Codes & Lattices: Computational Complexity and Constructions

Thesis Defense

May 27, 2025

Alexandra Veliche Hostetler

Outline

0. Introduction

I. Computational Complexity

- Fine-Grained Hardness of Learning With Errors
- Reductions Between Code Equivalence Problems

II. Constructions and Algorithms

- List-Decoding Reed-Solomon Codes over General Norms

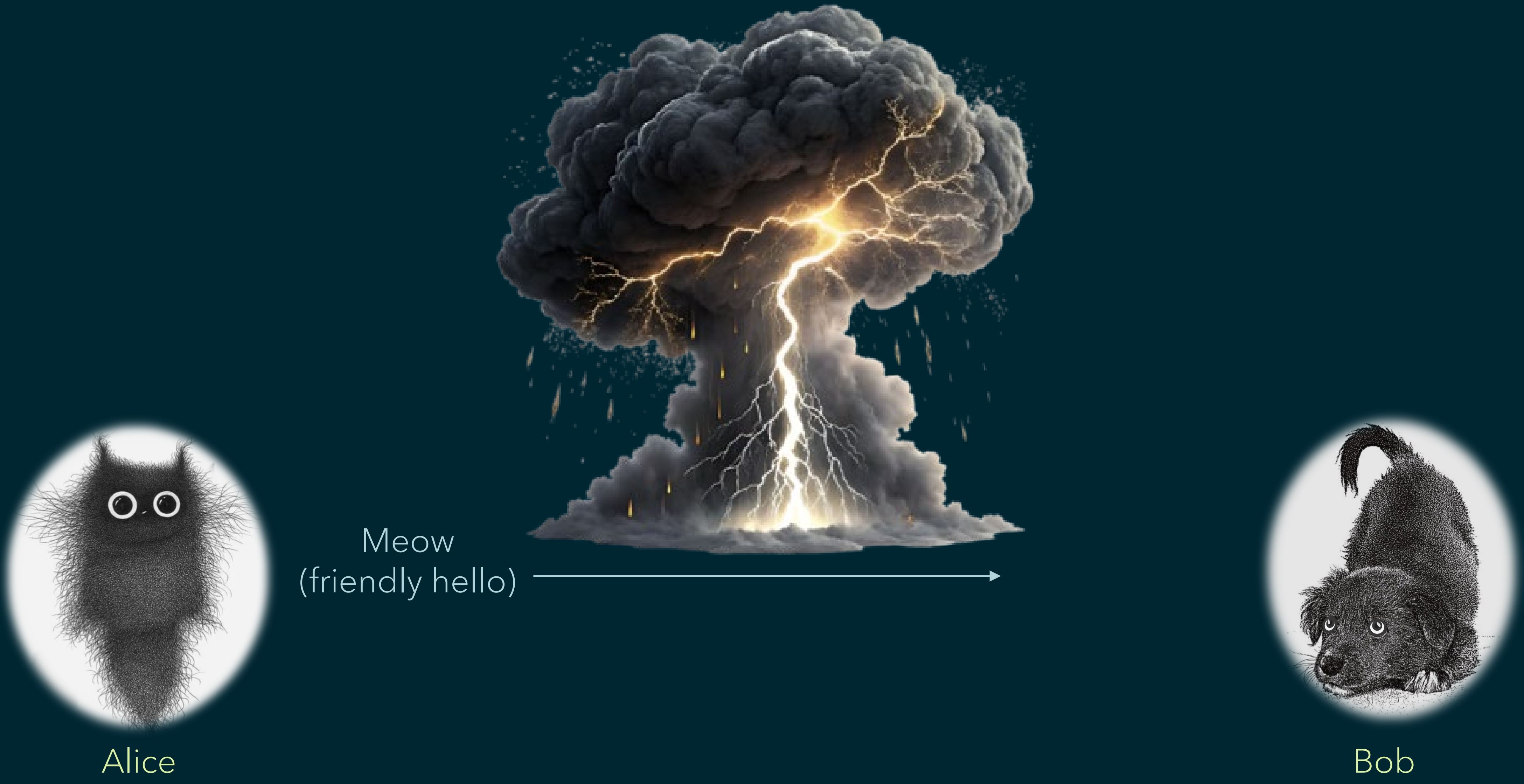
Alice

Bob

Meow
(friendly hello)

Alice

Bob


Meow
(friendly hello)
Alice
Bob


Meow
(friendly hello)
Mrrreoww
(scared threat)
Alice
Bob


Meow
(friendly hello)
Alice
Eve
Bob


Meow
(friendly hello)
Mrreeow
(mean threat)
Alice
Eve
Bob

Coding Theory:
Reliable Communication

Cryptography:
Secure Communication

Two objects frequently used in both areas:

linear codes and **lattices**

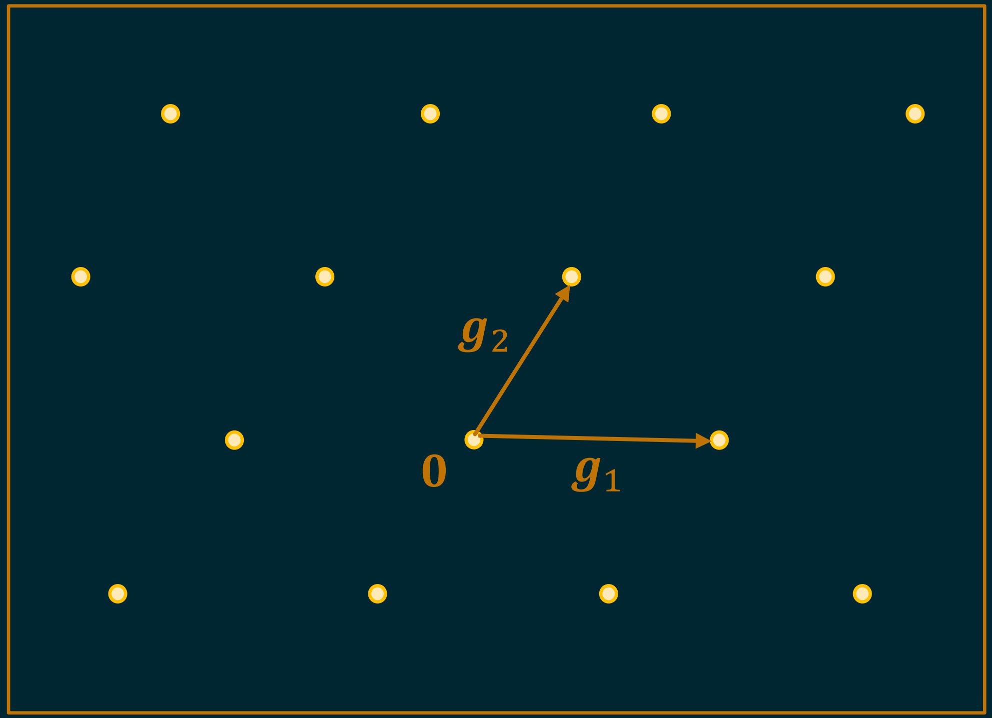
Codes

Linear Code:

A linear subspace over a finite field $\mathbb{F}_q$

$$\mathcal{C} = \left\{ a_1 \boldsymbol{g}_1 + \cdots + a_k \boldsymbol{g}_k \,:\, a_i \in \mathbb{F}_q \right\} \subseteq \mathbb{F}_q^n$$

of *generator* vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k \in \mathbb{F}_q^k$.



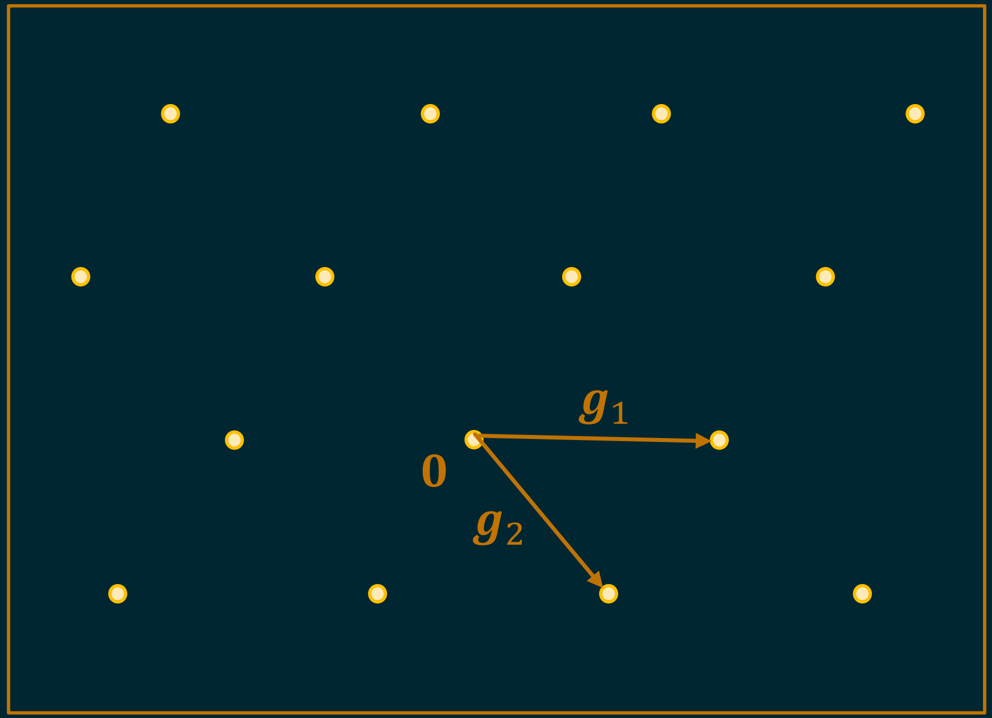
Codes

Linear Code:

A linear subspace over a finite field $\mathbb{F}_q$

$$\mathcal{C} = \{\mathbf{G}\,\mathrm{x} : \mathrm{x} \in \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^n \ .$$

There are many possible *generators* $\mathbf{G} = \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} \in \mathbb{F}_q^{k \times n}$.



Codes

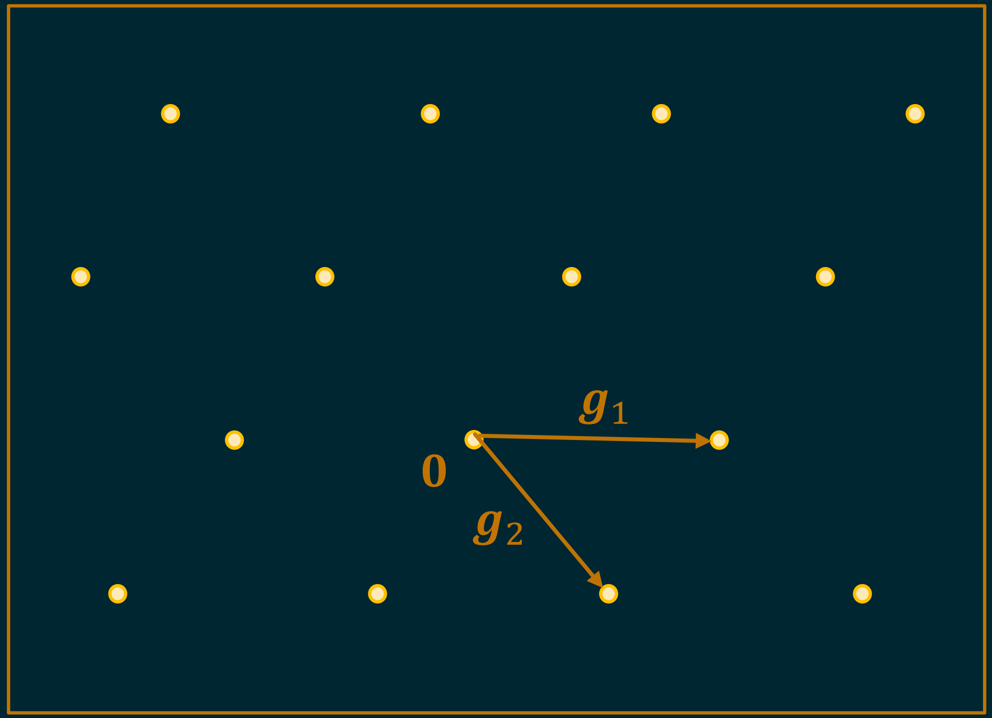
Linear Code:

A linear subspace over a finite field $\mathbb{F}_q$

$$\mathcal{C} = \{\mathbf{G}\,\mathrm{x} : \mathrm{x} \in \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^n \;.$$

There are many possible *generators* $\mathbf{G} = \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} \in \mathbb{F}_q^{k \times n}$.

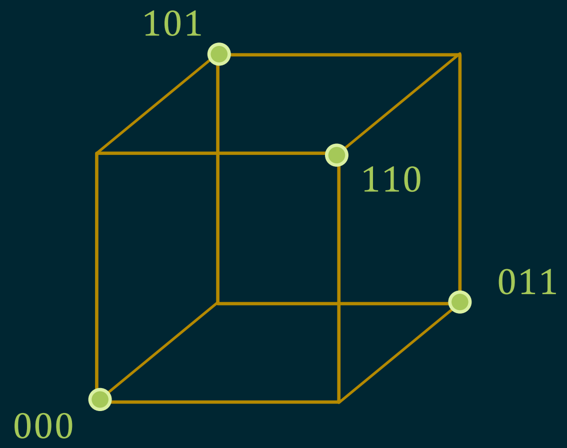
n is the *blocklength* and k is the *dimension.*



Codes

Linear Code:

ex: (over $\mathbb{F}_2^3$)



Lattices

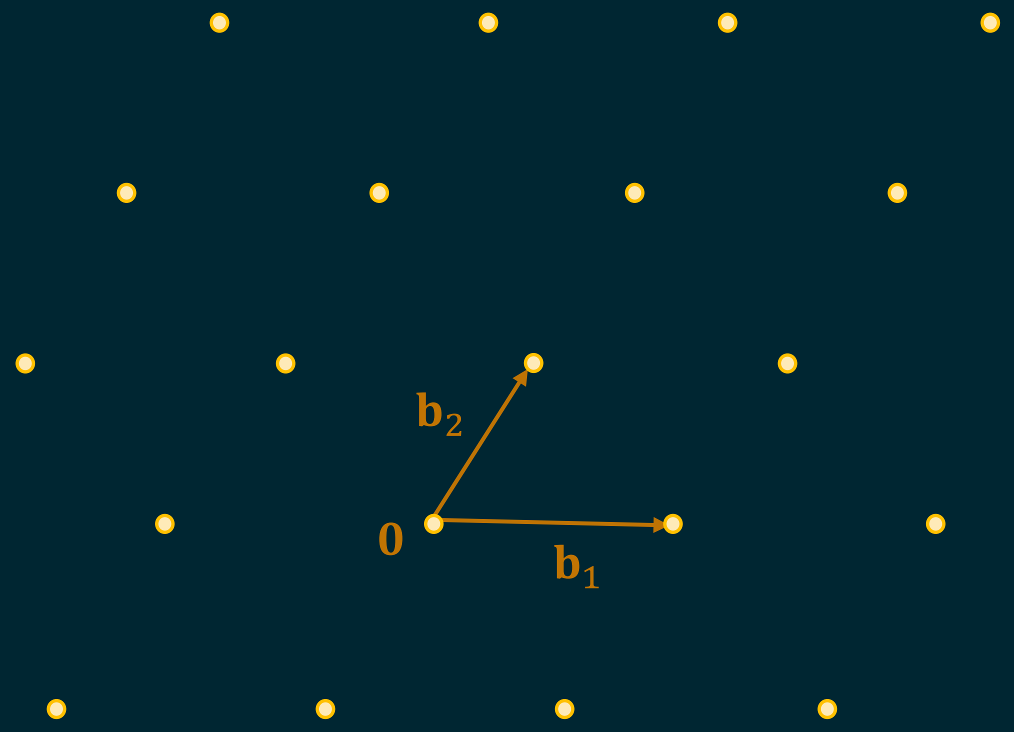
<u>Lattice:</u>

An infinite discrete set of vectors in $\mathbb{R}^n$

consisting of all integer linear combinations

$$\mathcal{L} = \{a_1\mathbf{b}_1 + \cdots + a_k\mathbf{b}_k \; : \; a_1, \ldots, a_k \in \mathbb{Z}\} \subset \mathbb{R}^n$$

of linearly independent *basis* vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$.



Lattices

Lattice:

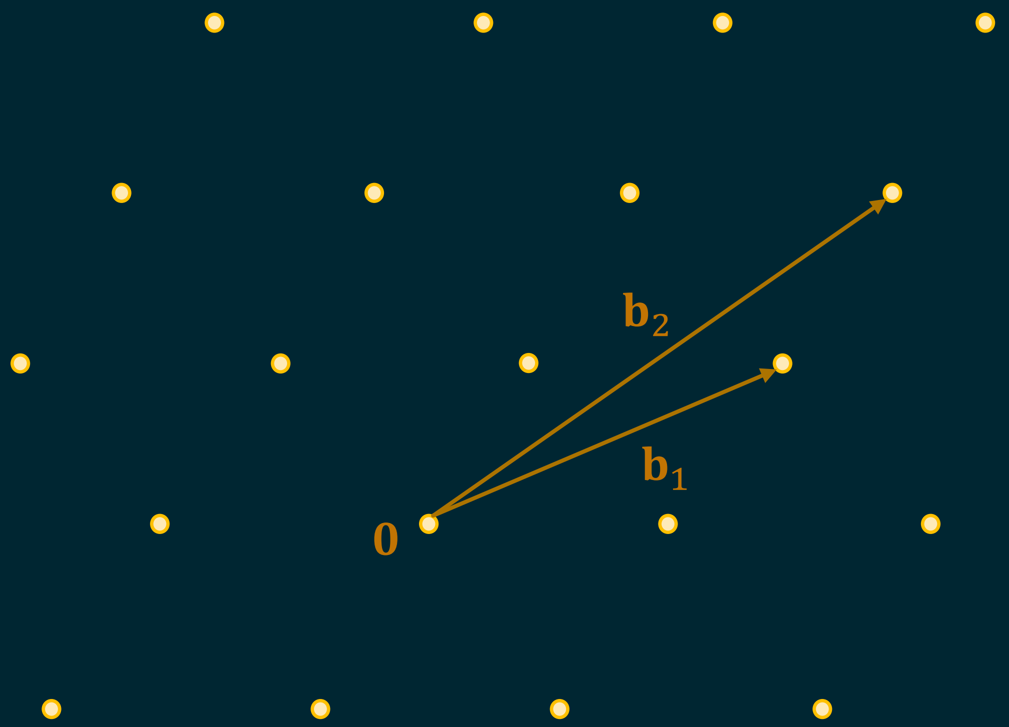
An infinite discrete set of vectors in $\mathbb{R}^n$

consisting of all integer linear combinations

$$\mathcal{L} = \{ \mathbf{B}\,\mathbf{x} : \mathrm{x} \in \mathbb{Z}^n \} \subset \mathbb{R}^n$$

of linearly independent *basis* vectors $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{R}^n$.

There are many possible *bases* $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_k]$.



Lattices

Lattice:

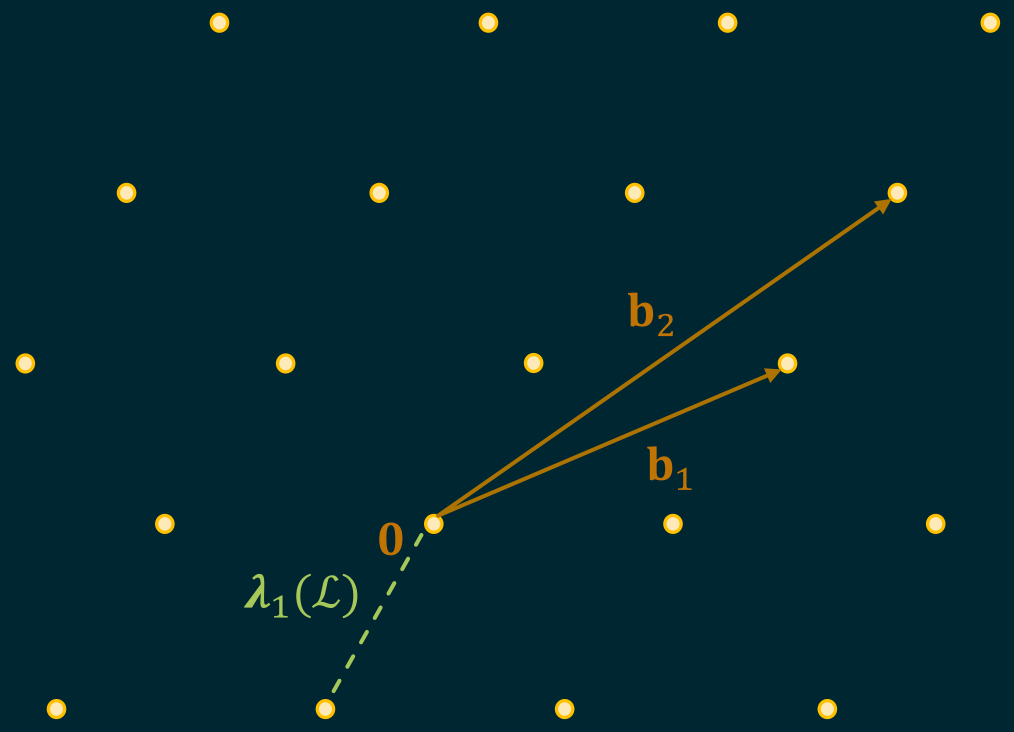
An infinite discrete set of vectors in $\mathbb{R}^n$

consisting of all integer linear combinations

$$\mathcal{L} = \{\, \mathbf{B}\,\mathbf{x} : \mathrm{x} \in \mathbb{Z}^n \,\} \subset \mathbb{R}^n$$

of linearly independent *basis* vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$.

The shortest distance between two lattice points is $\lambda_1(\mathcal{L})$.



Lattice Problems:

SVP — find the shortest lattice vector

CVP, **BDD** — find the closest lattice vector

GapSVP — decide how large is the shortest distance

LIP — decide if two lattices are isomorphic

Code Problems:

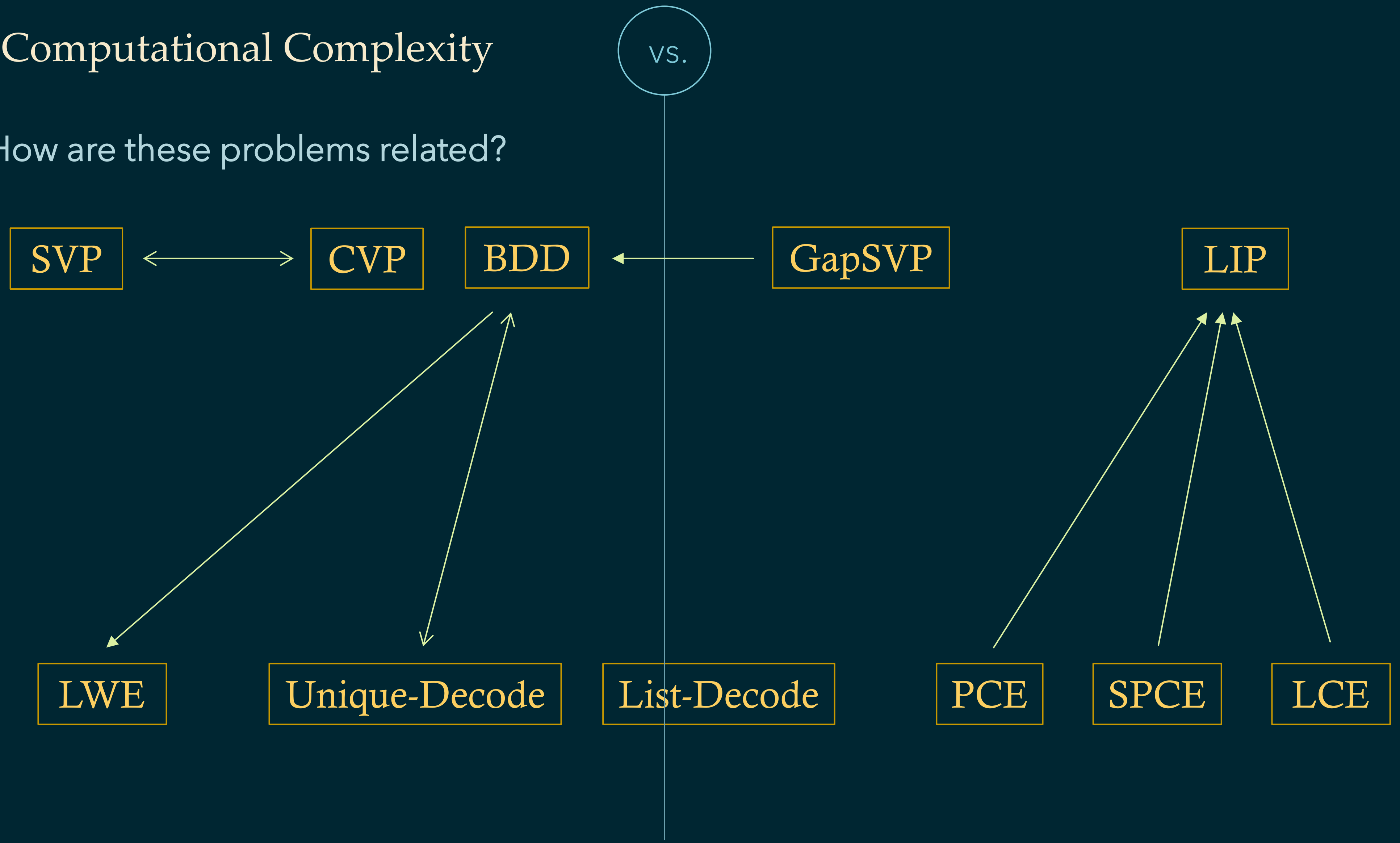
LWE — decode a random linear code

Unique-Decode, **List-Decode** — find the closest codeword(s)

PCE, **SPCE**, **LCE** — decide if two codes are equivalent

Computational Complexity

How are these problems related?



vs.

