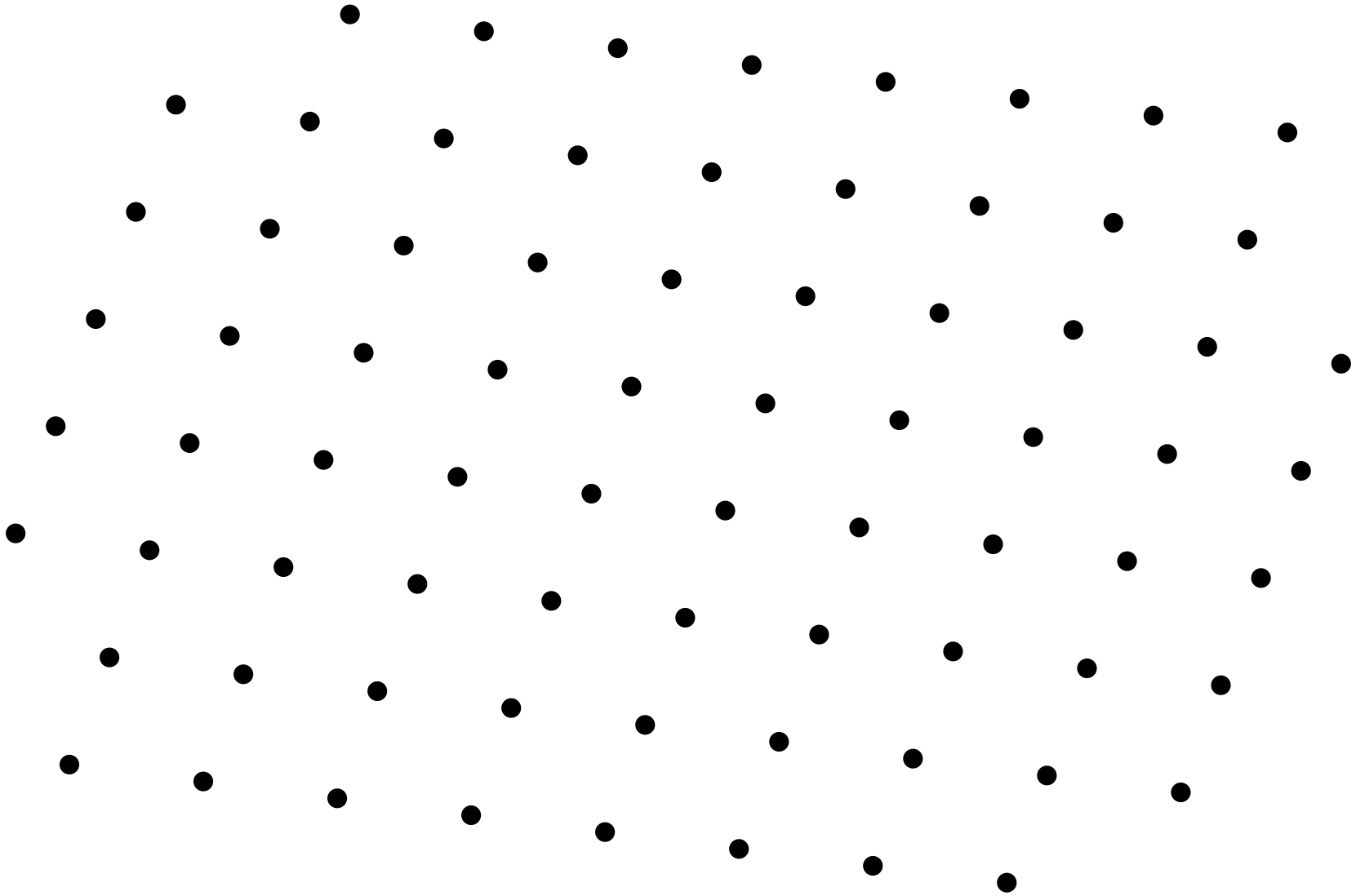
Some Applications of the QFT Convolution Theorem

Mark Zhandry (NTT Research & Stanford University)