Constructions and Algorithms

SVP

CVP

BDD

GapSVP

LIP

How can these problems be solved?

How can we construct efficient algorithms to solve these?

LWE

Unique-Decode

List-Decode

PCE

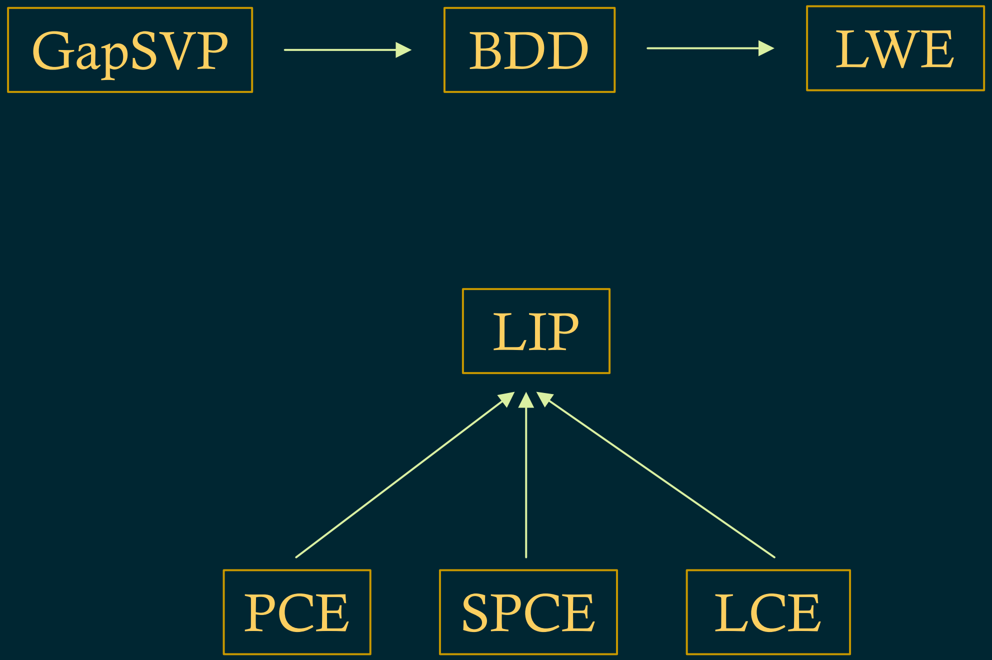
SPCE

LCE

Computational Complexity

vs.

How are these problems related?



Constructions and Algorithms

How can these problems be solved?

Unique-Decode

List-Decode

I. Computational Complexity

Fine-Grained Hardness of LWE

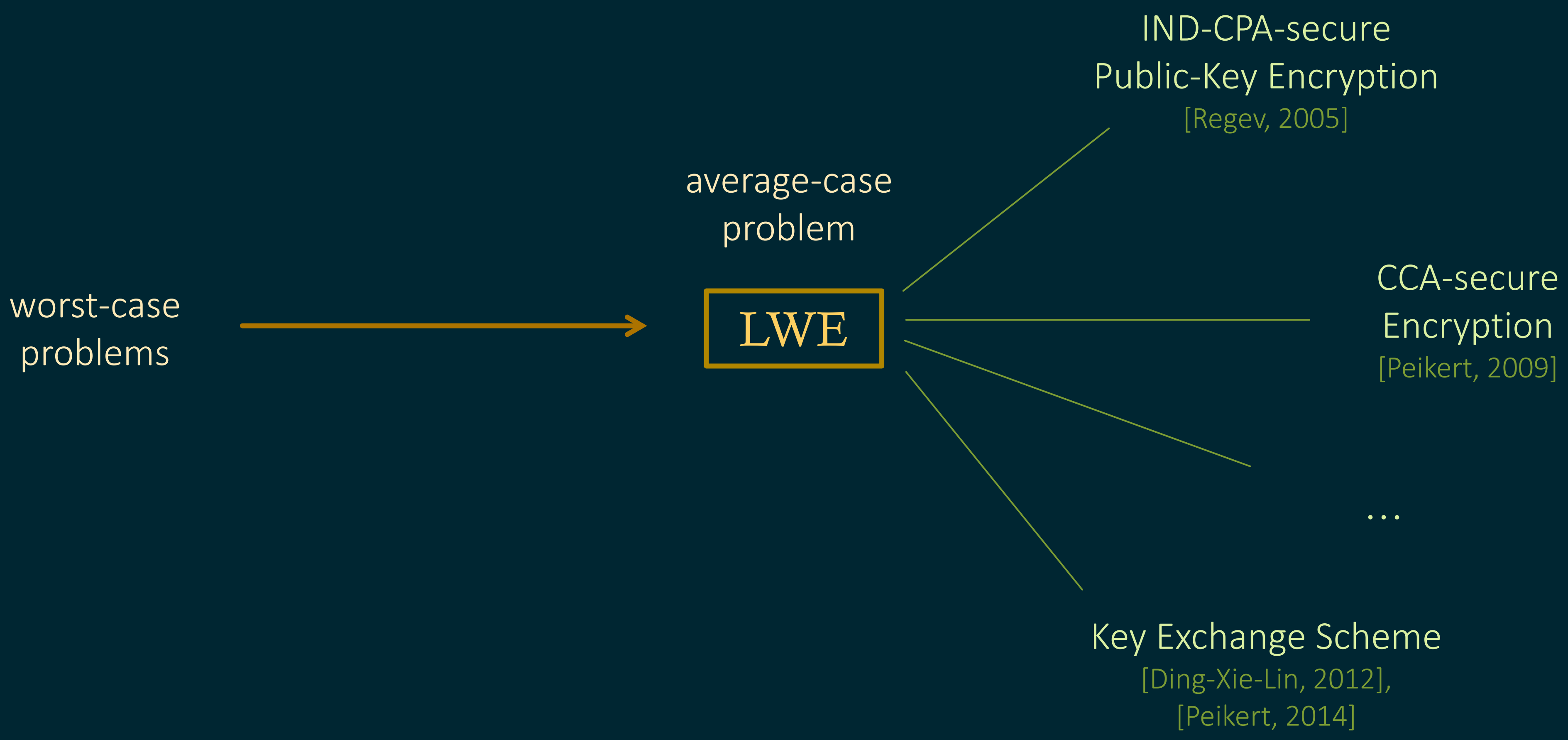
Based on joint work with Divesh Aggarwal and Leong Jin Ming

Cryptography from LWE

LWE

- IND-CPA-secure Public-Key Encryption [Regev, 2005]
- CCA-secure Encryption [Peikert, 2009]
- …
- Key Exchange Scheme [Ding-Xie-Lin, 2012], [Peikert, 2014]

Cryptographic Significance



Cryptographic Significance

worst-case hardness ⟹ average-case hardness ⟹ cryptosystems

GapSVP BDD ⟶ LWE

Cryptographic Significance

worst-case hardness ⟹ average-case hardness ⟹ **Post-quantum** cryptosystems

GapSVP BDD → LWE

no known efficient quantum algorithms for these problems

Learning With Errors

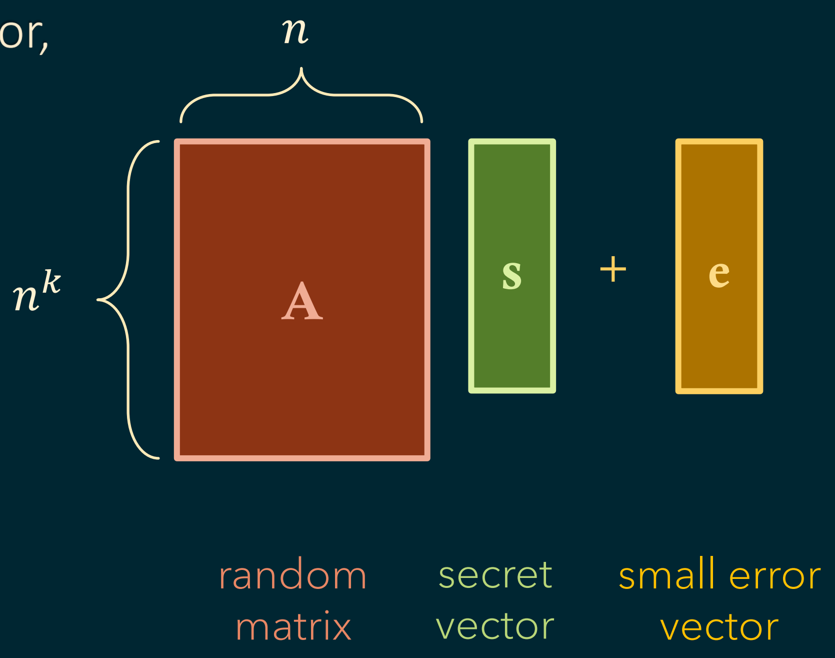
(search)

$\mathrm{LWE}_{n,p,\phi}$: n dimension, p modulus, $\phi \sim \mathbb{R}/\mathbb{Z}$ error distribution

Given noisy samples $(\mathbf{a}, \ \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where

$\mathbf{a} \leftarrow \mathbb{Z}_p^n$ uniformly random, $\mathbf{s} \in \mathbb{Z}_p^n$ unknown, $e \leftarrow \phi$ small error,

output $\mathbf{s}$.



Learning With Errors

(decision)

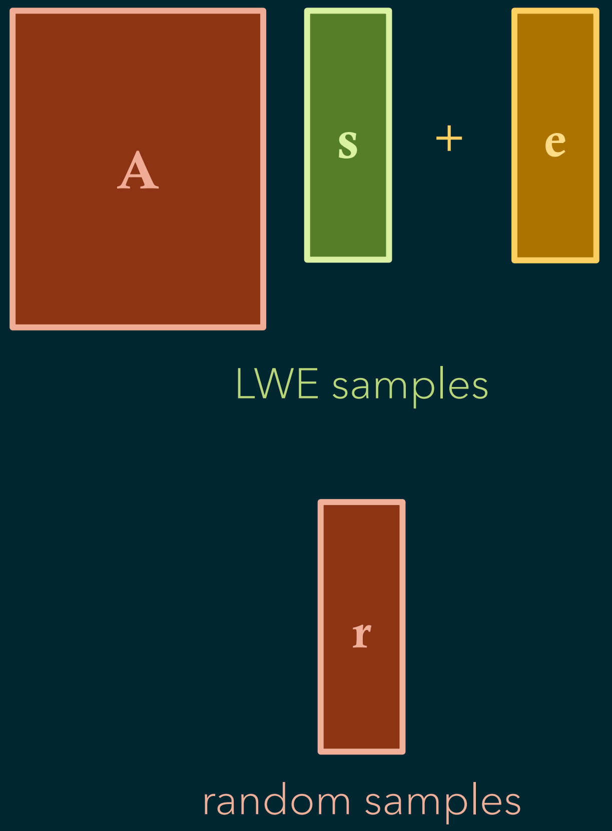
$\mathbf{LWE}_{n,p,\phi}$: n dimension, p modulus, $\phi \sim \mathbb{R}/\mathbb{Z}$ error distribution

Given noisy samples $(\mathbf{a}, b)$, where

$\mathbf{a} \leftarrow \mathbb{Z}_p^n$ uniformly random, $b \in \mathbb{Z}_p$,

output

- YES if samples are from the LWE distribution for $\mathbf{s}$ and ϕ,
- NO if samples are uniformly random.



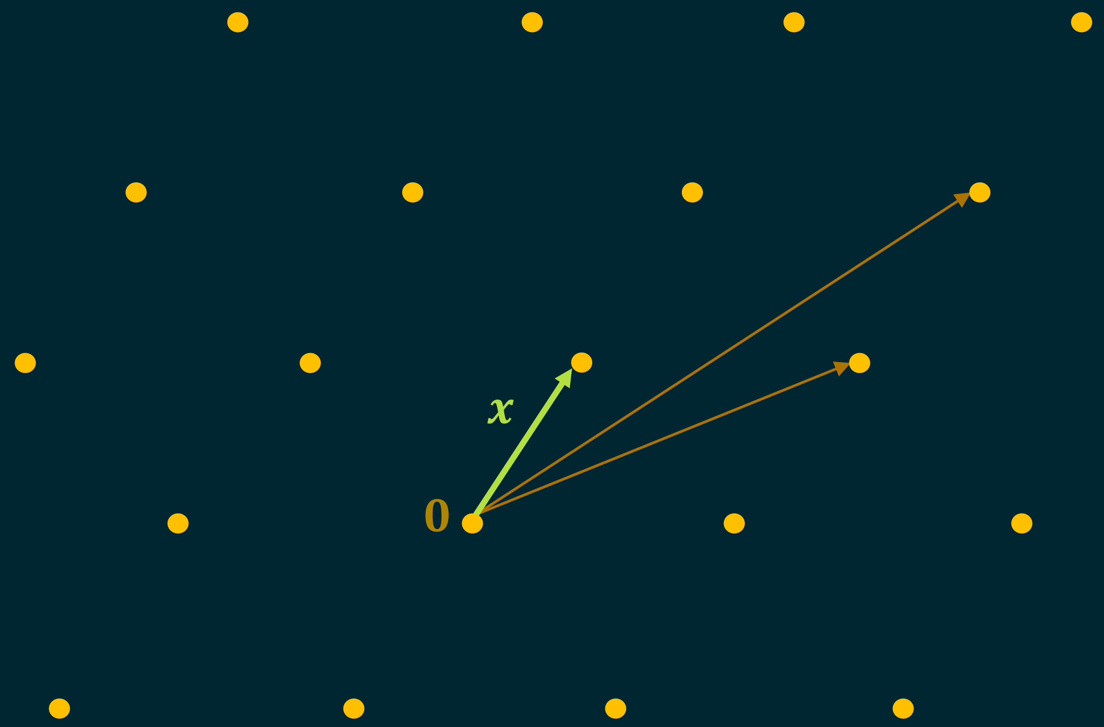
Shortest Vector Problem

SVP :

Given a basis $\mathcal{B}$ for lattice $\mathcal{L} \subset \mathbb{R}^n$,

find a shortest non-zero lattice vector $\boldsymbol{x}$,

i.e. $\boldsymbol{x} \in \mathcal{L} \setminus \{\mathbf{0}\}$, such that $\|\boldsymbol{x}\| = \lambda_1(\mathcal{L})$.



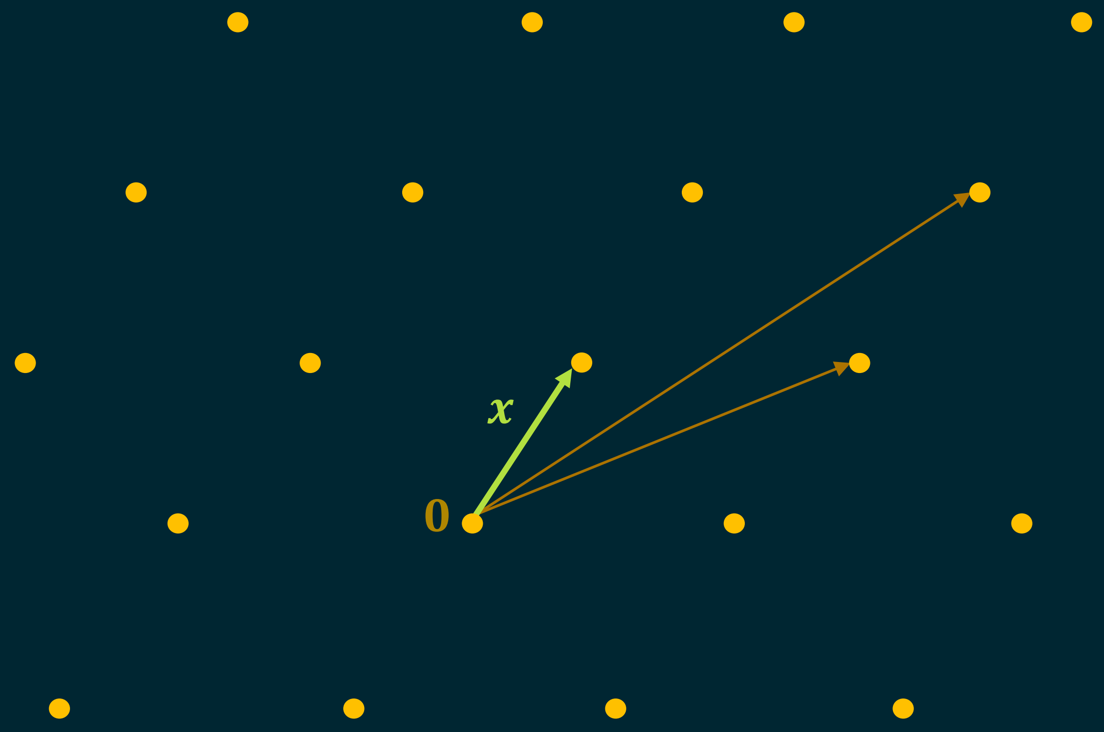
Shortest Vector Problem

SVP :

Given a basis $\mathcal{B}$ for lattice $\mathcal{L} \subset \mathbb{R}^n$,

find a shortest non-zero lattice vector $\boldsymbol{x}$,

i.e. $\boldsymbol{x} \in \mathcal{L} \setminus \{\mathbf{0}\}$, such that $\|\boldsymbol{x}\| = \lambda_1(\mathcal{L})$.



GapSVP$_\gamma$ is an approximate decision variant.

Approximate Shortest Vector Problem

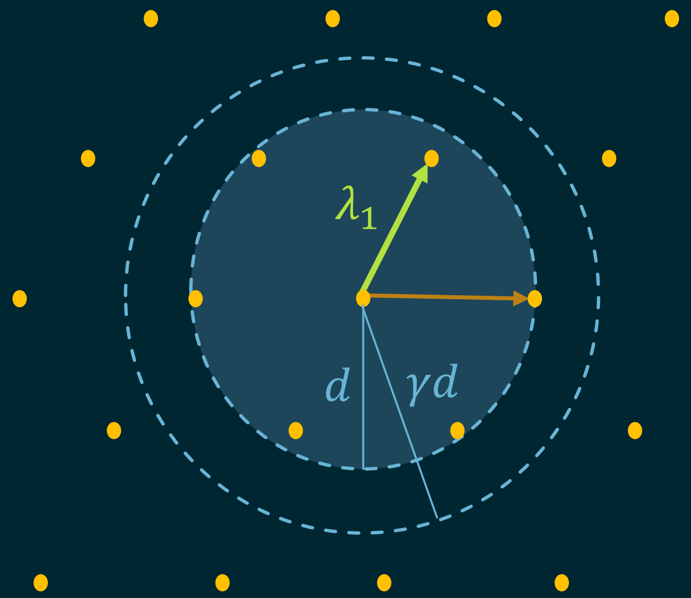
GapSVP_γ : $\gamma \geq 1$ approximation factor

Given a basis $\mathcal{B}$ for a full-rank lattice $\mathcal{L} \subset \mathbb{R}^n$

and a distance parameter $d > 0$,

output

- YES if $\lambda_1(\mathcal{L}) \leq d$
- NO if $\lambda_1(\mathcal{L}) \geq \gamma \cdot d$.



Closest Vector Problem

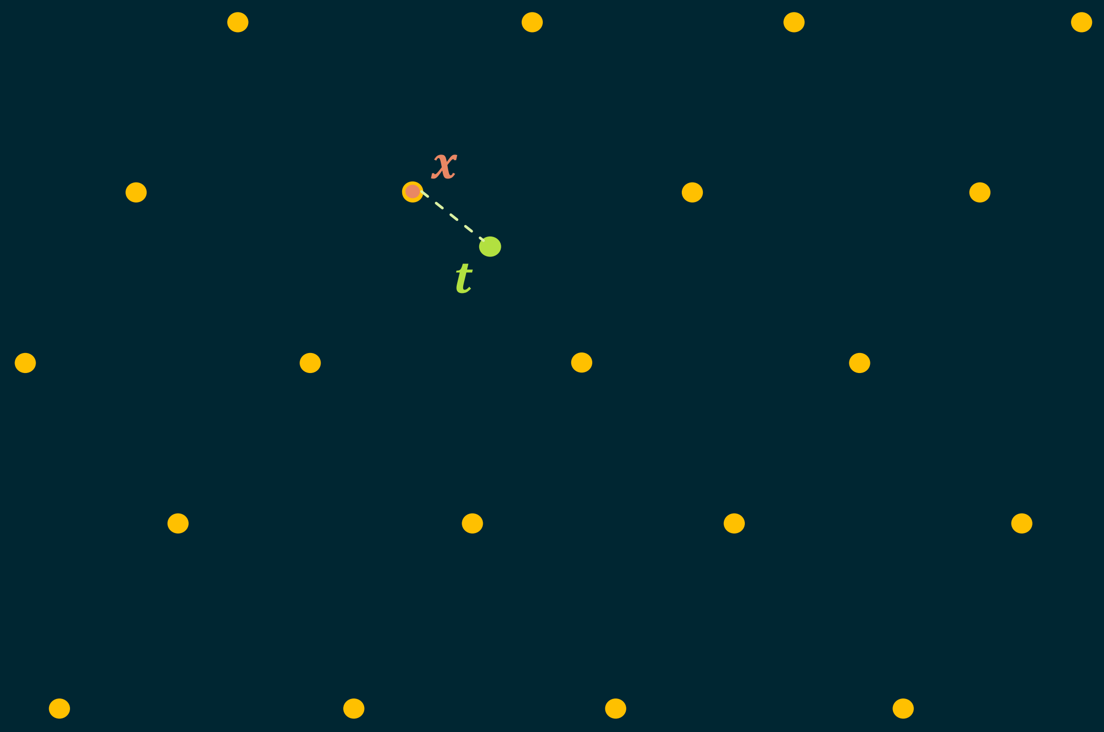
CVP :

Given a basis $\boldsymbol{\mathcal{B}}$ for lattice $\boldsymbol{\mathcal{L}} \subset \mathbb{R}^n$,

and a target vector $\boldsymbol{t} \in \mathbb{R}^{\boldsymbol{n}}$,

find a lattice vector $\boldsymbol{x}$ closest to $\boldsymbol{t}$,

i.e. $\boldsymbol{x} \in \boldsymbol{\mathcal{L}}$, such that $\|\boldsymbol{x} - \boldsymbol{t}\| = \mathrm{dist}(\boldsymbol{t}, \boldsymbol{\mathcal{L}})$.