Based on joint works with Yilei Chen, Qipeng Liu, and Takashi Yamakawa

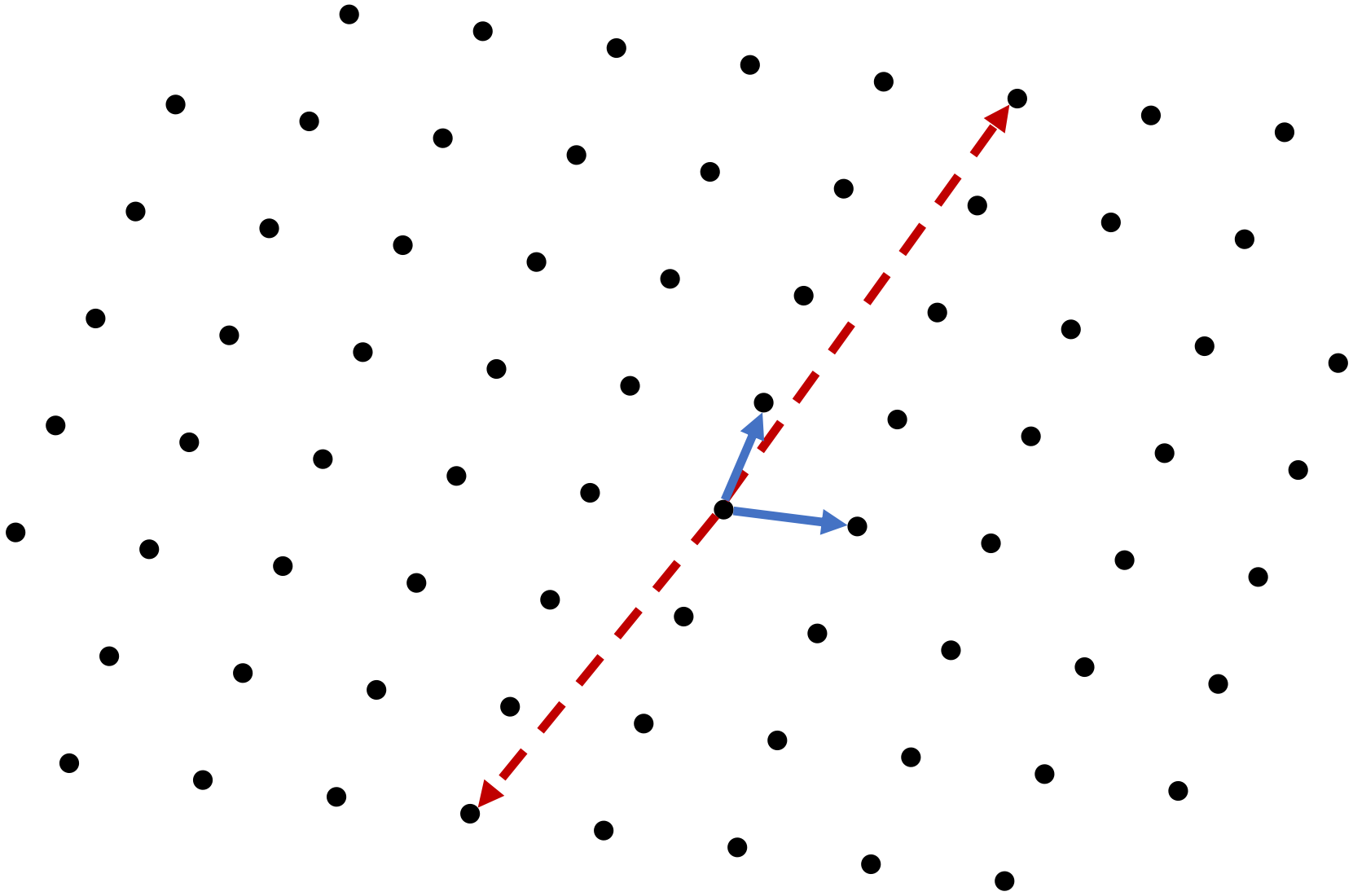
Part 1: Regev’s Reduction

Lattices



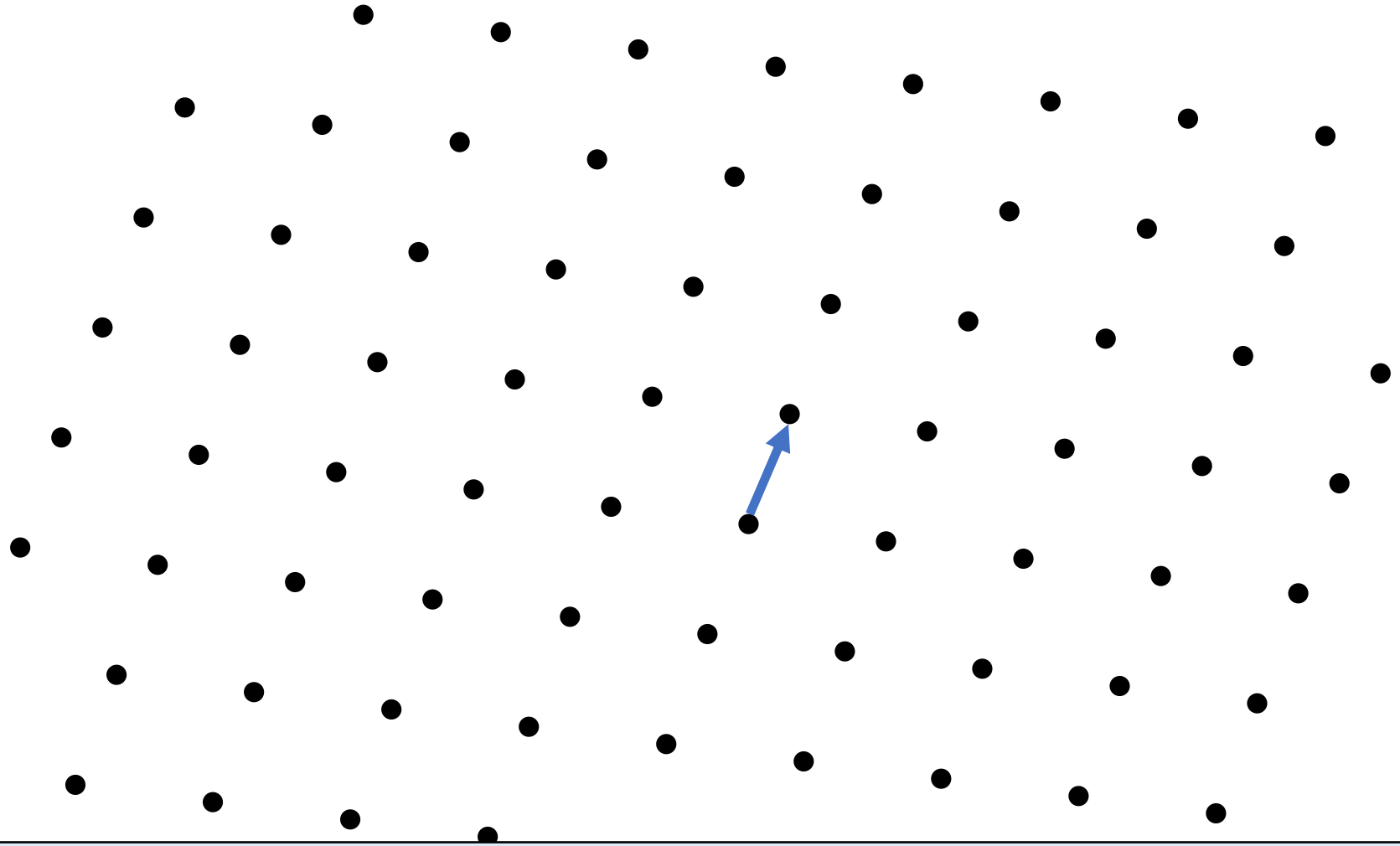
Imagine dimension in the 100s

Lattices



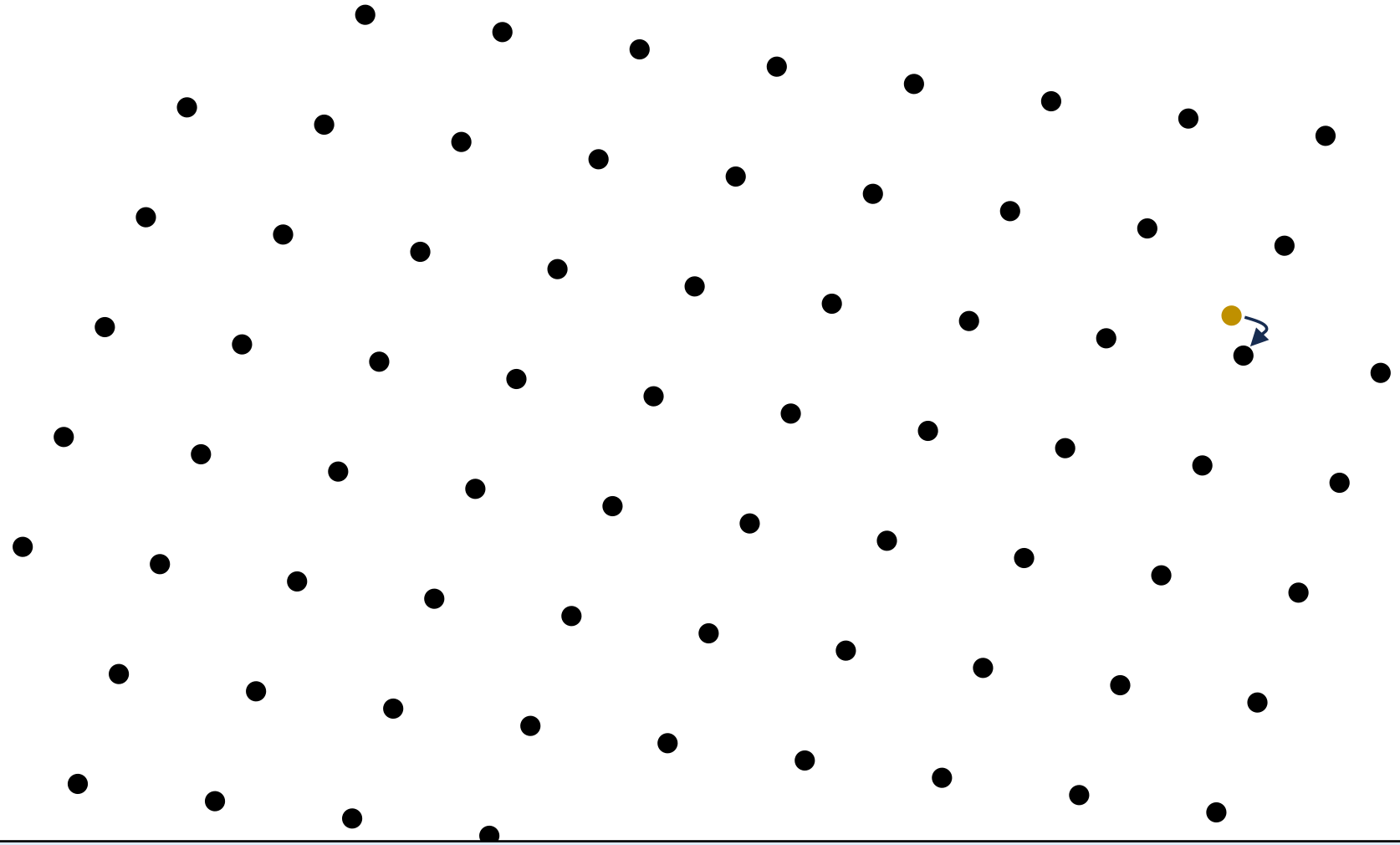
Basis: minimal set of vectors that generate lattice