Closest Vector Problem

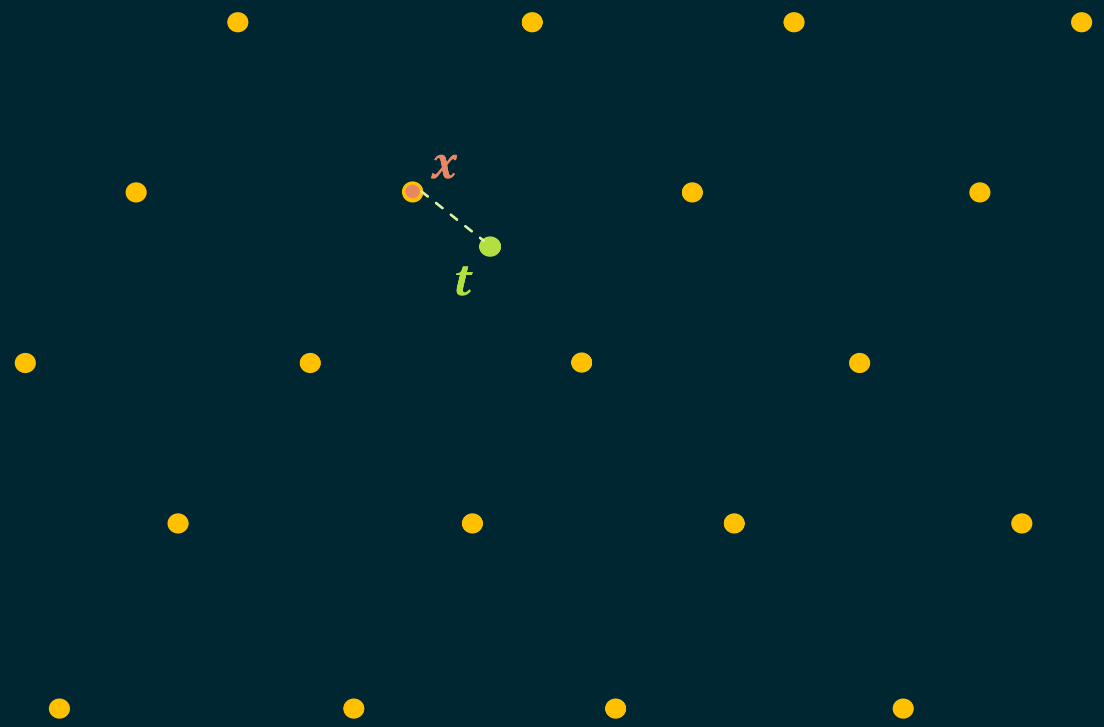
CVP :

Given a basis $\boldsymbol{\mathcal{B}}$ for lattice $\boldsymbol{\mathcal{L}} \subset \mathbb{R}^n$,

and a target vector $\boldsymbol{t} \in \mathbb{R}^n$,

find a lattice vector $\boldsymbol{x}$ closest to $\boldsymbol{t}$,

i.e. $\boldsymbol{x} \in \boldsymbol{\mathcal{L}}$, such that $||\boldsymbol{x} - \boldsymbol{t}|| = \mathrm{dist}(\boldsymbol{t}, \boldsymbol{\mathcal{L}})$.



BDD_α is an approximate variant.

Bounded Distance Decoding

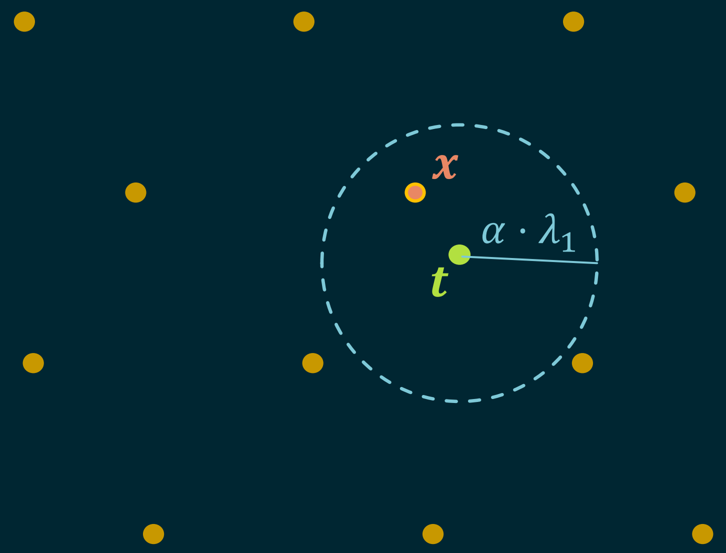
BDD_α : $\alpha > 0$ distance approximation factor

Given a basis $\mathcal{B}$ for a full-rank lattice $\mathcal{L} \subset \mathbb{R}^n$

and a target vector $\boldsymbol{t} \in \mathbb{R}^n$ close to the lattice,

find a lattice vector $\boldsymbol{x} \in \mathcal{L}$ closest to $\boldsymbol{t}$,

i.e. $\boldsymbol{x} \in \mathcal{L}$, such that $\|\boldsymbol{x} - \boldsymbol{t}\| < \alpha \cdot \lambda_1(\mathcal{L})$.



Bounded Distance Decoding

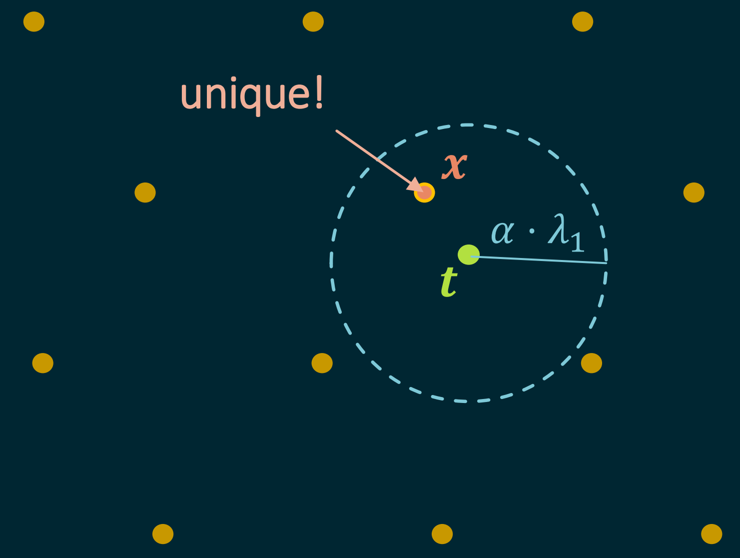
BDD$_\alpha$: $\alpha < \frac{1}{2}$ distance approximation factor

Given a basis $\mathcal{B}$ for a full-rank lattice $\mathcal{L} \subset \mathbb{R}^n$

and a target vector $t \in \mathbb{R}^n$ close to the lattice,

find *the unique* lattice vector $x \in \mathcal{L}$ closest to t,

i.e. $x \in \mathcal{L}$, such that $\|x - t\| < \alpha \cdot \lambda_1(\mathcal{L})$.



Hardness of LWE

[Regev, 2009] – quantum reduction from worst-case lattice problems to decision-LWE

Hardness of LWE

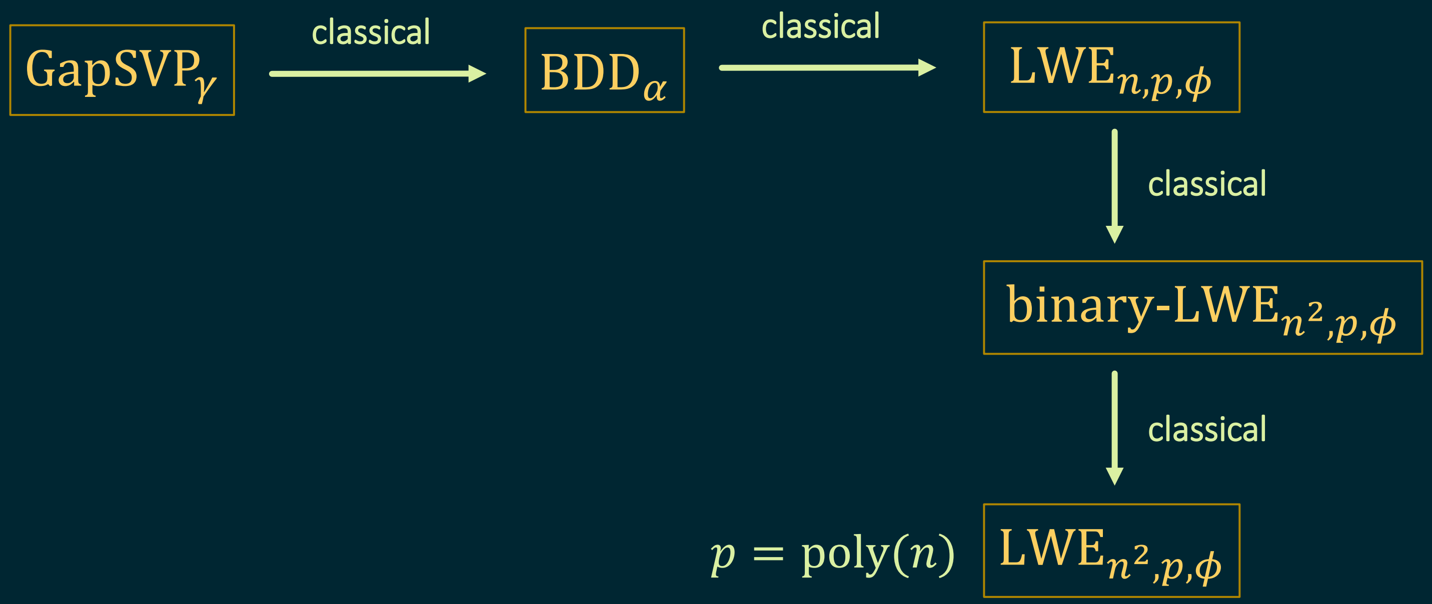
[Peikert, 2009] – classical reduction, but modulus becomes exponential

$\text{GapSVP}_\gamma \xrightarrow{\text{classical}} \text{BDD}_\alpha \xrightarrow{\text{classical}} \text{LWE}_{n,p,\phi}$

$p = \exp(n)$

Hardness of LWE

[Brakerski, Peikert, Langlois, Regev, Stehle, 2013] – classical reduction with polynomial modulus



Hardness of LWE

Algorithms for Lattice Problems

$\text{GapSVP}_\gamma \rightarrow \text{BDD}_\alpha \rightarrow \text{LWE}_{n,p,\phi} \rightarrow \text{binary-LWE}_{n^2,p,\phi} \rightarrow \text{LWE}_{n^2,p,\phi}$

Fastest algorithms for these problems run in $2^{\Theta(n)}$ time (for polynomial approximation factor).

What the Reduction says about LWE Algorithms

$\mathrm{GapSVP}_\gamma \rightarrow \mathrm{BDD}_\alpha \rightarrow \mathrm{LWE}_{n,p,\phi} \rightarrow \text{binary-}\mathrm{LWE}_{n^2,p,\phi} \rightarrow \mathrm{LWE}_{n^2,p,\phi}$

Conjecture: known algorithms are the best possible

What the Reduction says about LWE Algorithms

$$\mathrm{GapSVP}_\gamma \longrightarrow \mathrm{BDD}_\alpha \longrightarrow \mathrm{LWE}_{n,p,\phi} \longrightarrow \text{binary-}\mathrm{LWE}_{n^2,p,\phi} \longrightarrow \mathrm{LWE}_{n^2,p,\phi}$$

Conjecture: known algorithms are the best possible $\Longrightarrow$ solving LWE requires at least $2^{\Omega(\sqrt{n})}$ time

What the Reduction says about LWE Algorithms

[Blum-Kalai-Wasserman, 2000] — Best known algorithm for $\mathrm{LWE}_{n,p,\phi}$ runs in $2^{O\left(\frac{n}{\log n}\cdot\log p\right)}$ time.

GapSVP_γ → BDD_α → $\mathrm{LWE}_{n,p,\phi}$ → binary-$\mathrm{LWE}_{n^2,p,\phi}$ → $\mathrm{LWE}_{n^2,p,\phi}$

Conjecture: known algorithms are the best possible $\Longrightarrow$ solving LWE requires at least $2^{\Omega(\sqrt{n})}$ time

What the Reduction says about LWE Algorithms

[Blum-Kalai-Wasserman, 2000] — Best known algorithm for $\mathrm{LWE}_{n,p,\phi}$ runs in $2^{O\left(\frac{n}{\log n}\cdot \log p\right)}$ time.

GapSVP_{γ} → BDD_{α} → $\mathrm{LWE}_{n,p,\phi}$ → $\text{binary-LWE}_{n^2,p,\phi}$ → $\mathrm{LWE}_{n^2,p,\phi}$

Big Gap!

Conjecture: known algorithms are the best possible $\Longrightarrow$ solving LWE requires at least $2^{\Omega(\sqrt{n})}$ time

Our Contribution

We close this gap by changing our perspective!

Security in Practice

What does it mean for a cryptosystem to be 256-bit secure?

Security in Practice

What does it mean for a cryptosystem to be 256-bit secure?

(a) The fastest algorithm for breaking the cryptosystem runs in 2^{256} time.

(b) No reasonably efficient algorithm can break the cryptosystem with probability $> 2^{-256}$.

Security in Practice

What does it mean for a cryptosystem to be 256-bit secure?

(a) The fastest algorithm for breaking the cryptosystem runs in 2^{256} time.

(b) No reasonably efficient algorithm can break the cryptosystem with probability $> 2^{-256}$.

This is what we usually want for cryptographic security

An Alternative Perspective

An alternative measure of computational hardness:

The maximum success probability of any probabilistic polynomial-time algorithm that finds a solution.

An Alternative Perspective

An alternative measure of computational hardness:

The maximum success probability of any probabilistic polynomial-time algorithm

that finds a solution.

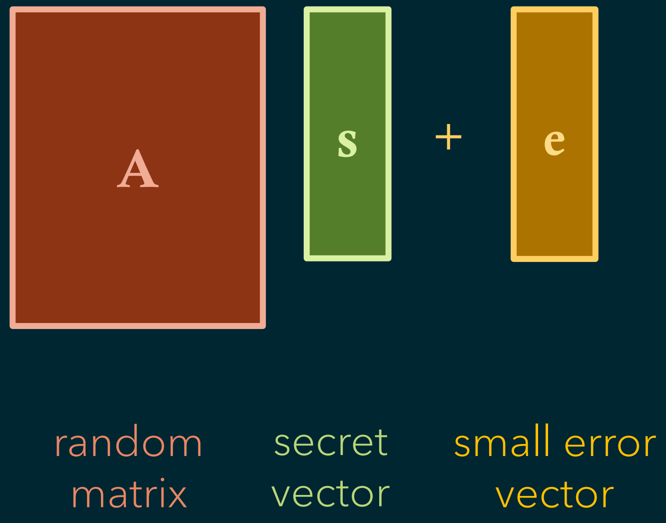
We study worst-case to average-case hardness of LWE under this framework.

Success Probability of Solving LWE

Trivial algorithm (guess the error): Success probability for solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n)}$.

Success Probability of Solving LWE

Trivial algorithm (guess the error): Success probability for solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n)}$.



Success Probability of Solving LWE

Trivial algorithm (guess the error): Success probability for solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n)}$.

All other algorithms are not efficient, so it is unlikely that we can achieve better than this.

Success Probability of Solving Lattice Problems

LLL / Slide Reduction + guess coefficients: Success probability of solving GapSVP_γ is $2^{-\Theta(n^2/\log n)}$.

Success Probability of Solving Lattice Problems

LLL / Slide Reduction + guess coefficients: Success probability of solving GapSVP_γ is $2^{-\Theta(n^2/\log n)}$.

Known techniques do not seem to improve this when restricted to efficient algorithms,

so it is unlikely that we can achieve much better than this.

Success Probability of Solving Lattice Problems

LLL / Slide Reduction + guess coefficients: Success probability of solving GapSVP_γ is $2^{-\Theta(n^2/\log n)}$.

When restricted to efficient algorithms, known techniques do not seem to improve this,

so it is unlikely that we can achieve much better than this.

BDD_α is closely related to GapSVP_γ for $\gamma = \mathrm{poly}(n) = 1/\alpha$,

so it is unlikely we can achieve better than known algorithms.

A Natural Conjecture

Conjecture: *(informal)* No algorithm can solve BDD_α on an arbitrary n-rank lattice for $\alpha = 1/\mathrm{poly}(n)$ in polynomial time with success probability better than $2^{-n^2/\log n}$.

What We Show

Trivial algorithm: Success probability for efficiently solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n)}$.

Conjecture $\Longrightarrow$ Maximum success probability for efficiently solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n/\log^2 n)}$.

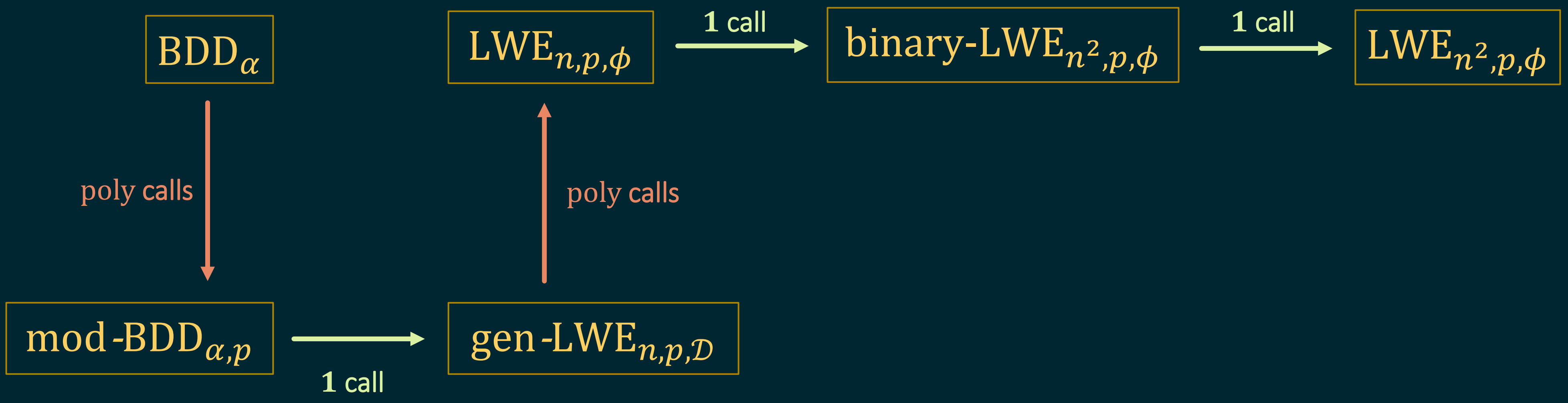
What We Show

Trivial algorithm: Success probability for efficiently solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n)}$.

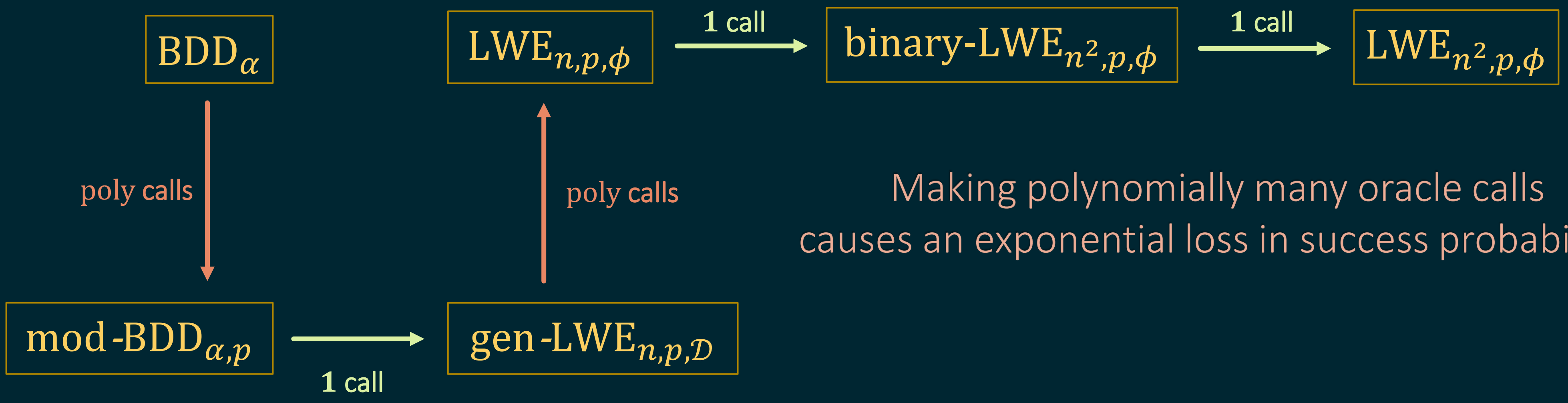
Tight!

Conjecture $\implies$ Maximum success probability for efficiently solving $\mathrm{LWE}_{n,p,\phi}$ is $p^{-\Omega(n/\log^2 n)}$.

Limitations of the Original Reduction



Limitations of the Original Reduction



Making polynomially many oracle calls
causes an exponential loss in success probability!

Limitations of the Original Reduction

Reduction algorithm for $\mathcal{P} \to \mathcal{Q}$ makes k calls to oracle for $\mathcal{Q}$.

Success probability of solving $\mathcal{Q}$ is $\geq \epsilon \implies$ success probability of solving $\mathcal{P}$ is $\geq \epsilon^k$.

Limitations of the Original Reduction

Reduction algorithm for $\mathcal{P} \to \mathcal{Q}$ makes k calls to oracle for $\mathcal{Q}$.

Success probability of solving $\mathcal{Q}$ is $\geq \epsilon \implies$ success probability of solving $\mathcal{P}$ is $\geq \epsilon^k$.

Success probability of solving $\mathcal{P}$ is $\leq \delta \implies$ success probability of solving $\mathcal{Q}$ is $\leq \delta^{1/k}$.

Limitations of the Original Reduction

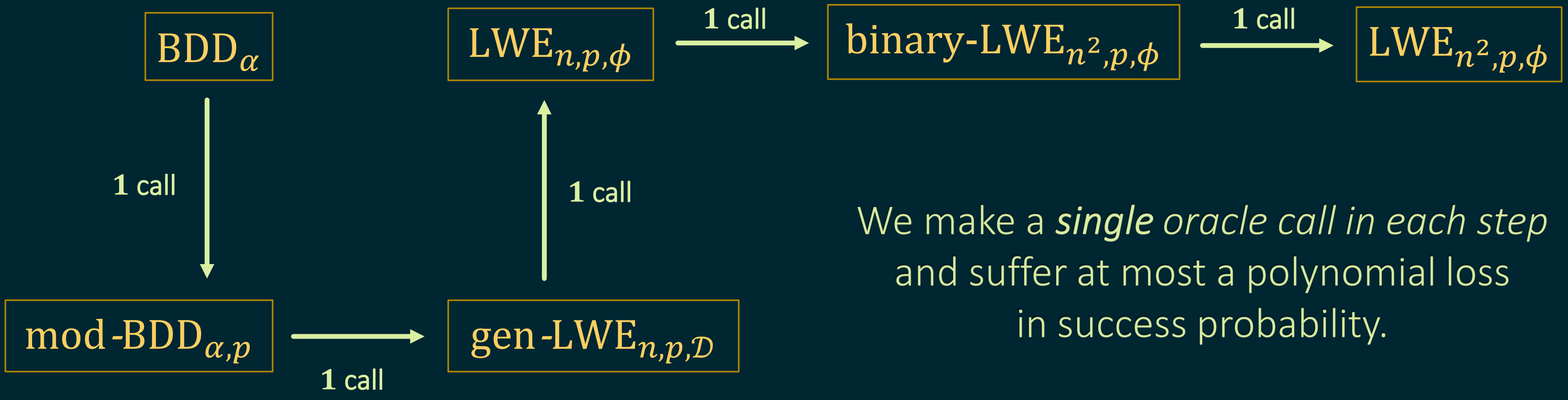
Reduction algorithm for $\mathcal{P} \to \mathcal{Q}$ makes k calls to oracle for $\mathcal{Q}$.

Success probability of solving $\mathcal{Q}$ is $\geq \epsilon \implies$ success probability of solving $\mathcal{P}$ is $\geq \epsilon^k$.

Success probability of solving $\mathcal{P}$ is $\leq \delta \implies$ success probability of solving $\mathcal{Q}$ is $\leq \delta^{1/k}$.

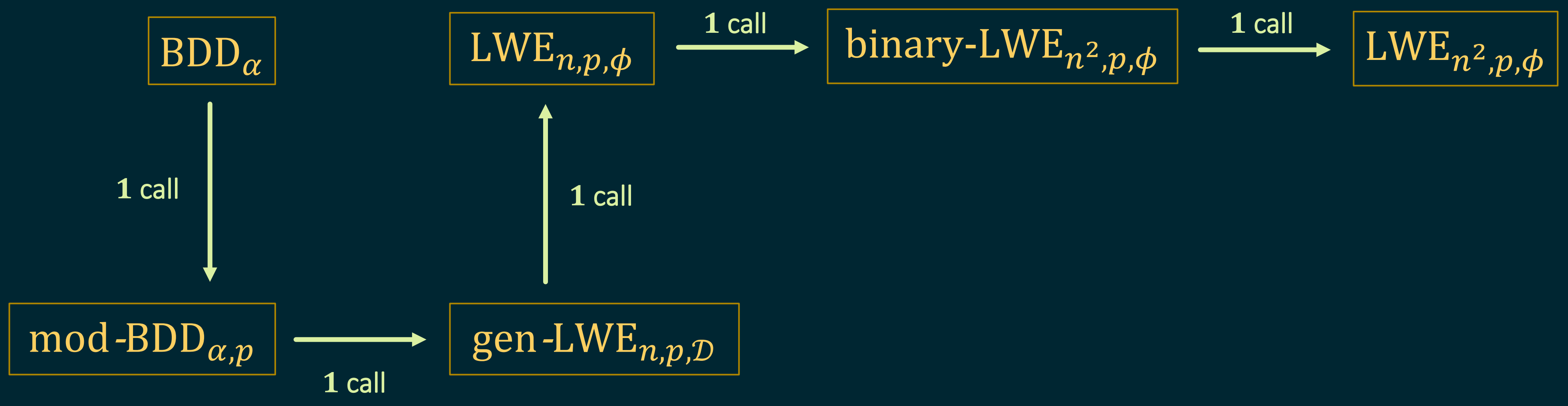
We want just $O(1)$ oracle calls to get a meaningful conclusion.

Our Reduction



We make a ***single** oracle call in each step*
and suffer at most a polynomial loss
in success probability.

Our Reduction



We use the same techniques as [Regev, 2005] and [Brakerski+, 2013],
but with great care to the *explicit loss in success probability* and *number of oracle calls*.

Our Main Result

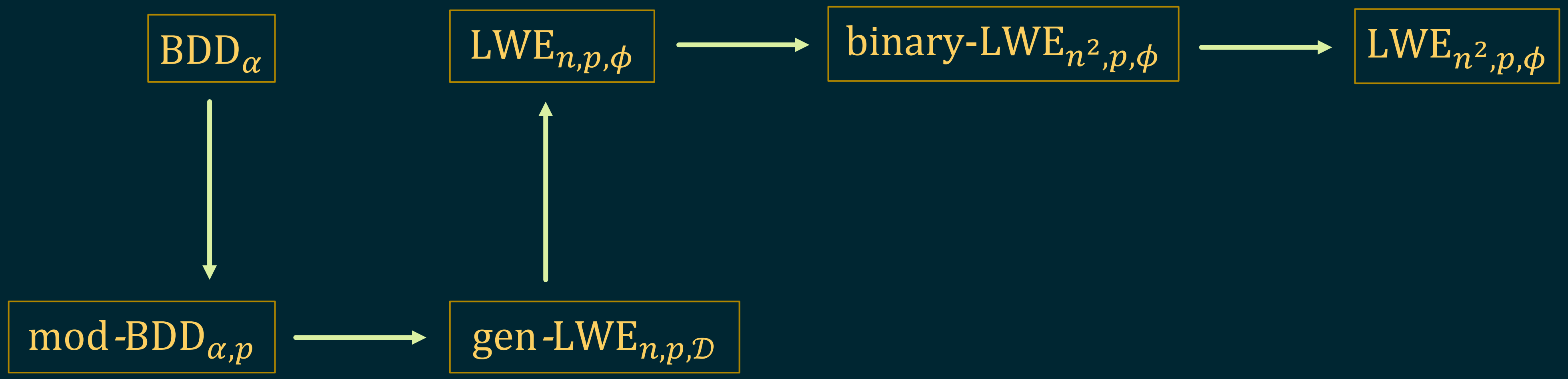
Theorem 1: *(informal)* If no efficient algorithm can solve BDD_α for $\alpha < \frac{1}{2}$

with success probability greater than $2^{-\Omega(n^2/\log n)}$,

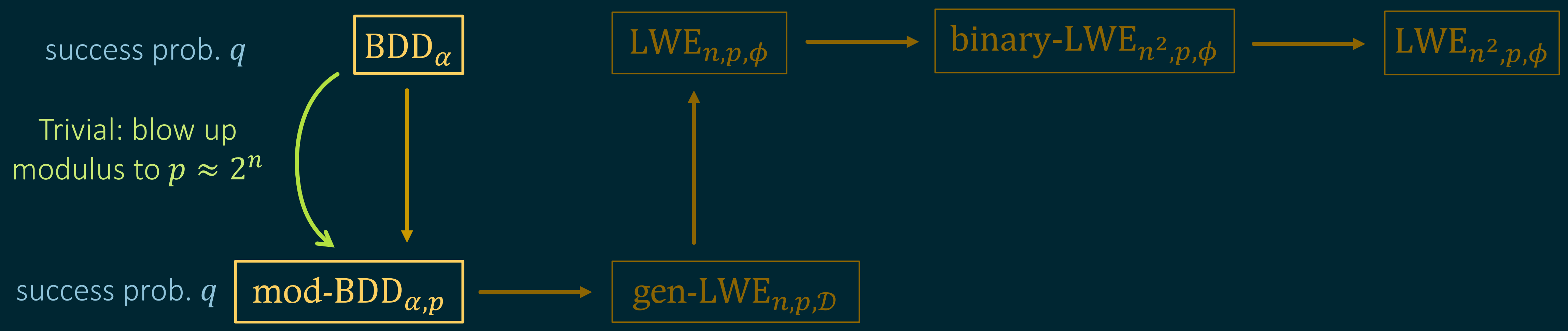
then no efficient algorithm can solve search-$\mathrm{LWE}_{n,p,\phi}$ (even for binary secret)

for dimension n, and modulus $p = \mathrm{poly}(n)$ with success probability $2^{-n/\log n}$.

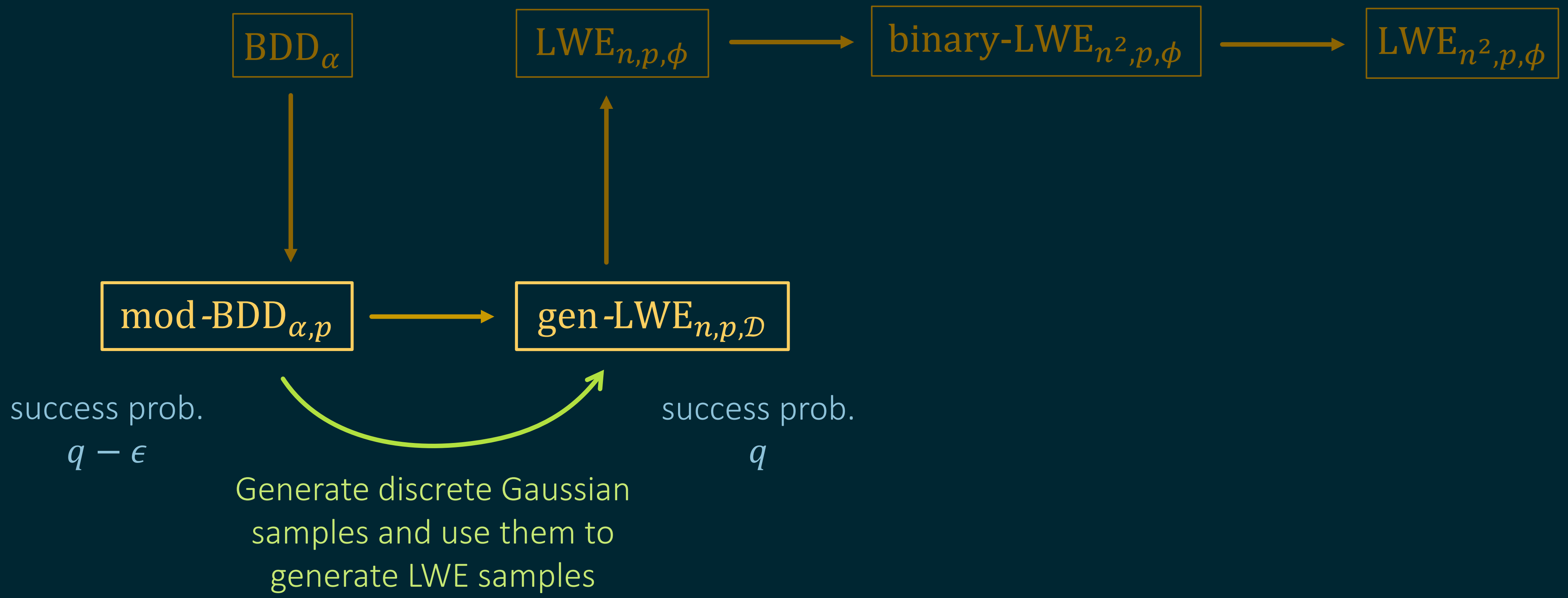
Our Reduction



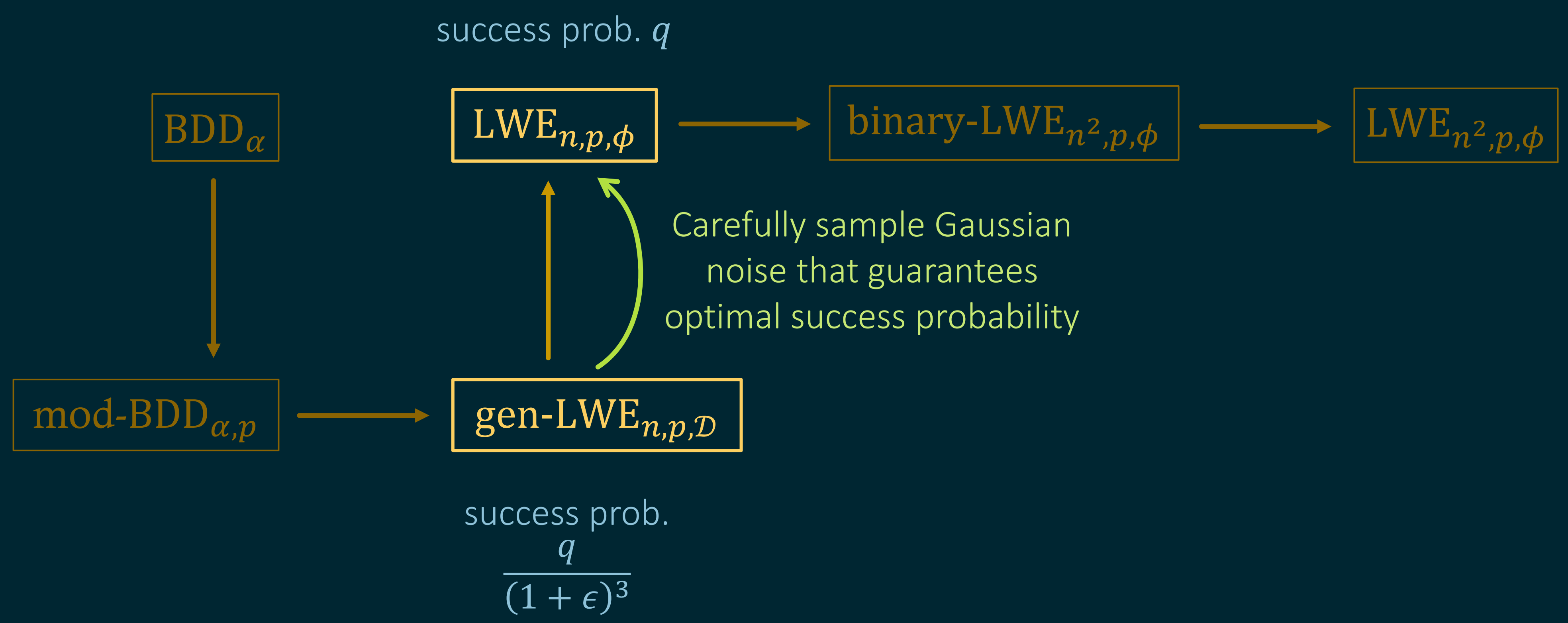
Our Proof Techniques



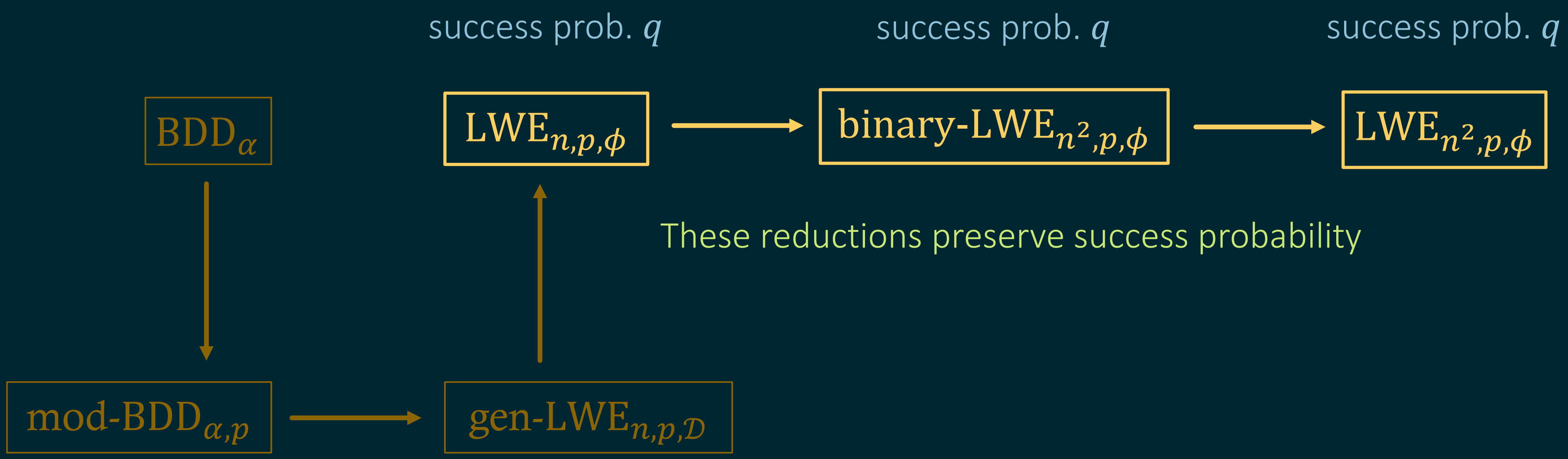
Our Proof Techniques



Our Proof Techniques



Our Proof Techniques



Our Second Result

Theorem 2: *(informal)* If no algorithm can solve search-$\mathrm{LWE}_{n,p}$ for polynomial modulus

with success probability $\boldsymbol{\alpha}$ in *expected* polynomial time,

then no efficient algorithm can "solve" decision-$\mathrm{LWE}_{n,p}$

with success probability $\approx \boldsymbol{\alpha}$.

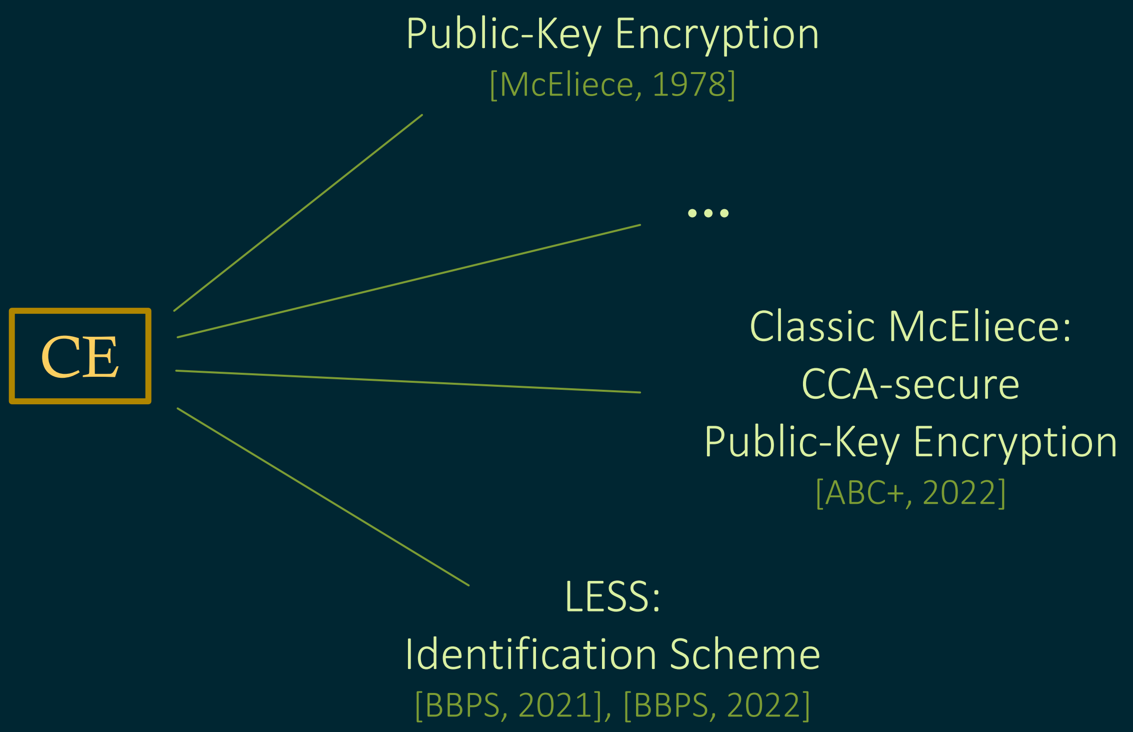
Open Directions

- Reductions $\mathrm{BDD} \to$ search-LWE and search-$\mathrm{LWE} \to$ decision-LWE are disconnected, because *expected* polynomial-time is a fundamental part of the second reduction. Is a workaround possible?
- Establish a similar result for $\mathrm{GapSVP} \to \mathrm{BDD}$ (or prove impossibility).
- Use this alternative framework to study the complexity of other computational problems relevant to cryptography or learning.

Reductions Between Code Equivalence Problems

Based on joint work with Mahdi Cheraghchi and Nikhil Shagrithaya

Cryptographic Significance



Code Equivalence Problem

CE : Given two codes $\boldsymbol{\mathcal{C}_1}, \boldsymbol{\mathcal{C}_2} \subseteq \mathbb{F}_q^n$, decide whether $\boldsymbol{\mathcal{C}_1}$ and $\boldsymbol{\mathcal{C}_2}$ are equivalent.

Code Equivalence Problem

CE : Given two codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$, decide whether $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent.

ex: **PCE** Permutation CE

SPCE Signed Permutation CE

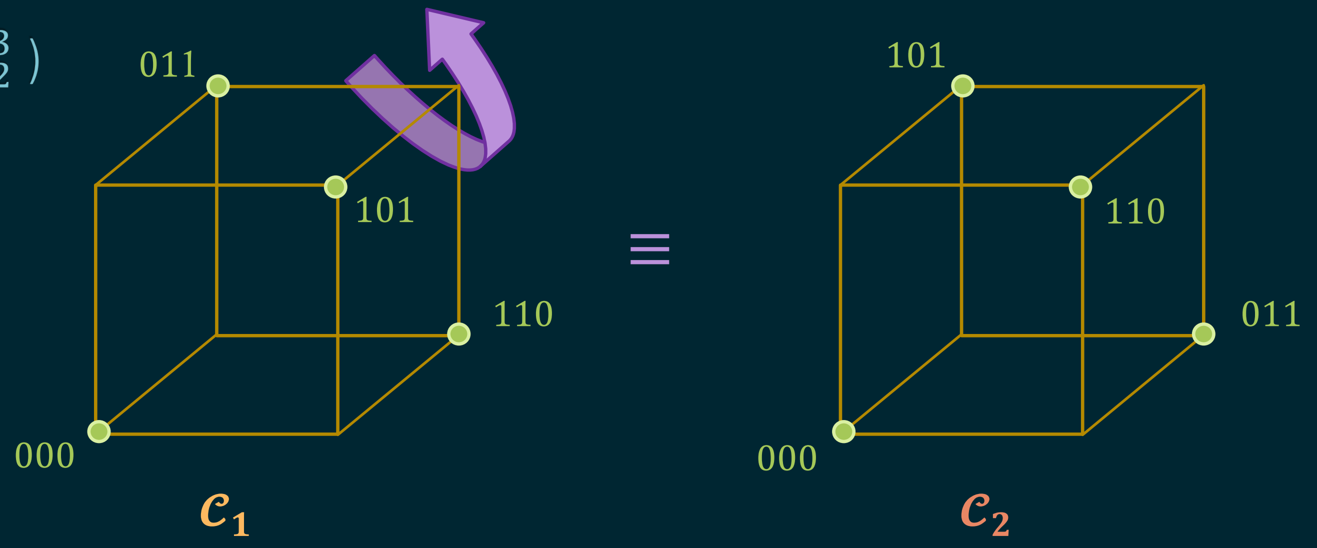
LCE Linear CE

Permutation Code Equivalence

PCE : Given generator matrices $\boldsymbol{G_1}, \boldsymbol{G_2} \in \mathbb{F}_q^{k\times n}$ for codes $\boldsymbol{\mathcal{C}_1}, \boldsymbol{\mathcal{C}_2} \subseteq \mathbb{F}_q^n$,

decide if $\boldsymbol{\mathcal{C}_1}$ and $\boldsymbol{\mathcal{C}_2}$ are the same up to permutation of coordinates.

ex: (for $\mathbb{F}_2^3$)

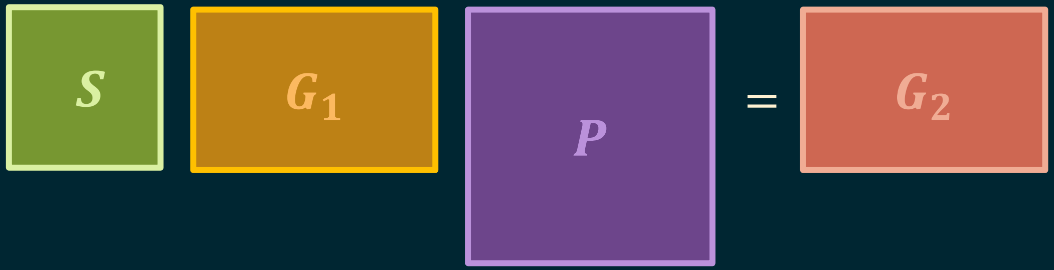


Permutation Code Equivalence

PCE : Given generator matrices $\boldsymbol{G_1}, \boldsymbol{G_2} \in \mathbb{F}_q^{k \times n}$ for codes $\boldsymbol{\mathcal{C}_1}, \boldsymbol{\mathcal{C}_2} \subseteq \mathbb{F}_q^n$,

output

- YES if there exists invertible $\mathbf{S} \in GL_k$ and *permutation* $\mathbf{P} \in \mathcal{P}_n$ such that $\mathbf{S}\mathbf{G_1}\mathbf{P} = \mathbf{G_2}$
- NO if otherwise.



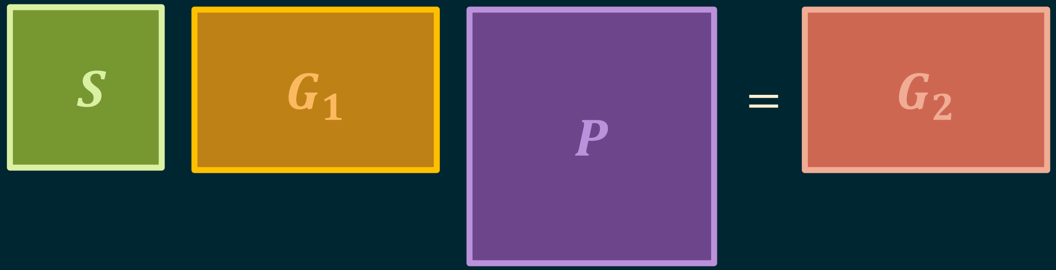
[Biasse-Micheli, 2023] Efficient search-to-decision reduction for PCE.

Signed Permutation Code Equivalence

SPCE : Given generator matrices $\boldsymbol{G_1}, \boldsymbol{G_2} \in \mathbb{F}_q^{k\times n}$ for codes $\boldsymbol{\mathcal{C}_1}, \boldsymbol{\mathcal{C}_2} \subseteq \mathbb{F}_q^n$,

output

- YES if there exists invertible $\mathbf{S} \in GL_k$ and *signed permutation* $\mathbf{P} \in S\mathcal{P}_n$ such that $\mathbf{SG_1P} = \mathbf{G_2}$
- NO if otherwise.