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

Numerous applications:

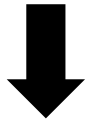
- Disproving Merten's conjecture [Odlyzko-te Riele'85]
- Finding "close" algebraic relations

  $x \rightarrow a,b,c$ s.t. $a x^2+b x+c \approx 0$
- (Classical) cryptanalysis [Shamir'82, Coppersmith'96]
- Cryptography with amazing functionalities

  (e.g. computing on encrypted data [Gentry'08,…])
- **Crypto with post-quantum security**

Why Should Lattice Problems be Quantum Hard?

Lattices are periodic, but lattice/period typically known (a basis); SVP/SIS asks to find *short* description (short basis) of period



Period-finding itself doesn't seem relevant

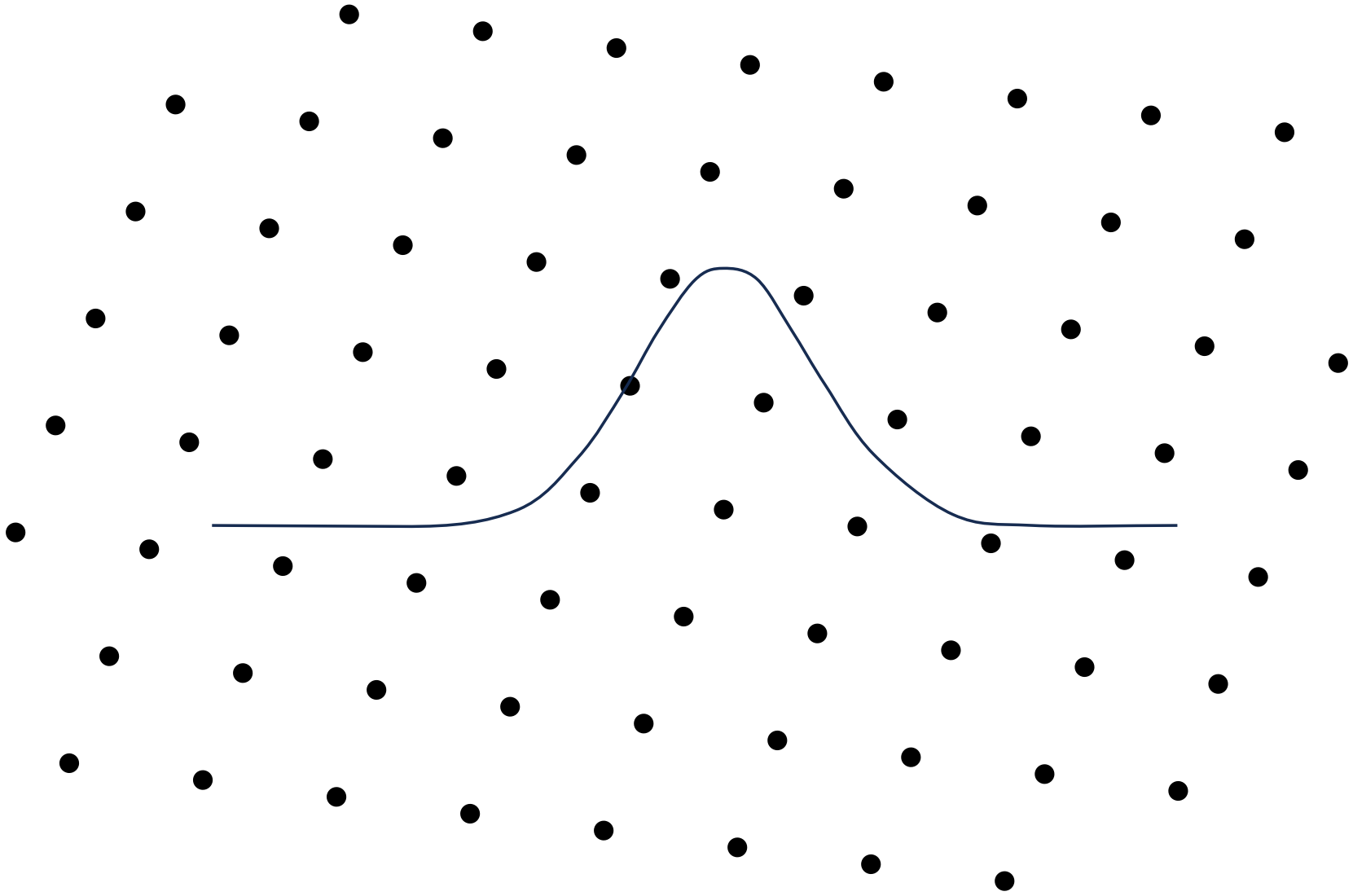
But, maybe there are other algorithms?

Regev's "Algorithm" for Approx SVP

[Regev'05]

Step **n=4**: Construct superposition over short lattice points, then measure

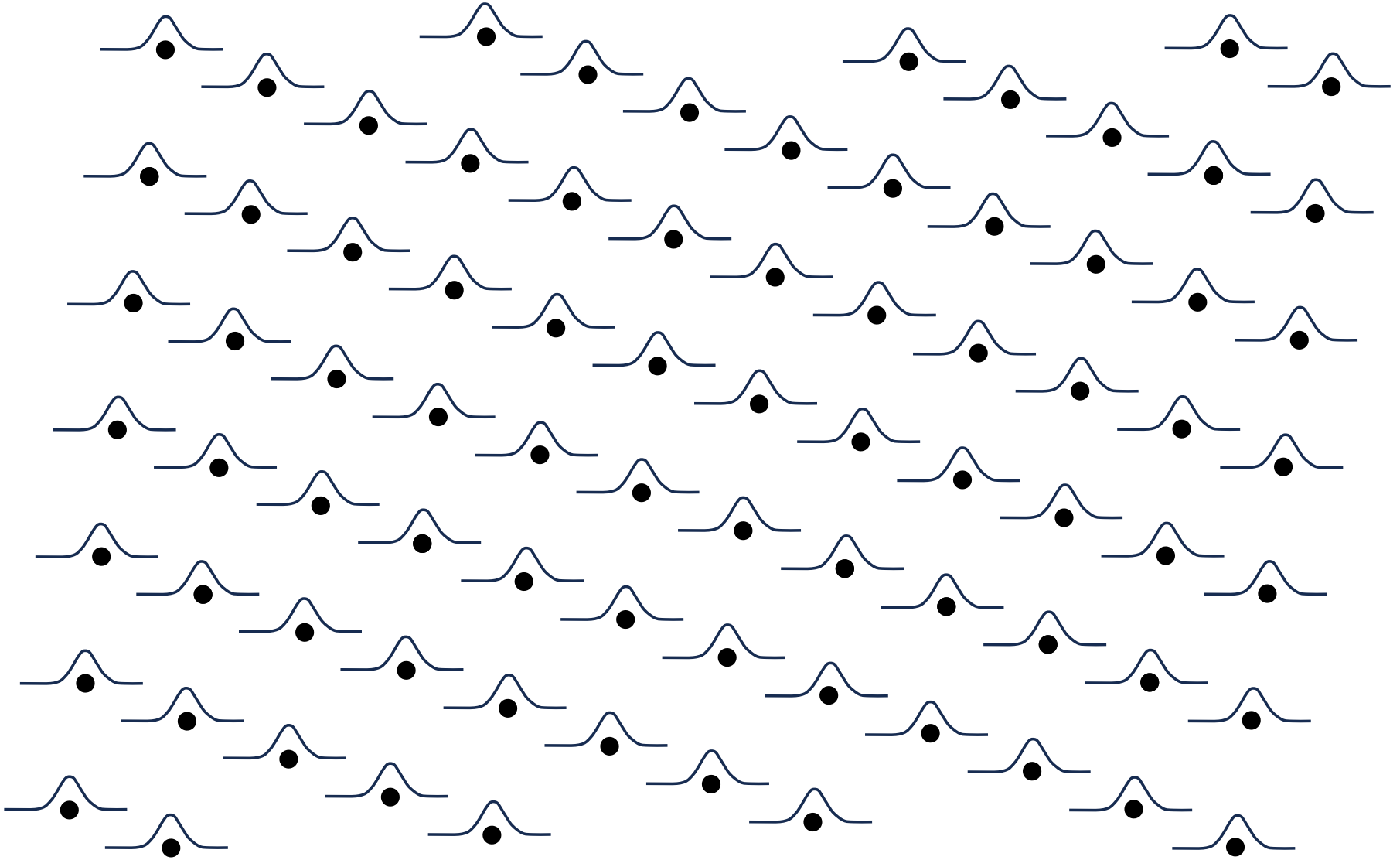
Regev's "Algorithm"



Regev's "Algorithm"

Step **3**: Construct superposition over points close to *dual* lattice → perform QFT to get **Step 4**

Regev's "Algorithm"



Regev's "Algorithm"

Short lattice points  Close to dual lattice points

QFT

Convolution Thm:

$$\sum_x \alpha_x \beta_x |x\rangle \quad \overset{\text{QFT}}{\Longleftrightarrow} \quad \sum_{x,y} \hat{\alpha}_x \hat{\beta}_y |x+y\rangle$$

α_x = "is lattice point"

β_x = "is short"

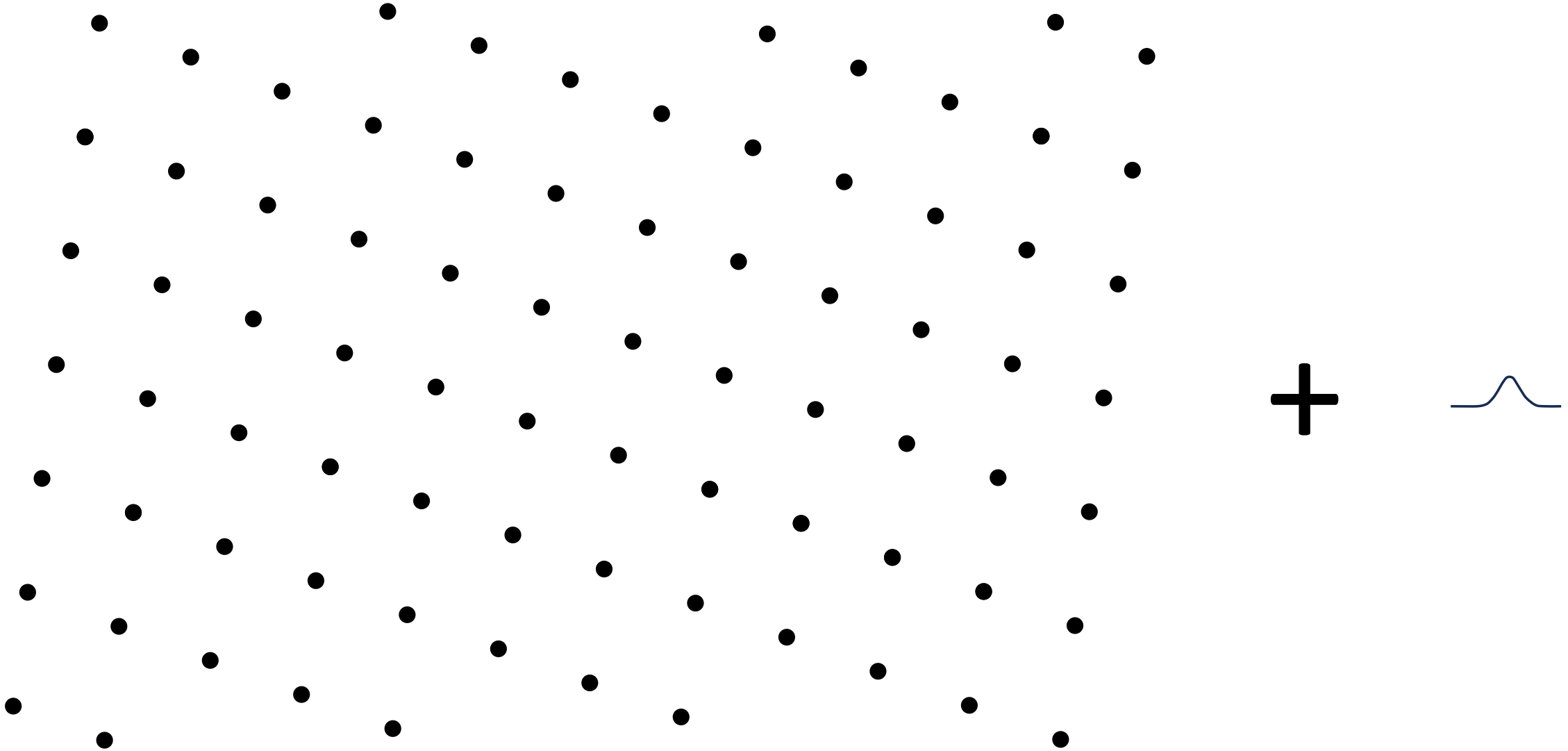
$\hat{\alpha}_x$ = "is dual lattice point"