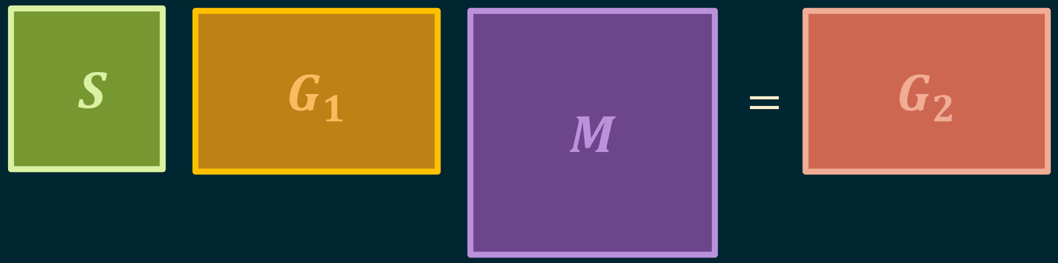


Linear Code Equivalence

LCE : Given generator matrices $\boldsymbol{G_1}, \boldsymbol{G_2} \in \mathbb{F}_q^{k \times n}$ for codes $\boldsymbol{\mathcal{C}_1}, \boldsymbol{\mathcal{C}_2} \subseteq \mathbb{F}_q^n$,

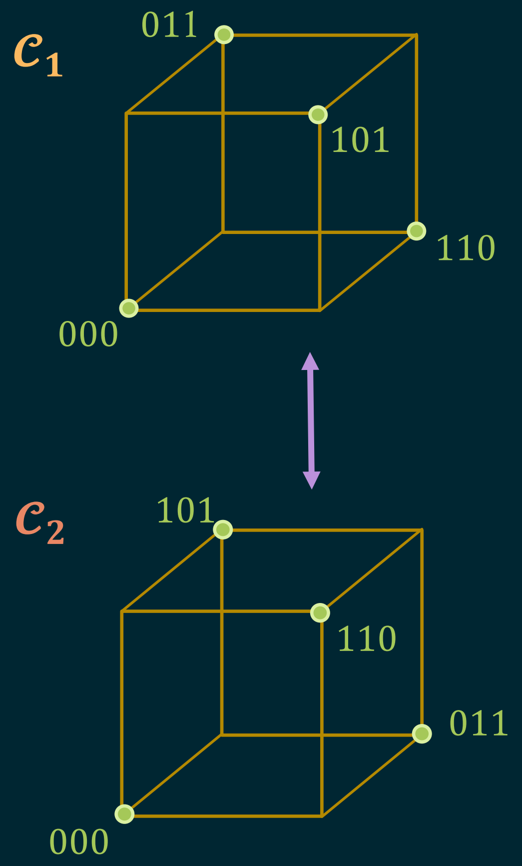
output

- YES if there exists invertible $\mathbf{S} \in GL_k$ and *monomial* $\mathbf{M} \in \mathcal{M}_n$ such that $\mathbf{S}\mathbf{G_1}\mathbf{M} = \mathbf{G_2}$
- NO if otherwise.

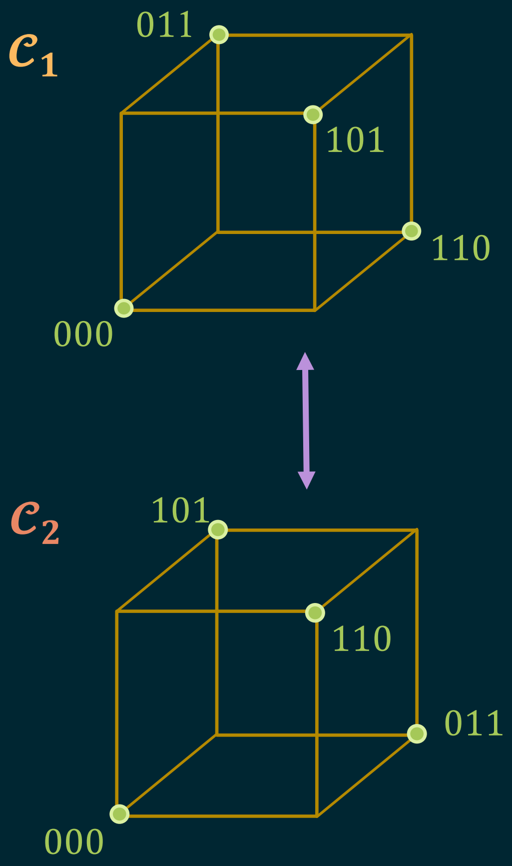


[Biasse-Micheli, 2023] Efficient search-to-decision reduction for PCE.

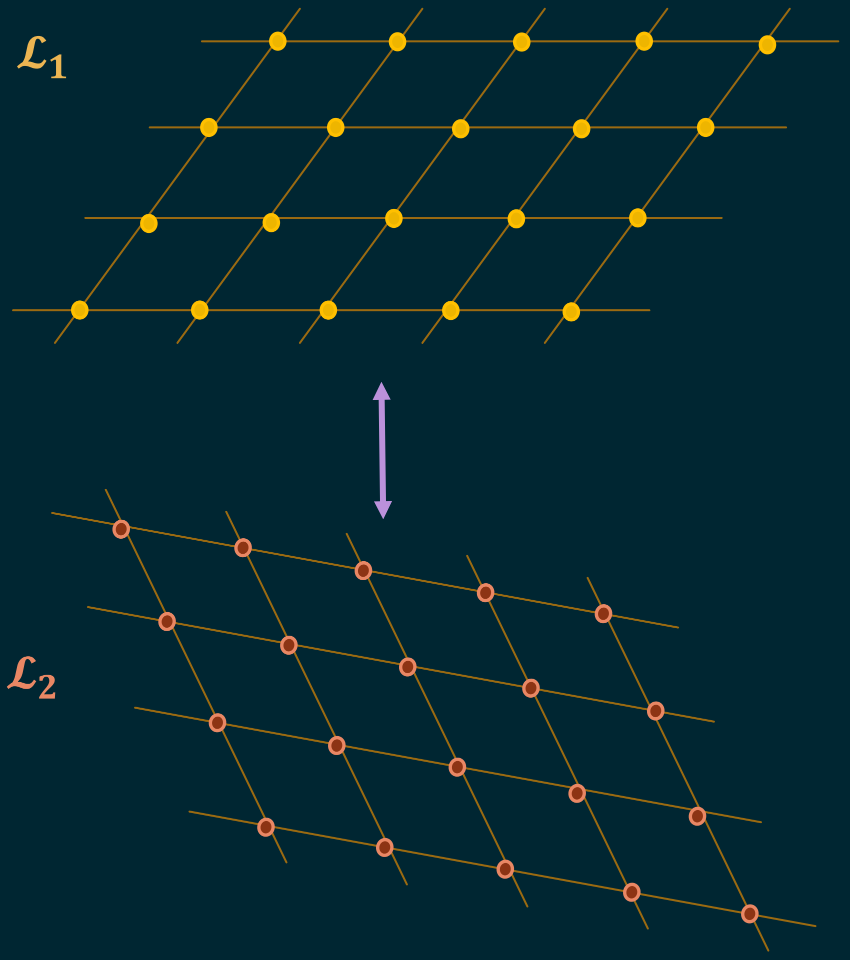
Code Equivalence



Code Equivalence



Lattice Isomorphism

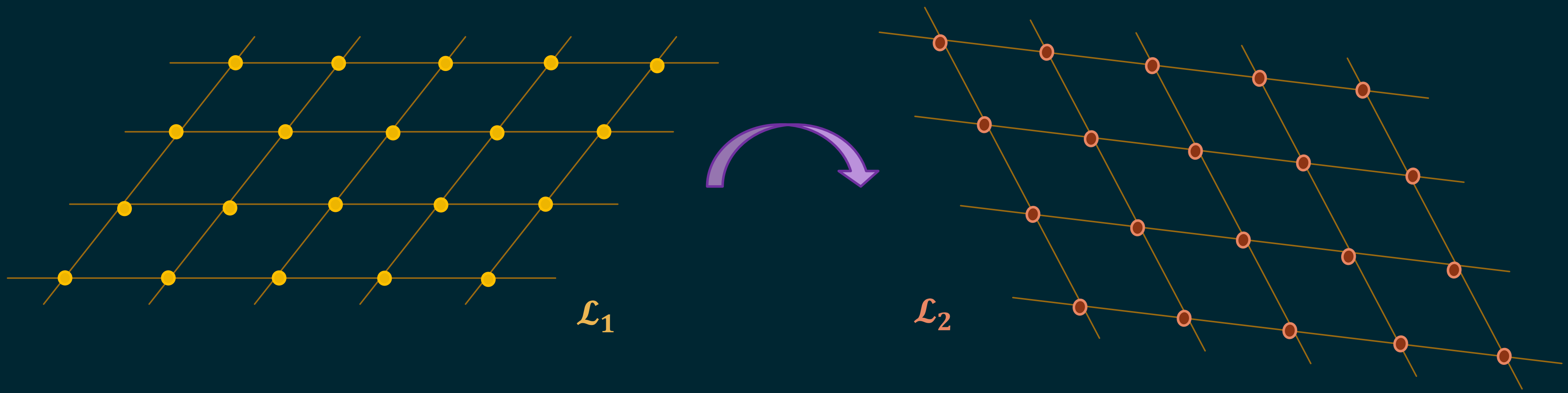


Lattice Isomorphism Problem

LIP : Given basis matrices $\boldsymbol{B_1}, \boldsymbol{B_2} \in \mathbb{R}^{k \times n}$ for lattices $\boldsymbol{\mathcal{L}_1}, \boldsymbol{\mathcal{L}_2} \subset \mathbb{R}^n$,

decide if $\boldsymbol{\mathcal{L}_1}$ and $\boldsymbol{\mathcal{L}_2}$ are the same lattice under some orthogonal transformation.

ex: (for $\mathbb{R}^2$)

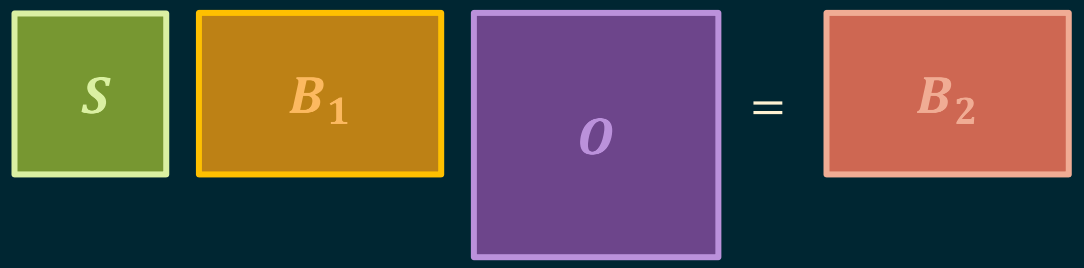


Lattice Isomorphism Problem

LIP : Given basis matrices $\boldsymbol{B_1}, \boldsymbol{B_2} \in \mathbb{R}^{k \times n}$ for lattices $\mathcal{L}_1, \mathcal{L}_2 \subset \mathbb{R}^n$,

output

- YES if there exists invertible $\mathbf{S} \in GL_k$ and *orthogonal* $\mathbf{O} \in \mathcal{O}_n$ such that $\mathbf{S}\,\mathbf{B_1}\mathbf{O} = \mathbf{B_2}$
- NO if otherwise.



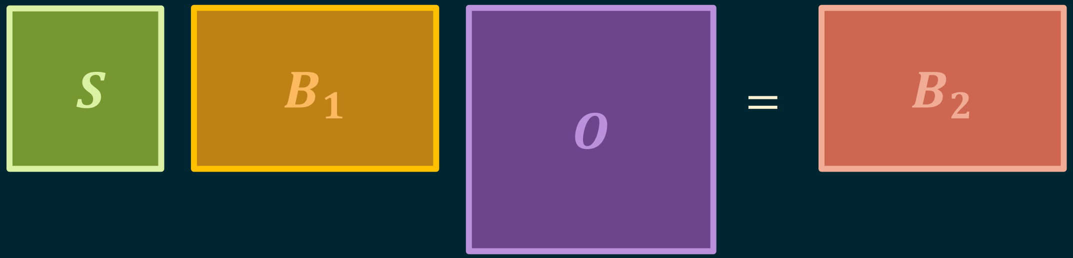
Lattice Isomorphism Problem

LIP : Given basis matrices $\boldsymbol{B_1}, \boldsymbol{B_2} \in \mathbb{R}^{k \times n}$ for lattices $\boldsymbol{\mathcal{L}_1}, \boldsymbol{\mathcal{L}_2} \subset \mathbb{R}^n$,

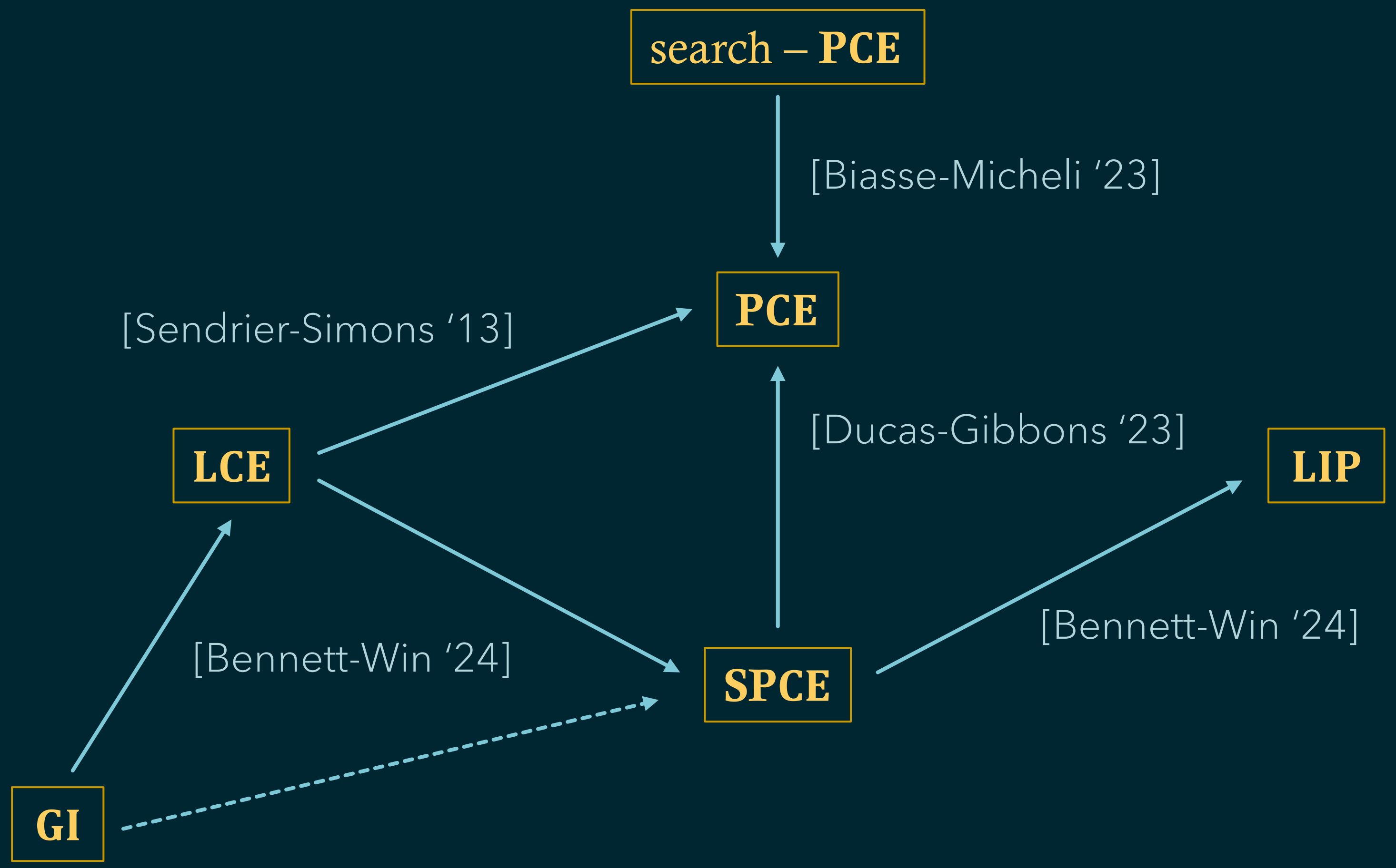
row span

output

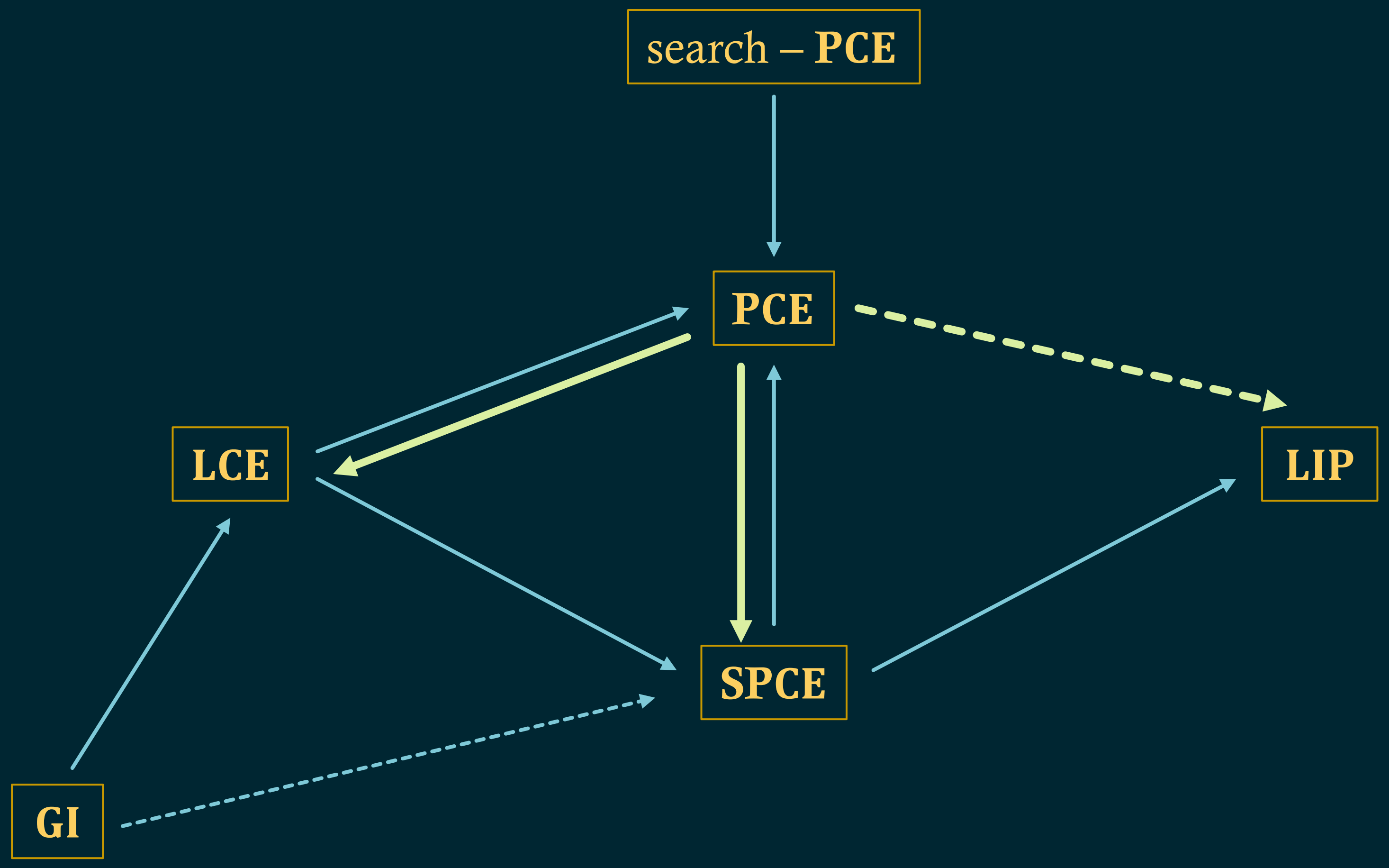
- YES if there exists invertible $\mathbf{S} \in GL_k$ and *orthogonal* $\mathbf{O} \in \mathcal{O}_n$ such that $\mathbf{S}\,\mathbf{B_1}\mathbf{O} = \mathbf{B_2}$
- NO if otherwise.



Known Reductions



Our Reductions



Our Results

Theorem 1: There is a Karp reduction from PCE to LCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

Theorem 2: There is a Karp reduction from PCE to SPCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

Our Results

Theorem 1: There is a Karp reduction from PCE to LCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

Theorem 2: There is a Karp reduction from PCE to SPCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

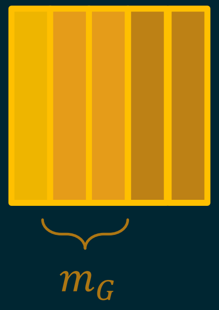
We construct a map that transforms

$$\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k \times n} \rightarrow \mathbf{G}_1', \mathbf{G}_2' \in \mathbb{F}_q^{k' \times n'}$$

such that $(\mathbf{G}_1, \mathbf{G}_2) \in \mathrm{PCE} \Leftrightarrow (\mathbf{G}_1', \mathbf{G}_2') \in \mathrm{LCE}$ (or SPCE).

Our Construction

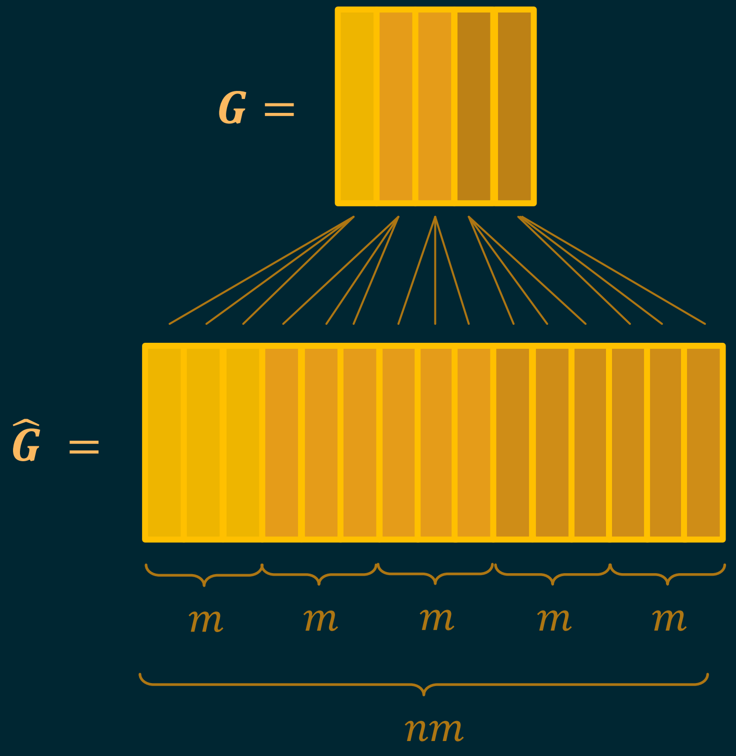
Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, where m_G = maximum number of times a column appears in $\boldsymbol{G}$.

$\boldsymbol{G} =$ 

Our Construction

Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, define $m = m_G + 1$.

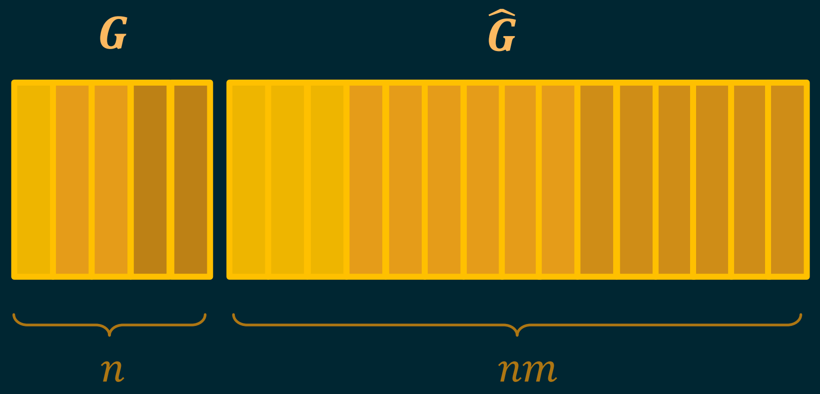
Construct $\widehat{\boldsymbol{G}} \in \mathbb{F}_q^{k \times nm}$:



Our Construction

Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, define $m = m_G + 1$.