$\hat{\beta}_x$ = "is short"

Regev's "Algorithm"

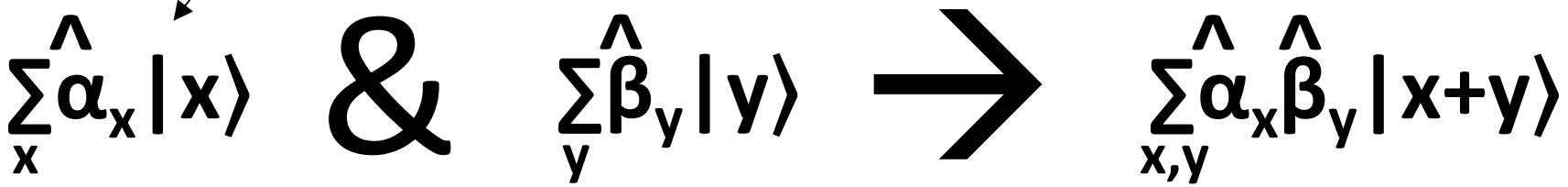
Step **1**: Construct superposition over dual lattice
+ construct superposition over short vectors,
then add superpositions

Regev's "Algorithm"



Regev's "Algorithm"

Separately easy to construct



$\hat{\alpha}_x$ = "is dual lattice point"

$\hat{\beta}_x$ = "is short"

Regev's "Algorithm"

Needed to ensure reversibility of addition

$$\sum_x \hat{\alpha}_x |x\rangle \;\&\; \sum_y \hat{\beta}_y |y\rangle \rightarrow \sum_{x,y} \hat{\alpha}_x \hat{\beta}_y |\, x,\, x+y\rangle$$

$\hat{\alpha}_x$ = "is dual lattice point"

$\hat{\beta}_x$ = "is short"

Regev's "Algorithm"

Step **2**: Eliminate **x** by "decoding" **x+y**

Regev's "Algorithm"

$$\sum_{x,y} \hat{\alpha}_x \hat{\beta}_y |\, x,\, x+y\rangle \;\Rightarrow\; \sum_{x,y} \hat{\alpha}_x \hat{\beta}_y |\, x+y\rangle$$

In Regev's case, **x+y** is close (dual) lattice point, **x** is lattice point

⇒ Can solve via CVP, but presumed hard!!!

But, we actually have a potentially much easier problem…

Regev's "Algorithm"

Learning with errors (LWE): solving CVP when point & errors are random, and errors are very small

(Un)fortunately, LWE appears hard, even quantumly

Instead, Regev's "Algorithm" is viewed as a justification for the hardness of LWE

(if it wasn't hard, then (approx.) SVP would be easy)

Regev's "Algorithm"

Since Regev's work, LWE has become a major tool for the design of cryptosystem

Part 2: Completing Regev's algorithm in extreme cases

[Chen-Liu-**Z**'22]

A special case of (approx.) SVP

$$A \in \mathbb{Z}_q^{n \times m}$$

$$m >> n$$

$$L_A = \{x : A.x \bmod q = 0\}$$

Thm (trivial): Can efficiently find $x \in L_A$ with $\| x \|_\infty \leq q/2$
Proof: Gaussian elimination mod $q \rightarrow x' \in \mathbb{Z}_q^m$ s.t. $A.x \bmod q = 0$ Coordinate of x = representative of x' in $\{-(q-1)/2,\ldots,(q-1)/2\}$

A special case of (approx.) SVP

$$A \in \mathbb{Z}_q^{n \times m}$$

$$m >> n$$

$$L_A = \{x : A.x \bmod q = 0\}$$

Thm (trivial): Can efficiently find $x \in L_A$ with $|x|_\infty \leq q/2$

Can we do any better?

A special case of (approx.) SVP

$$A \in \mathbb{Z}_q^{n \times m}$$

$$m >> n$$

$$L_A = \{x : A.x \bmod q = 0\}$$

Thm (trivial): Can efficiently find $x \in L_A$ with $|x|_\infty \leq q/2$

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $x \in L_A$ with $|x|_\infty \leq (q-c)/2$, assuming $m > n^{c+1}\,\mathrm{poly}(q)$

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Attempt Regev's algorithm with:

$$\alpha_x = \text{“is in } L_A \pmod q\text{”}$$

$$\beta_x = \text{“}|\, x \,|_\infty \le (q-c)/2\text{”}$$

$$\left(\sum_{x=-(q-c)/2}^{(q-c)/2} |x\rangle \right)^{\otimes m}$$

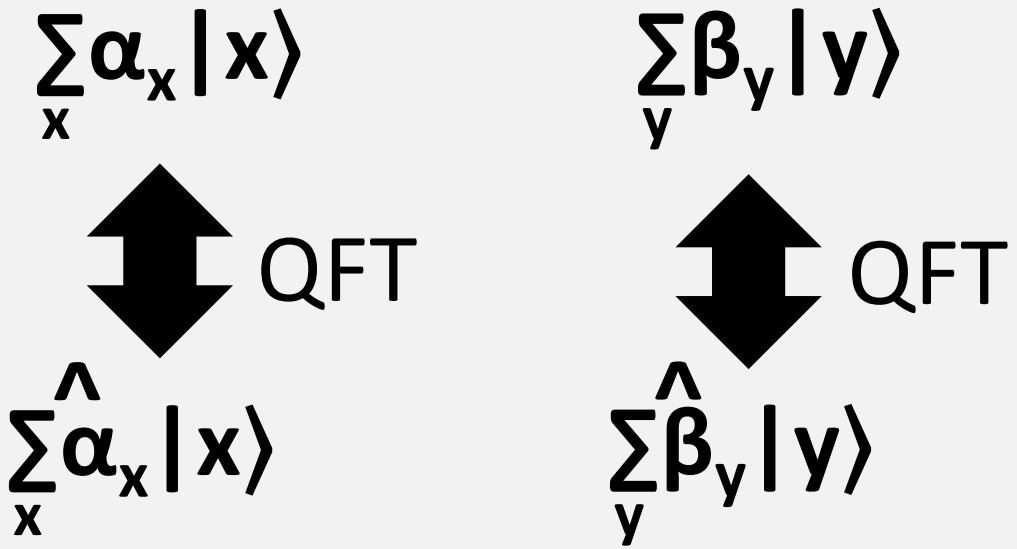
Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: separately construct