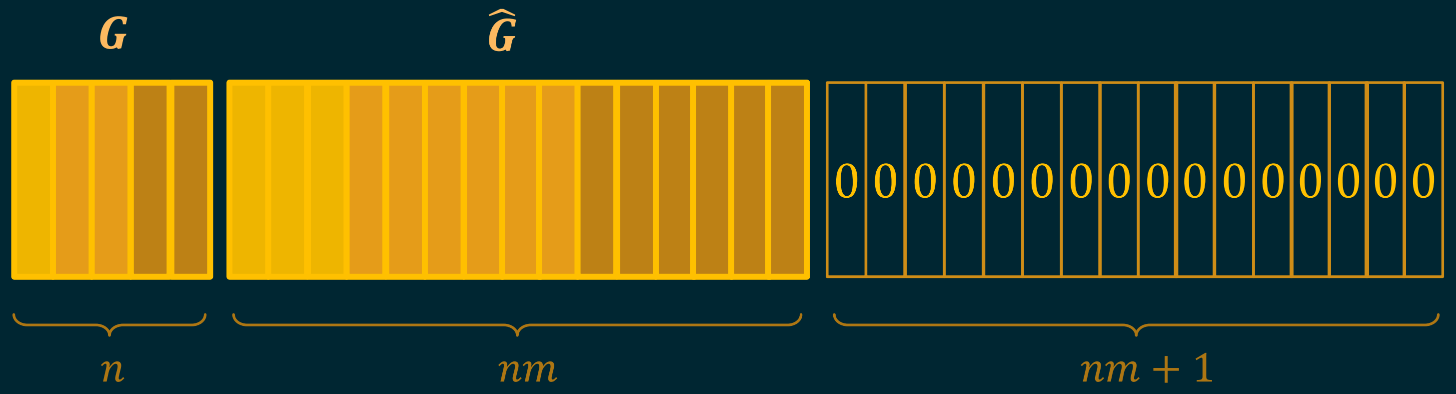
Append $\widehat{\boldsymbol{G}}$ to $\boldsymbol{G}$:



Our Construction

Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, define $m = m_G + 1$.

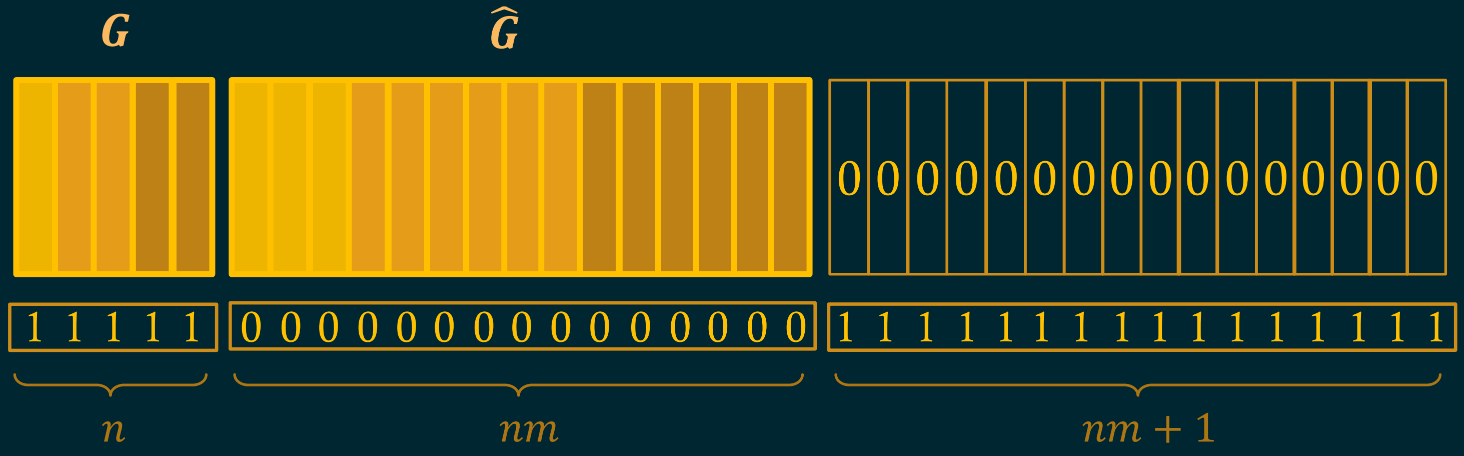
Append zero columns:



Our Construction

Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, define $m = m_G + 1$.

Append the last row:

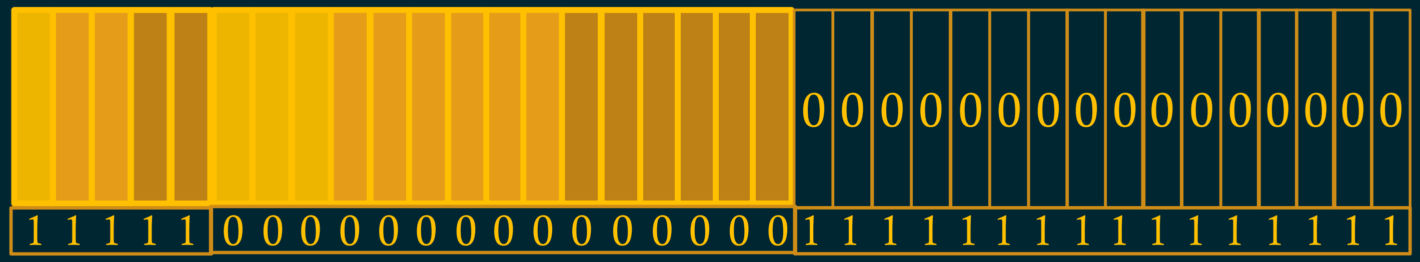


Our Construction

Given generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, define $m = m_G + 1$.

Final matrix is $\boldsymbol{G}' \in \mathbb{F}_q^{(k+1)\times(2nm+n+1)}$:

$\boldsymbol{G}' :=$



Our Results

Theorem 1: There is a Karp reduction from PCE to LCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

Theorem 2: There is a Karp reduction from PCE to SPCE that runs in $\mathbf{poly}(n, \log q)$ time, where the input pair of codes have blocklength n and field size q.

Our map transforms

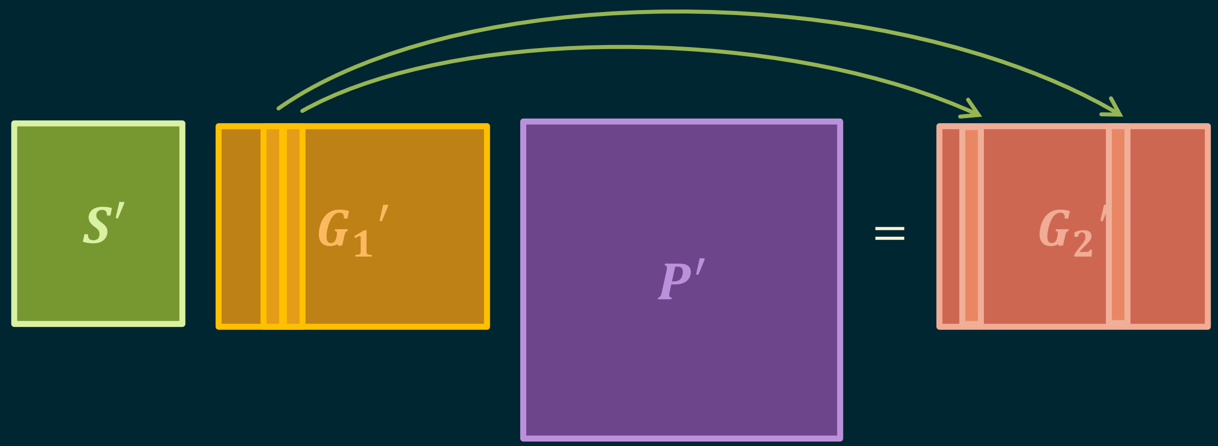
$$\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k \times n} \rightarrow \mathbf{G}_1', \mathbf{G}_2' \in \mathbb{F}_q^{k' \times n'}$$

such that $(\mathbf{G}_1, \mathbf{G}_2) \in \mathrm{PCE} \Leftrightarrow (\mathbf{G}_1', \mathbf{G}_2') \in \mathrm{LCE}$ (or SPCE).

Proof Idea

$$S' \; {G_1}' \; P' = {G_2}'$$

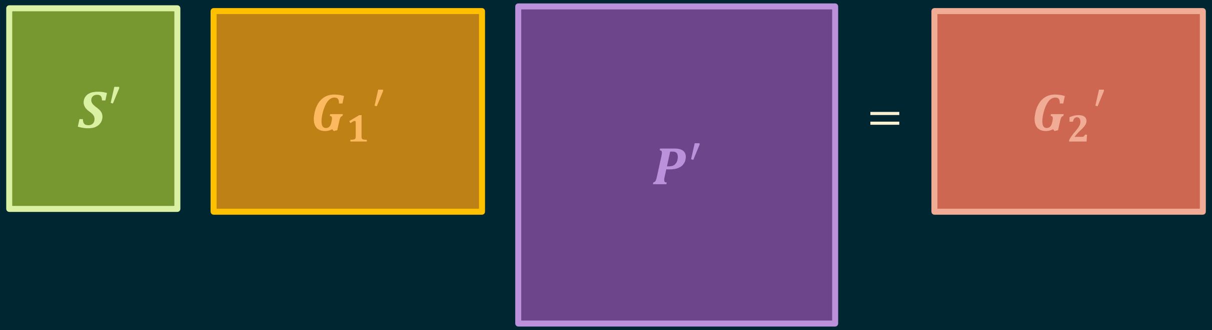
Proof Idea



$\mathbf{S}'$ is a change of basis matrix that defines a bijection over $\mathbb{F}_q^n$.

It maps identical columns in $\mathbf{G_1}'$ to identical columns in $\mathbf{G_2}'$.

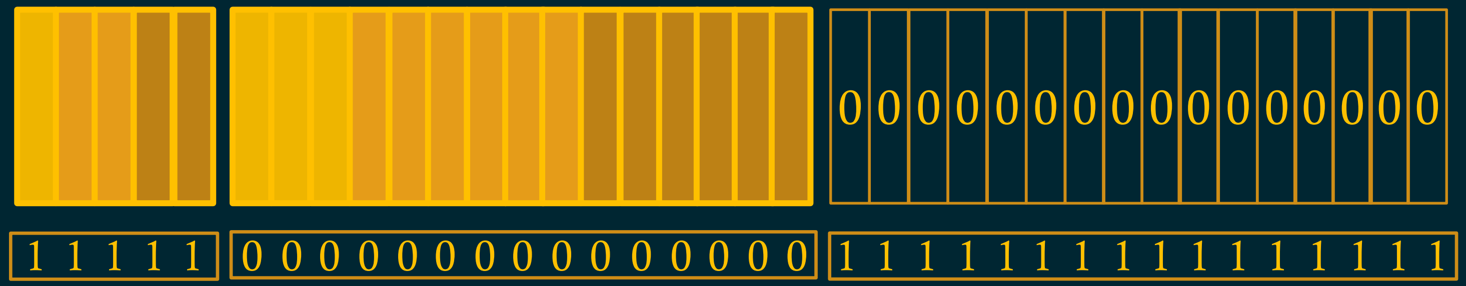
Proof Idea



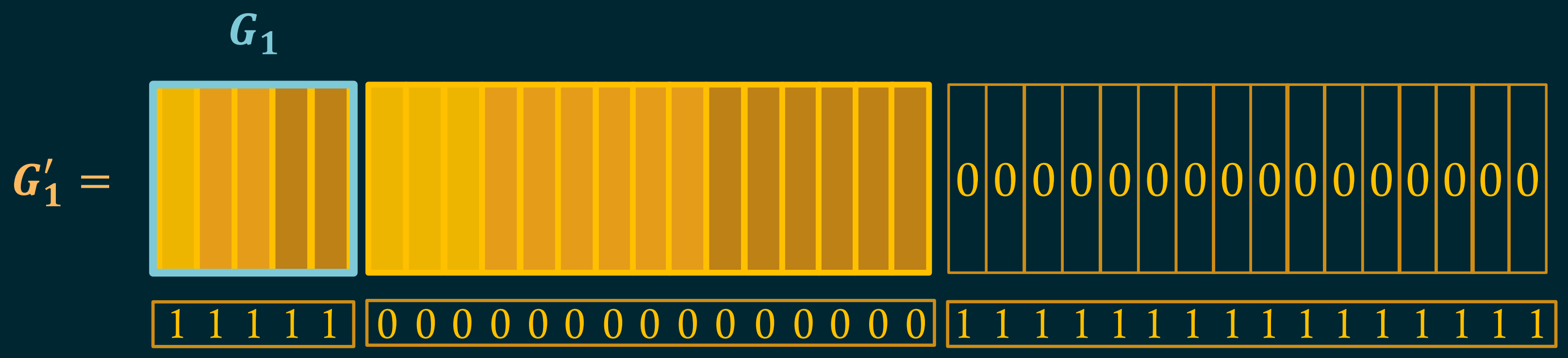
We analyze the structure of the permutation $\mathbf{P}'$ and how it permutes the columns of $\mathbf{G_1}'$.

Proof Idea

$G'_1 =$

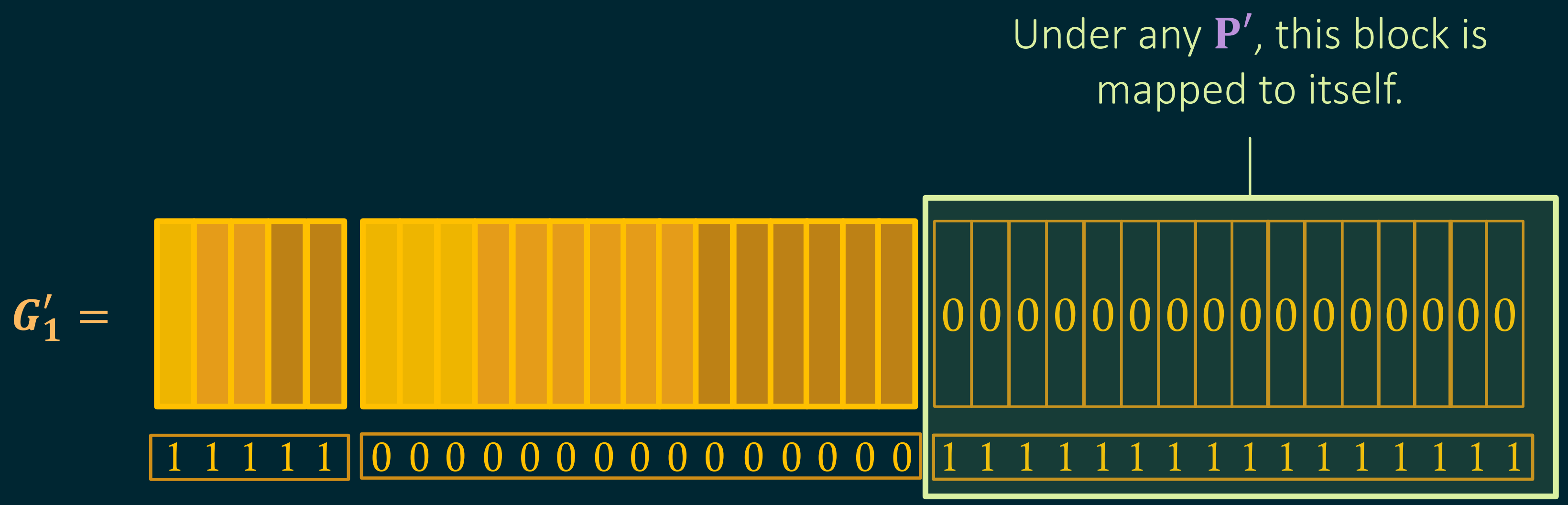


Proof Idea



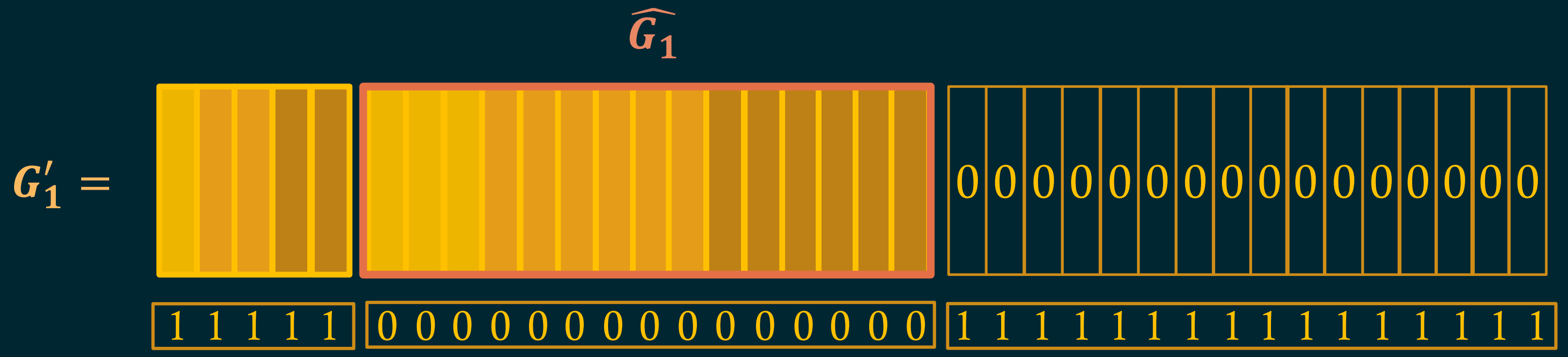
Without loss of generality, we assume that G_1 does not contain an all-zero column.

Proof Idea



Without loss of generality, we assume that $\boldsymbol{G}_1$ does not contain an all-zero column.

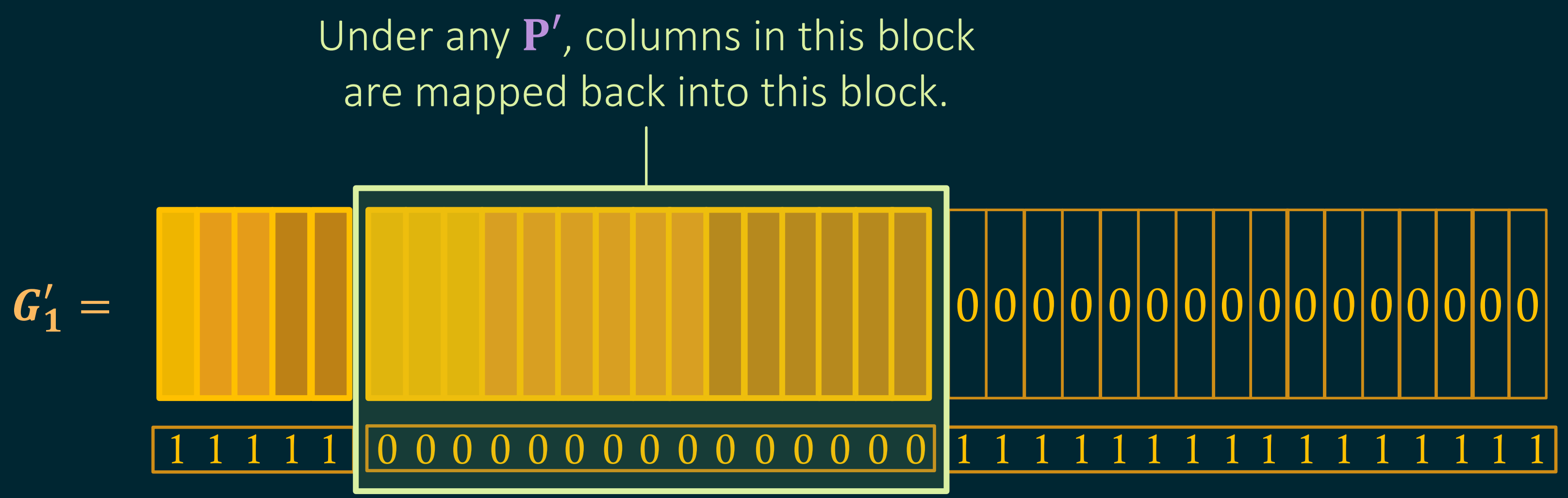
Proof Idea



Every column of G_1 appears $< m$ times.

But every column of $\widehat{G_1}$ appears $\geq m$ times.

Proof Idea

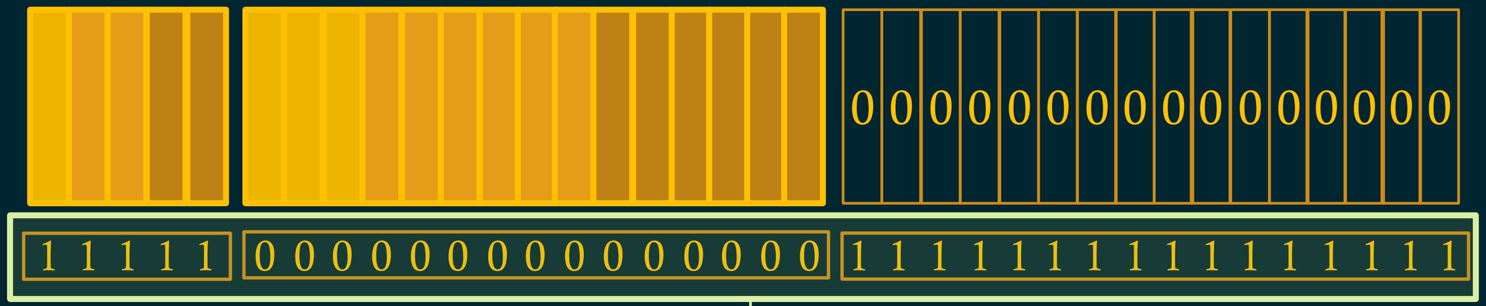


Every column of G_1 appears $< m$ times.

But every column of $\widehat{G_1}$ appears $\geq m$ times.

Proof Idea

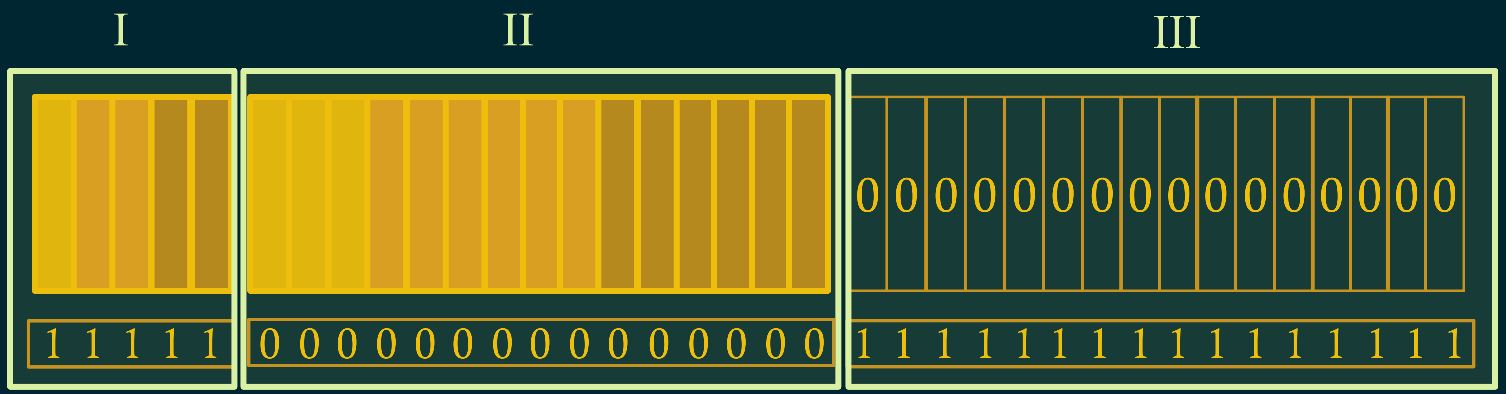
$G_1' =$



This last row prevents $\mathbf{P}'$ from swapping columns from different blocks.

Proof Idea

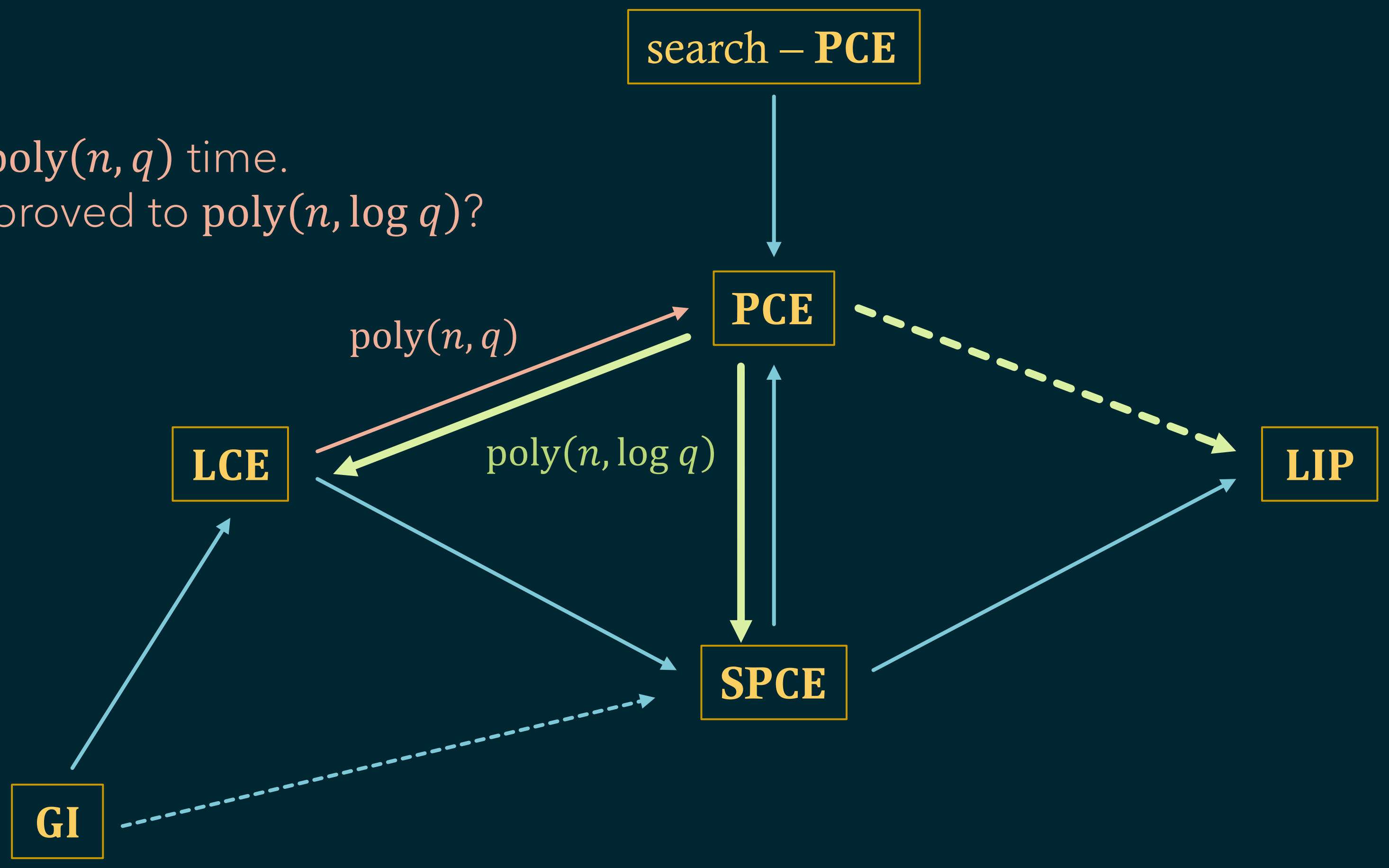
$G_1' =$



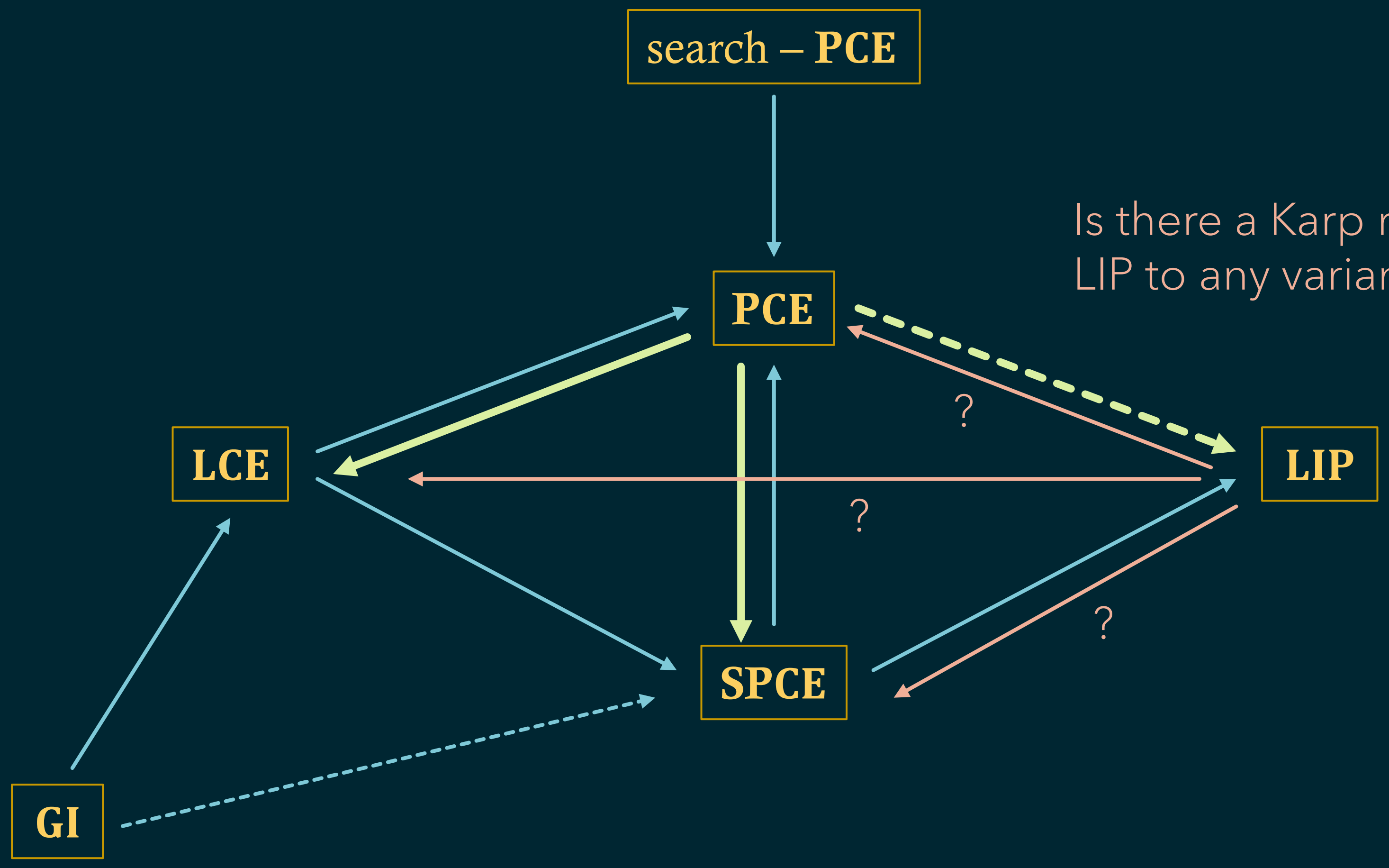
All together, the distribution of columns, zero columns, and last row forces any permutation $\mathbf{P}'$ to respect boundaries and have a block diagonal structure.

Future Directions

This runs in $\mathbf{poly}(n, q)$ time.
Can it be improved to $\mathbf{poly}(n, \log q)$?



Future Directions



Is there a Karp reduction *from* LIP to any variant of CE?

II. Constructions and Algorithms

List-Decoding GRS Codes over General Norms

Based on joint work with Chris Peikert

Codes

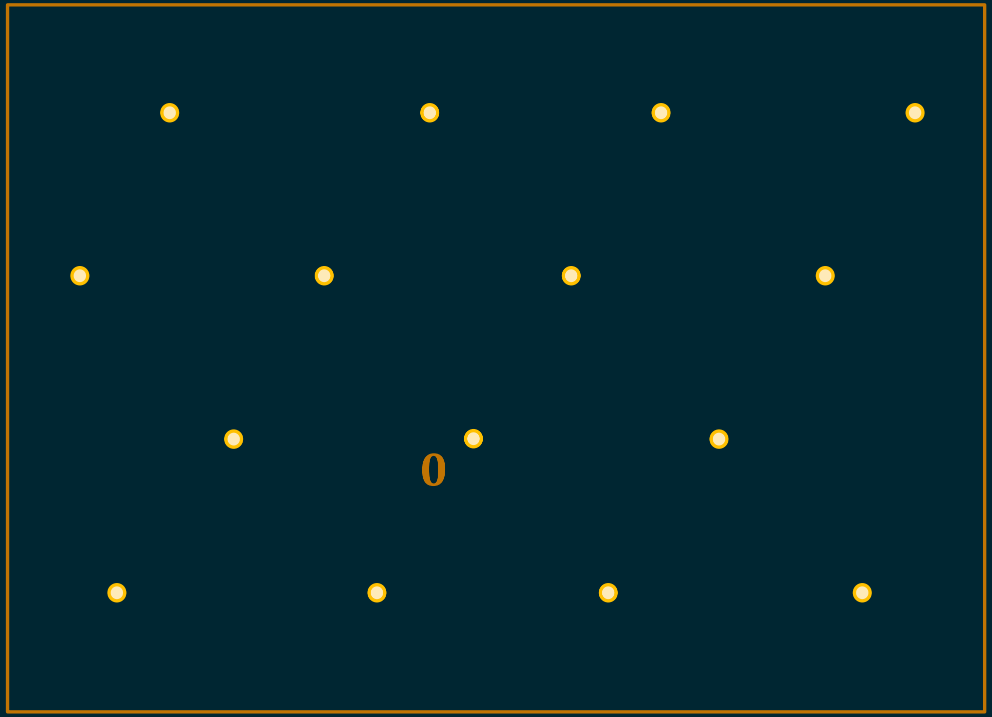
Linear Code:

A linear subspace over a finite field $\mathbb{F}_q$

$$\mathcal{C} = \{ \mathbf{x}\,\mathbf{G} \;:\; \mathbf{x} \in \mathbb{F}_q^k \} \subseteq \mathbb{F}_q^n$$

generated by $\mathbf{G} \in \mathbb{F}_q^{k \times n}$.

n is the *blocklength* and k is the *dimension.*



Generalized Reed-Solomon Codes

GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \dots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) \coloneqq \left\{\left(t_1 \cdot f(\alpha_1), \dots, t_n \cdot f(\alpha_n)\right) : f \in \mathbb{F}_q[x], \deg(f) < k\right\} \subseteq \mathbb{F}_q^n.$$

Generalized Reed-Solomon Codes

GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \dots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) \coloneqq \left\{ \left(t_1 \cdot f(\alpha_1), \dots, t_n \cdot f(\alpha_n) \right) : f \in \mathbb{F}_q[x], \deg(f) < k \right\} \subseteq \mathbb{F}_q^n.$$

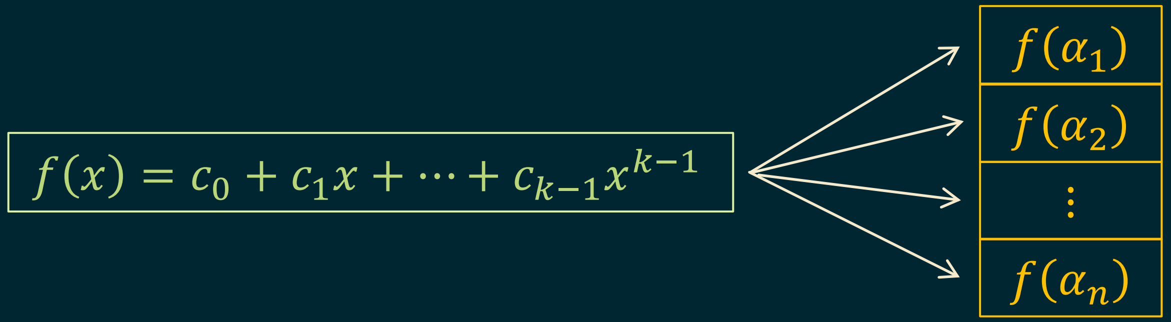
$$f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1}$$

Generalized Reed-Solomon Codes

GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \dots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) \coloneqq \{(t_1 \cdot f(\alpha_1), \dots, t_n \cdot f(\alpha_n)) : f \in \mathbb{F}_q[x], \deg(f) < k\} \subseteq \mathbb{F}_q^n.$$

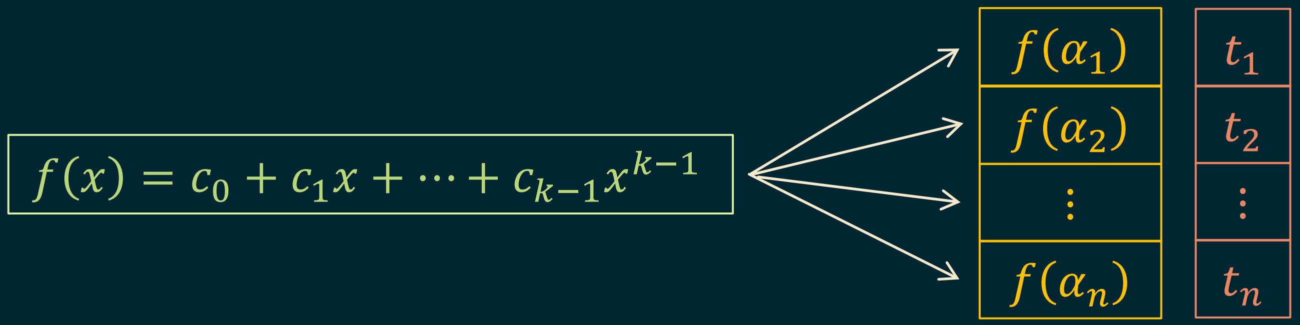


Generalized Reed-Solomon Codes

GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \ldots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) := \{(t_1 \cdot f(\alpha_1), \ldots, t_n \cdot f(\alpha_n)) : f \in \mathbb{F}_q[x], \deg(f) < k\} \subseteq \mathbb{F}_q^n.$$

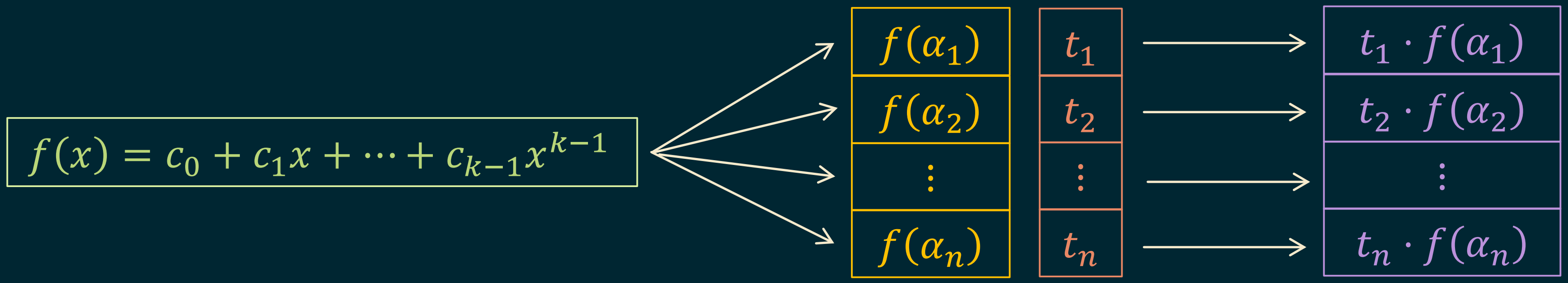


Generalized Reed-Solomon Codes

GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \ldots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) \coloneqq \{(t_1 \cdot f(\alpha_1), \ldots, t_n \cdot f(\alpha_n)) : f \in \mathbb{F}_q[x], \deg(f) < k\} \subseteq \mathbb{F}_q^n.$$

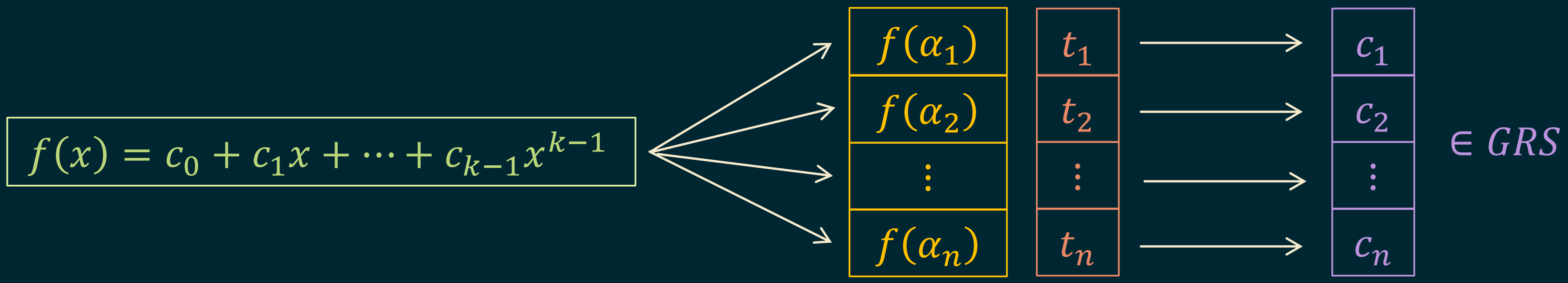


Generalized Reed-Solomon Codes

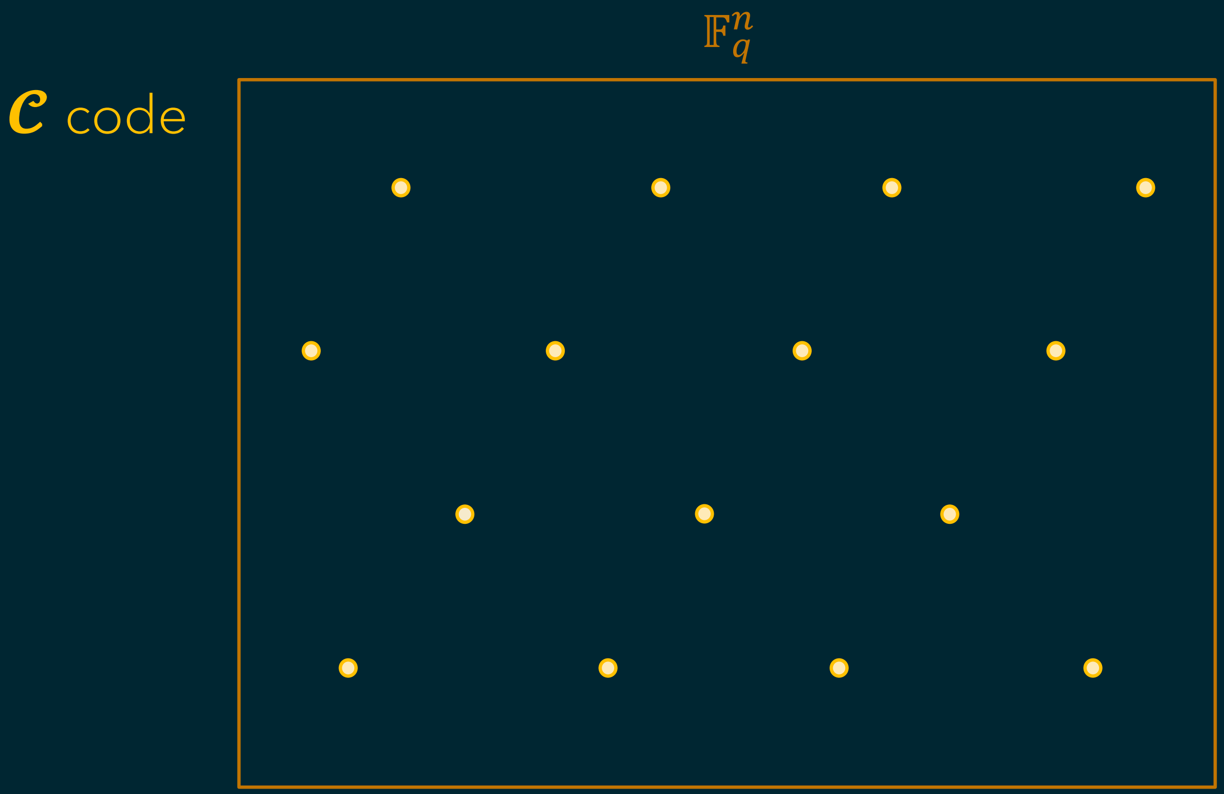
GRS Code: n blocklength, $\mathbb{F}_q$ finite field of size $q \geq n$, k dimension,

$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_q^n$ evaluation points, $\boldsymbol{t} = (t_1, \dots, t_n) \in \mathbb{F}_q^n$ non-zero twist factors

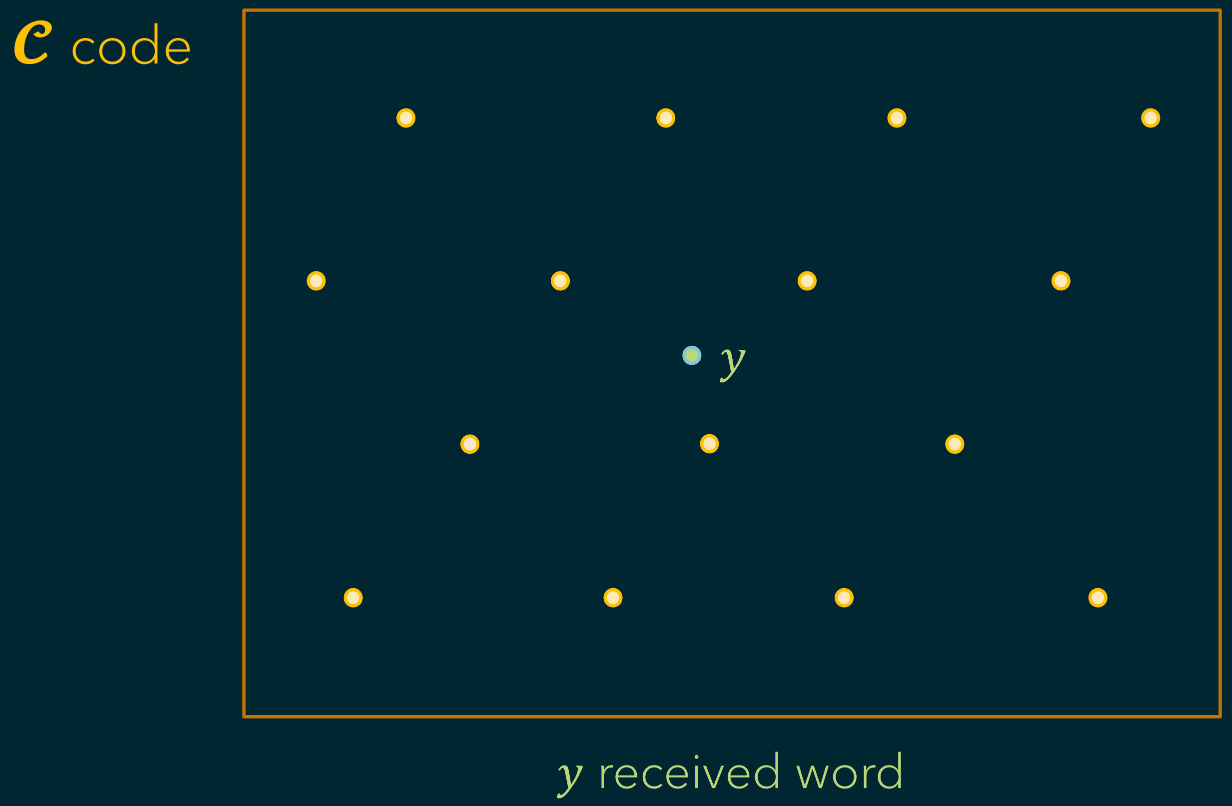
$$GRS_{q,k}(\boldsymbol{\alpha}, \boldsymbol{t}) \coloneqq \left\{ \left(t_1 \cdot f(\alpha_1), \dots, t_n \cdot f(\alpha_n) \right) : f \in \mathbb{F}_q[x], \deg(f) < k \right\} \subseteq \mathbb{F}_q^n.$$