$$\sum_x \alpha_x |x\rangle \qquad \sum_y \beta_y |y\rangle$$

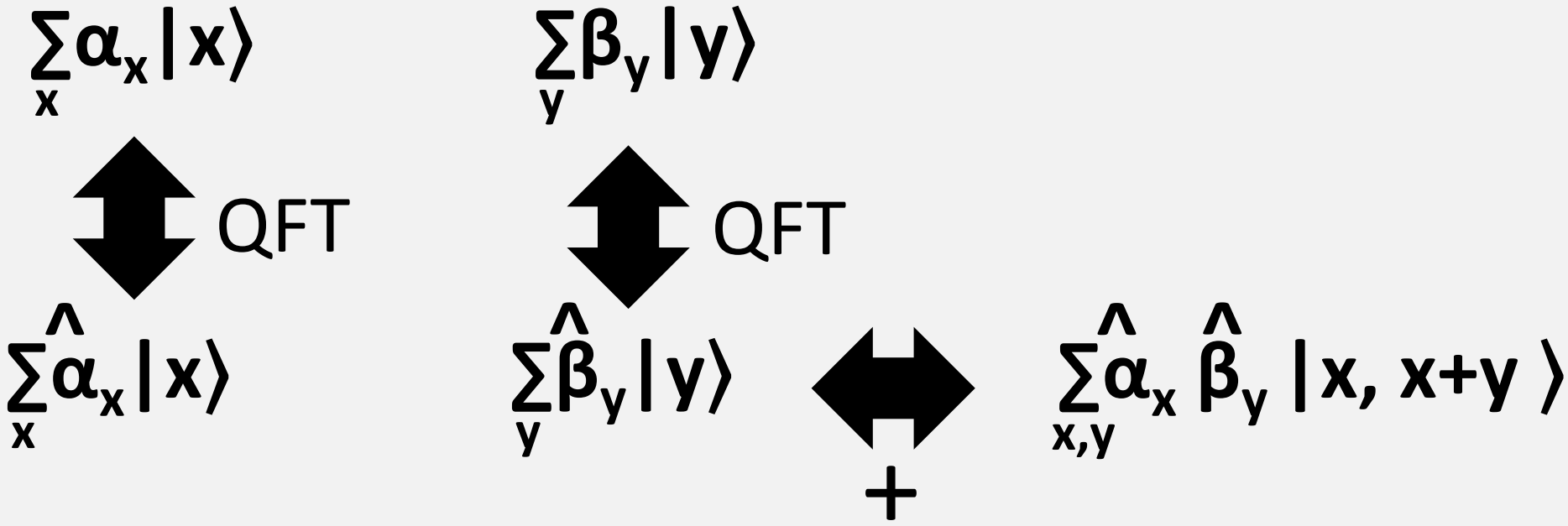
Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:



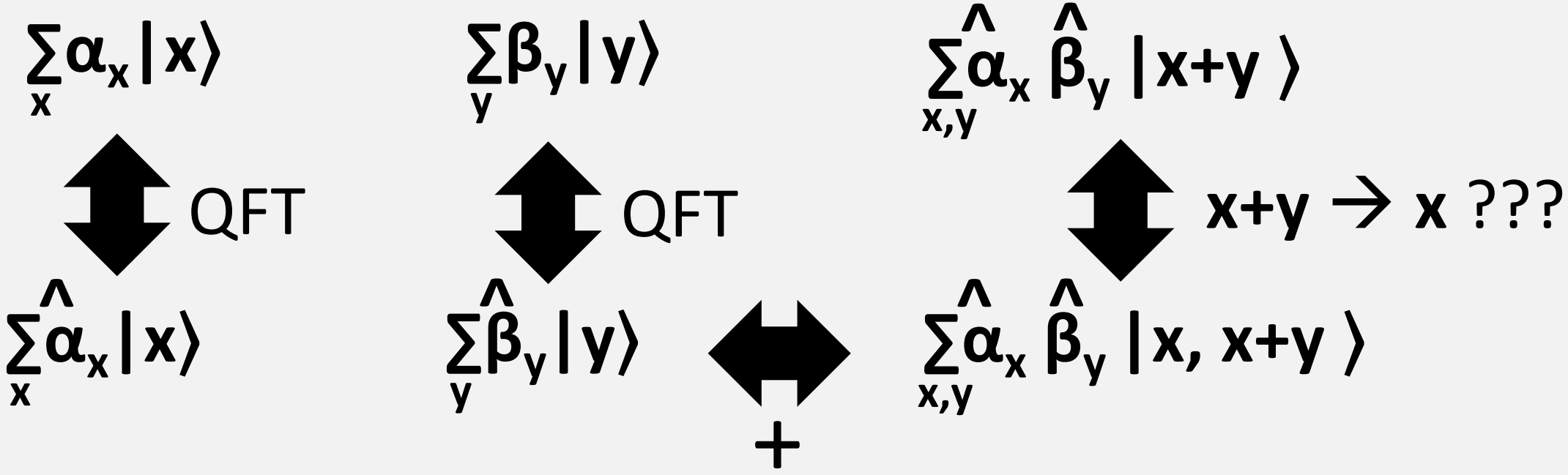
Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:



Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:

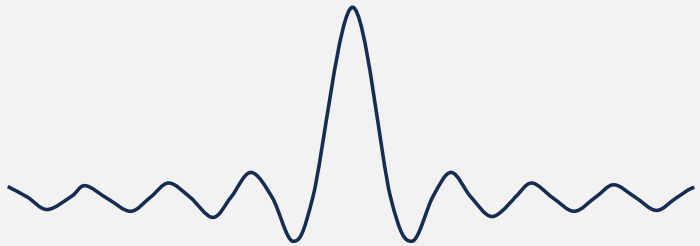


Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x} \in L_A$ with $|\mathbf{x}|_\infty \leq (q-c)/2$, assuming $m > n^{c+1}\,\mathrm{poly}(q)$

Proof: Our decoding problem:

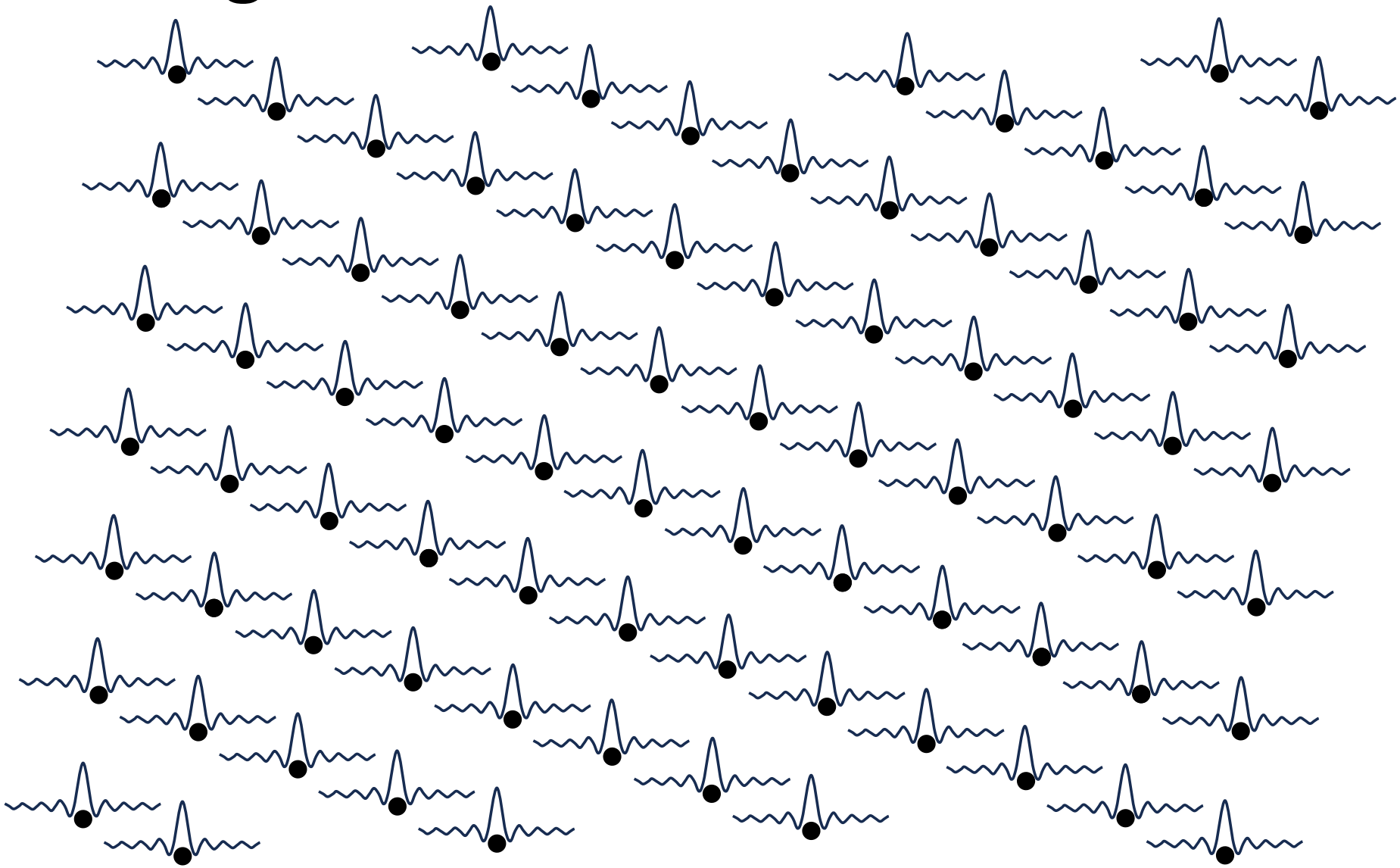
$$\alpha_x = \text{“}\mathbf{x} = \mathbf{A}^T.\mathbf{r} \bmod q\text{”}$$

$\beta_x =$



Still seems hard to decode

Our Decoding Problem



Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\ poly(q)}$

Proof: Idea 1 - Look at superposition over errors

$$\sum_y \hat{\beta}_y \,|A.r+y\rangle \quad \xrightarrow{\text{Goal}} \quad r$$

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Zoom in on one coordinate

$$\sum_y \hat{\beta}_y \,|\, d+y \,\rangle$$

Goal ⟹ **d**

Search space is small, so maybe easy

If we can find **d** for all coordinates, we can find **r** via linear algebra

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Zoom in on one coordinate

$$|\Psi_d\rangle = \sum_y \hat{\beta}_y\, |d+y\rangle \quad \xrightarrow{\text{Goal}} \quad d$$

Problem: $|\Psi_d\rangle$ are not orthogonal (only have rank **q-c**)

➡ Impossible to compute **d** perfectly

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Idea 2: Unambiguous state discrimination

$$|\Psi_d\rangle = \sum_y \hat{\beta}_y \, |d+y\rangle$$

Project onto $\mathbf{Span(\{ |\Psi_d\rangle \}_{-(q-c)/2+1 \le d \le (q-c)/2})}$

Accept

Reject

Don't learn much :-(

$\mathbf{(q-c)/2+1 \le d \le q-(q-c)/2=(q+c)/2}$

set **S** of size **c**

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:

A ,

Accept
Reject
Reject
Accept
Reject
…

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:

$\mathbf{a_1}$, Accept

$\mathbf{a_2}$, Reject

$\mathbf{a_3}$, Reject

$\mathbf{a_4}$, Accept

$\mathbf{a_5}$, Reject

…

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof:

$\mathbf{a_2}$, Reject
$\mathbf{a_3}$, Reject
$\mathbf{a_5}$, Reject
} $\mathbf{d_i = a_i \,.\, r \in S}$
…

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Idea 3: Re-linearization a la Arora-Ge

$$\mathbf{d_i \in S} \quad \Rightarrow \quad \underbrace{\mathbf{\prod_{s \in S}(d_i - s) = 0}}_{\text{Degree } \mathbf{c}}$$

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Idea 3: Re-linearization a la Arora-Ge

$$\mathbf{d_i \in S} \Rightarrow \underbrace{\mathbf{\prod_{s \in S}(d_i - s) = 0}}_{\text{Degree } \mathbf{c}}$$

Each reject gives us degree **c** polynomial in **r**

$\Rightarrow$ Linear in degree-**c** monomials

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

Proof: Each reject → linear constraint over degree-**c** monomials

$O(n^c)$ such monomials

Intuitively, once we have **$O(n^c)$** constraints (rejects), should be able to compute monomials with linear algebra

⬇

r

Thm [Chen-Liu-Z'22]: For any constant **c**, can quantumly efficiently find $\mathbf{x \in L_A}$ with $\mathbf{|\, x \,|_\infty \le (q-c)/2}$, assuming $\mathbf{m > n^{c+1}\, poly(q)}$

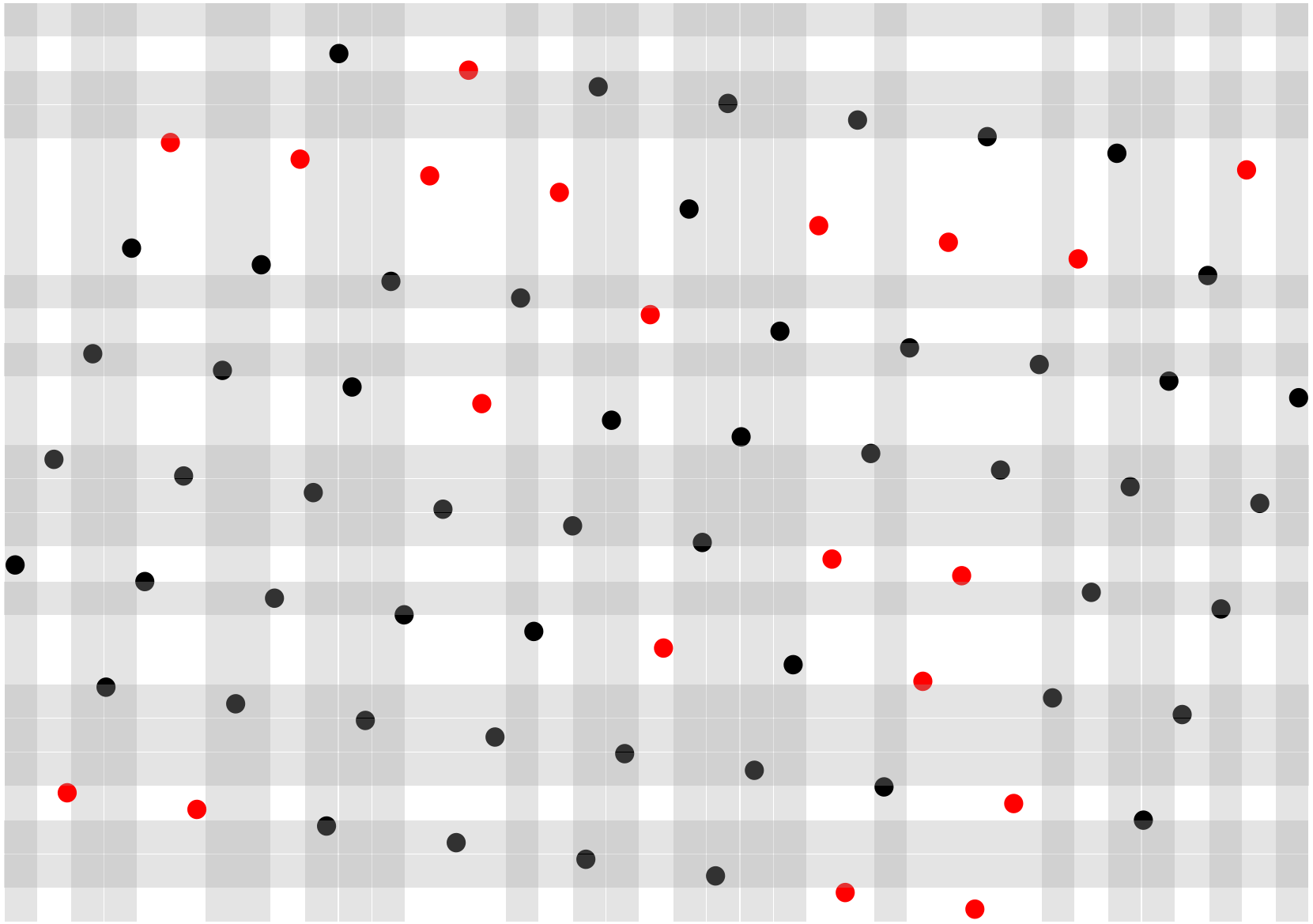
Proof:

Issues to work out:

- Probability of reject
- Guarantee re-linearized system is uniquely solvable

Part 3: Completing Regev's algorithm using ECCs

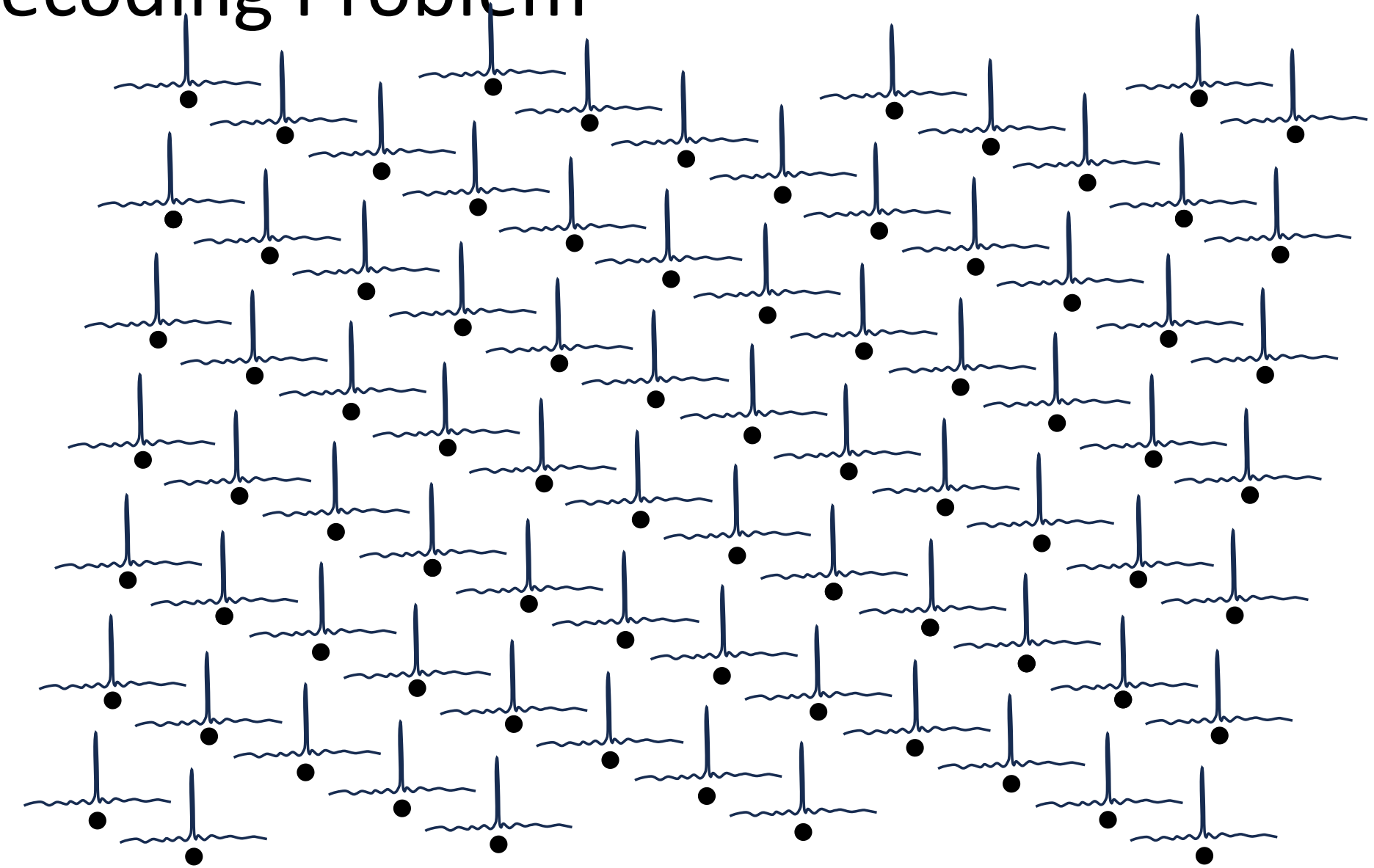
[Yamakawa-**Z**'22]



$f_i: \Sigma \rightarrow \{0,1\}$

$f_i(x_i) = 0$

Our Decoding Problem



Assuming $\mathbf{f}_i$ ”random-looking”

Thm [Yamakawa-Z'22]: If **A** is generator matrix for a "good" error-correcting code and each $\mathbf{f}_i$ are "random-looking", then can solve decoding problem

→ can find points in $\mathbf{L}_\mathbf{A}$ such that $\mathbf{f}_i(\mathbf{x}_i) = \mathbf{0}$

Thm [Yamakawa-Z'22]: If **A** is *parity-check* matrix for a "good" error-correcting code, no efficient classical algorithm making "black-box" queries to $\mathbf{f}_i$

Reasonable model for cryptographic hash functions

Consequences

Relative to random oracle, or alternatively under assumption about hash functions,
∃ **NP**-search problem in **BQP \ BPP**

Public verification previously unknown except under very strong assumptions

Hash functions considered much milder than prior assumptions (e.g. Factoring)

Additionally under Aaronson-Ambainis conjecture or appropriate hash function assumption, ∃ publicly verifiable certified randomness