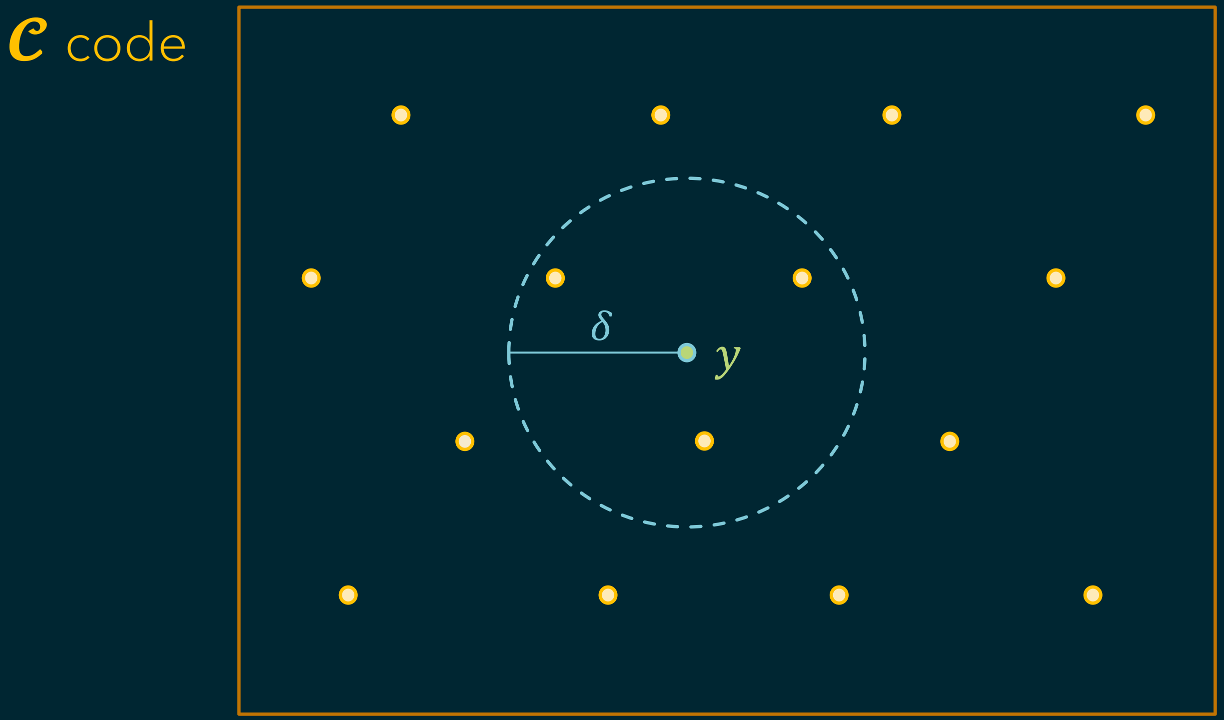
List-Decoding Problem



List-Decoding Problem



List-Decoding Problem

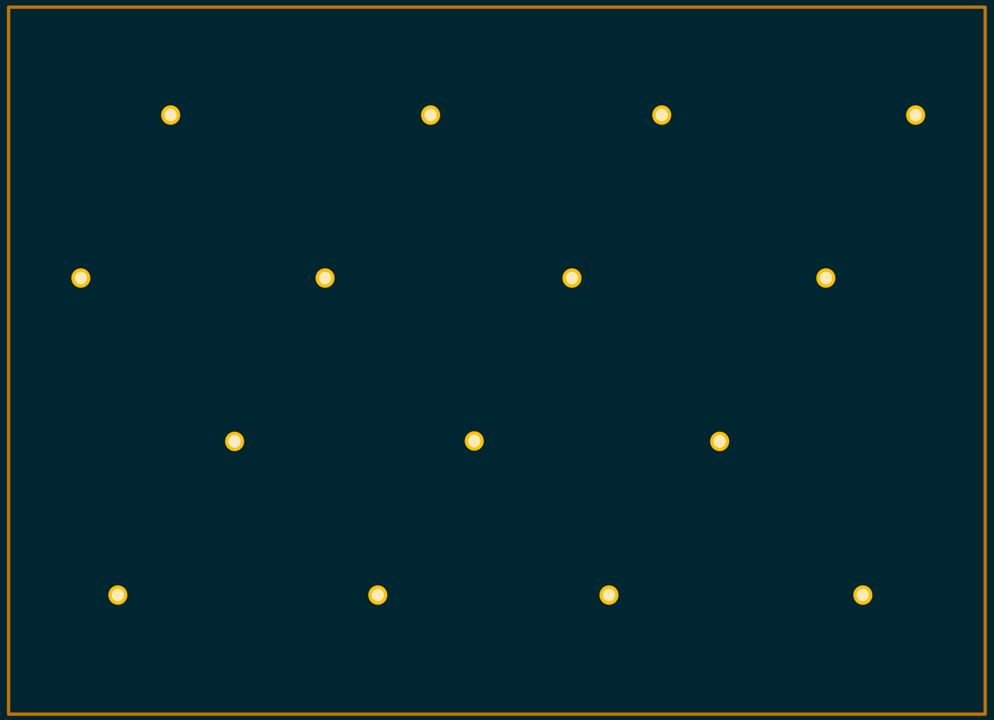


find all codewords within distance δ of y

Rate-Distance Trade-off

$\mathcal{C} \subseteq \mathbb{F}_q^n$

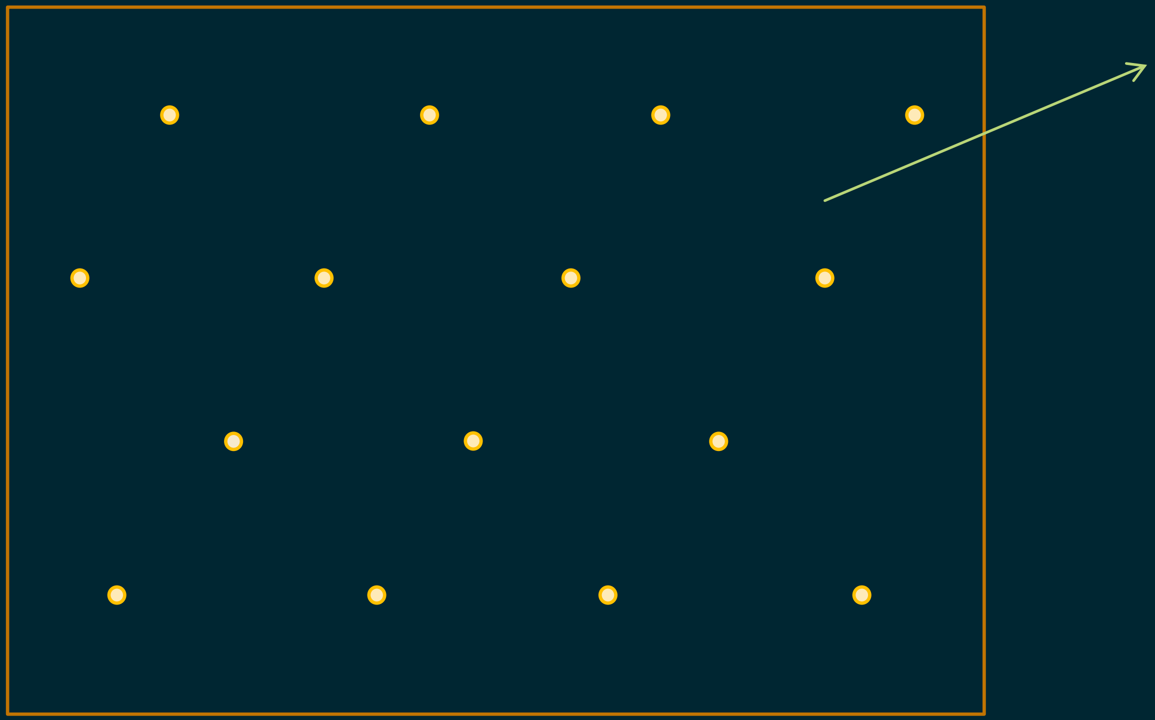
k dimension



Rate-Distance Trade-off

$\mathcal{C} \subseteq \mathbb{F}_q^n$

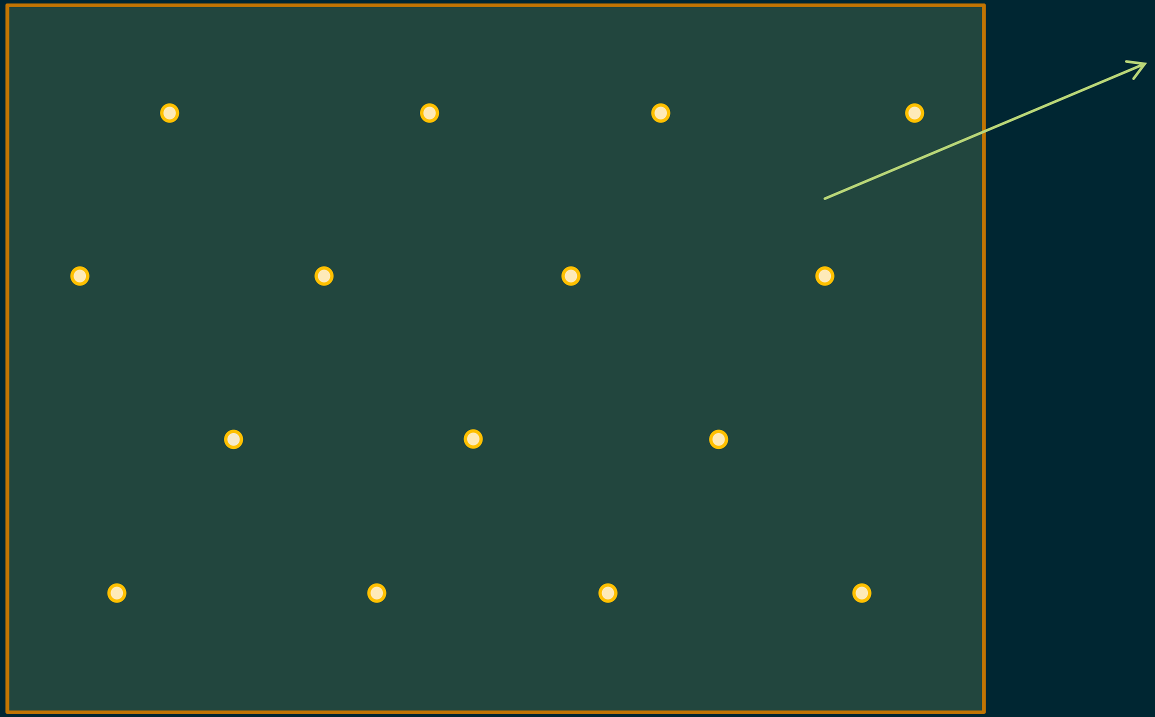
k dimension



Rate-Distance Trade-off

$\mathcal{C} \subseteq \mathbb{F}_q^n$

k dimension



"density" of a code

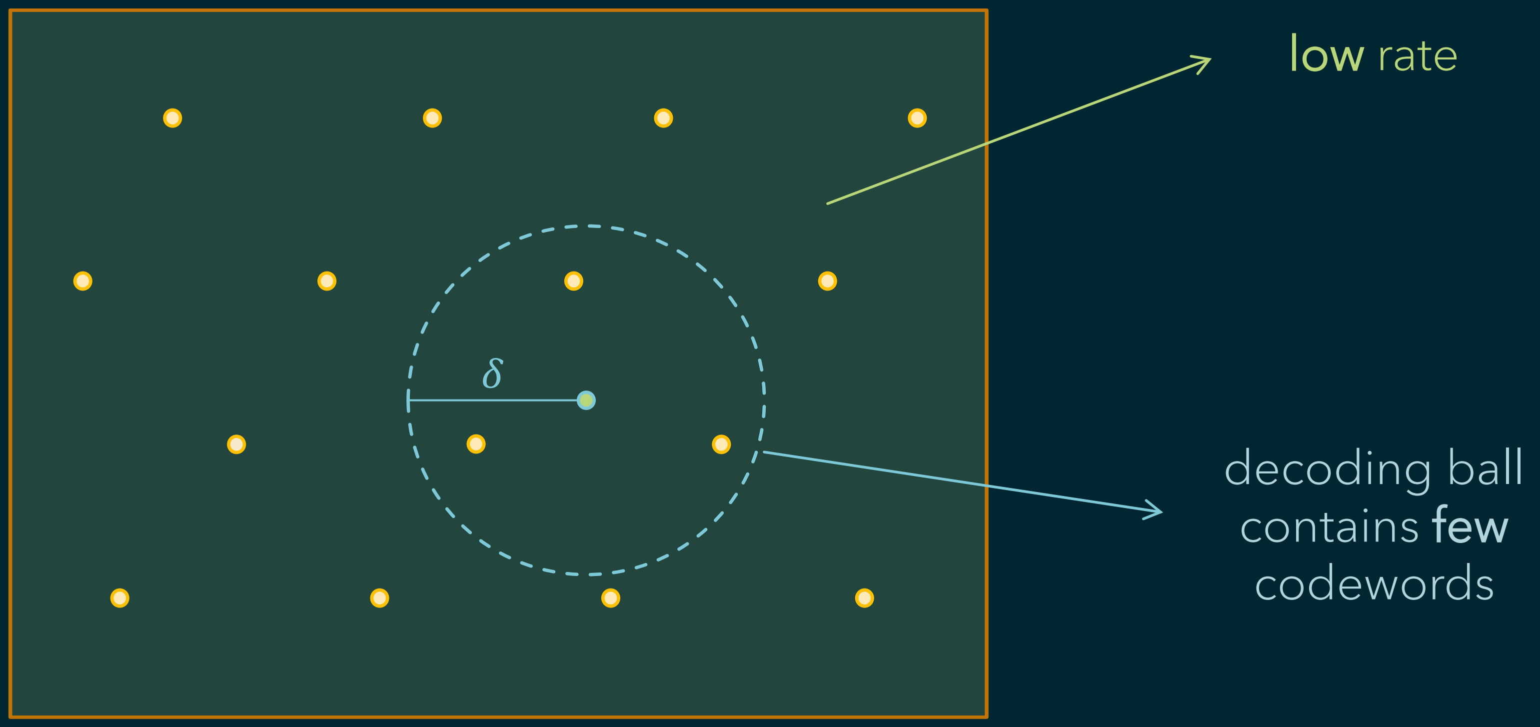
$$R^* = \frac{k-1}{n}$$

(adjusted) **rate**

Rate-Distance Trade-off

$\mathcal{C} \subseteq \mathbb{F}_q^n$

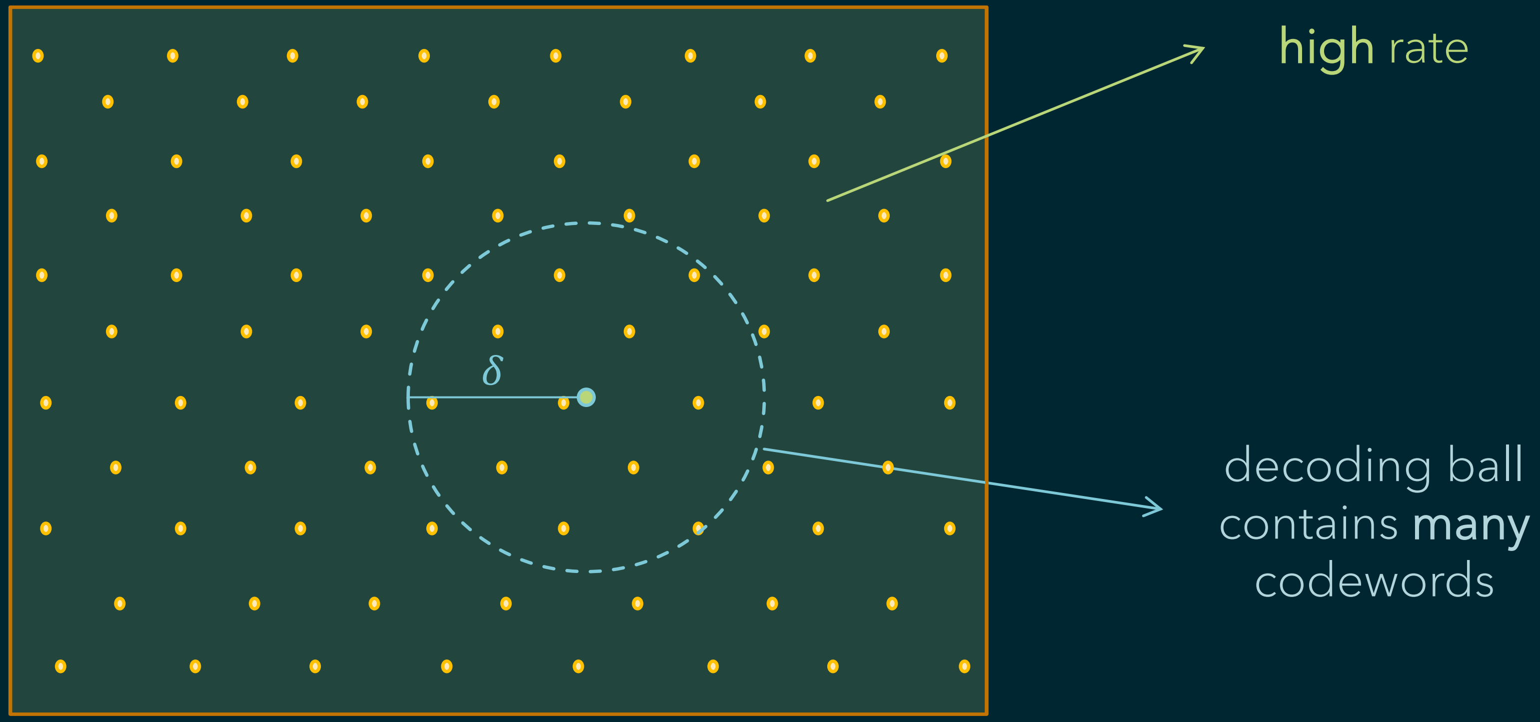
k dimension



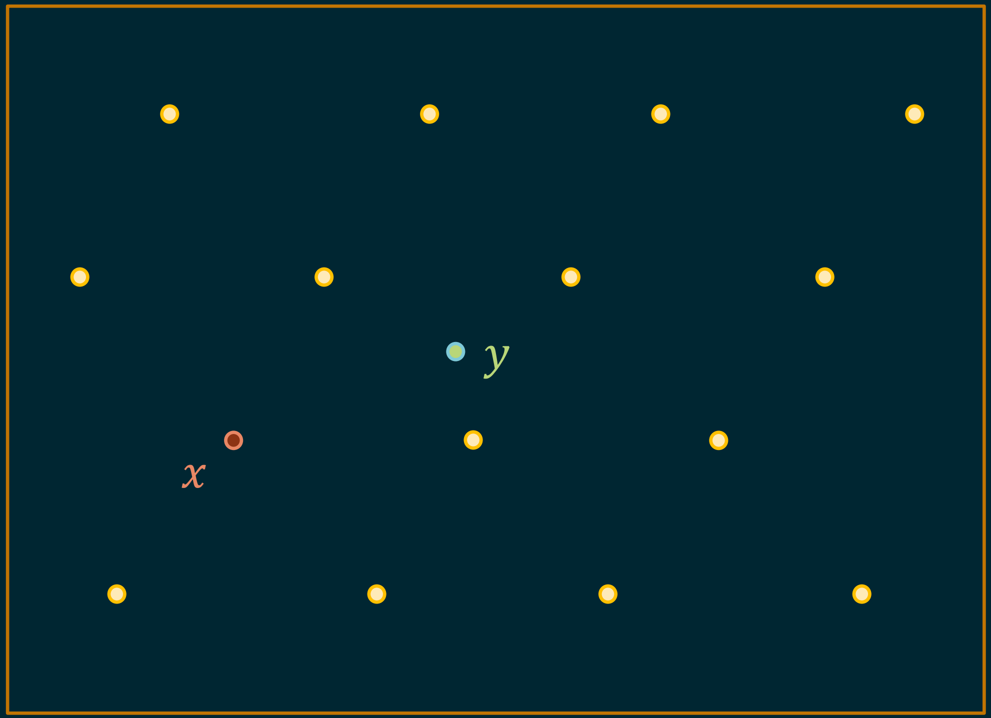
Rate-Distance Trade-off

$\mathcal{C} \subseteq \mathbb{F}_q^n$

k dimension



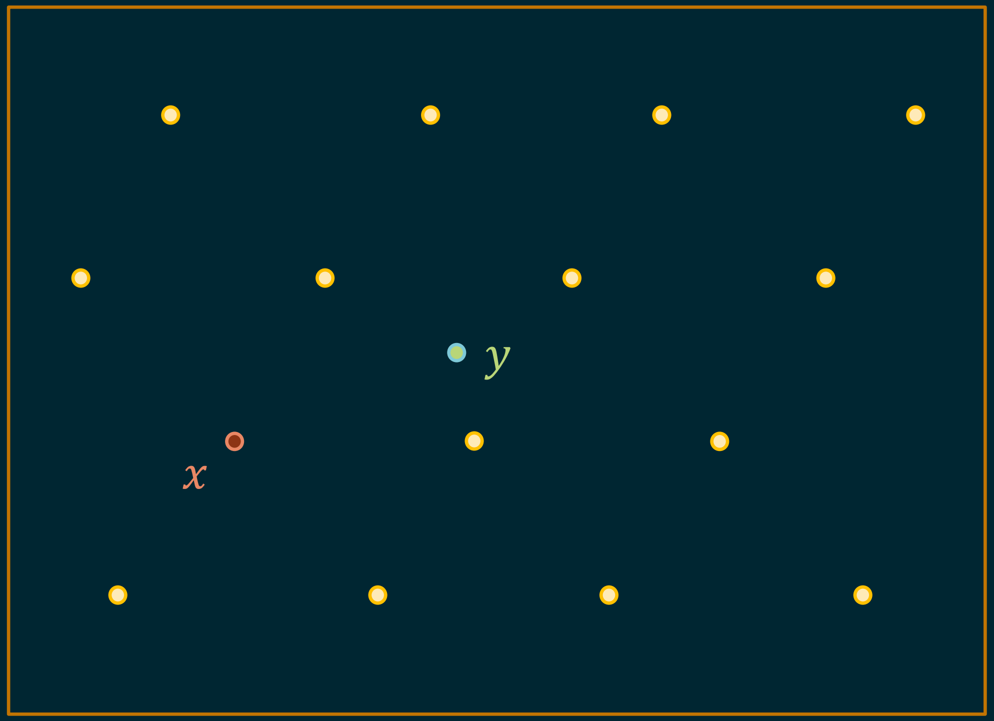
Measuring Distance



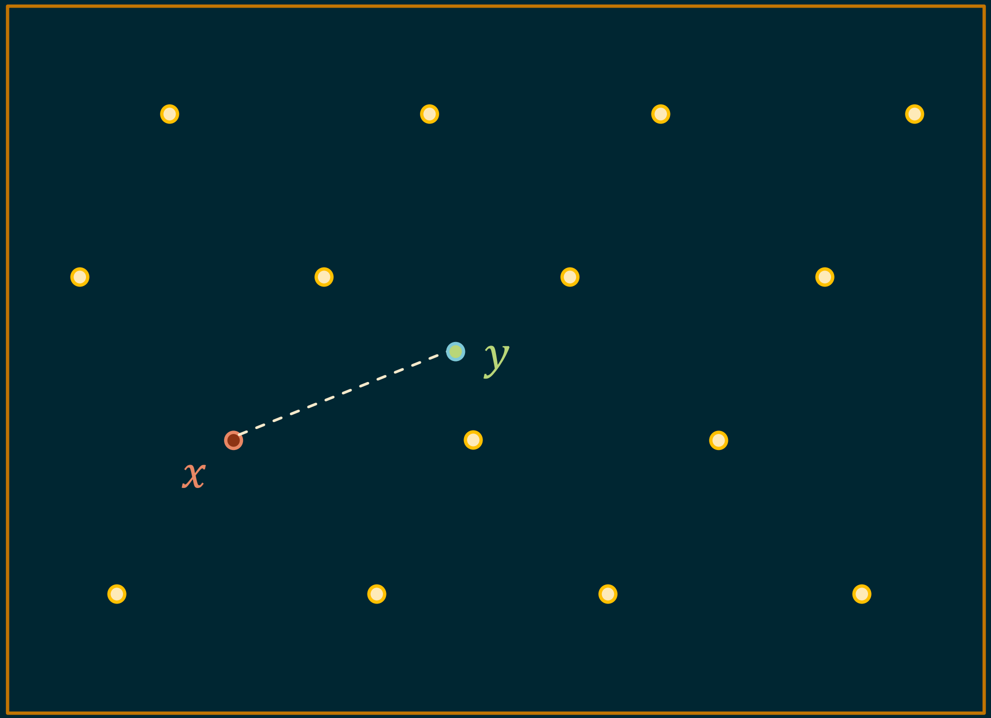
How is distance measured?

Measuring Distance

$$\mathbb{R}_q^n = (\mathbb{R}/q\mathbb{Z})^n$$

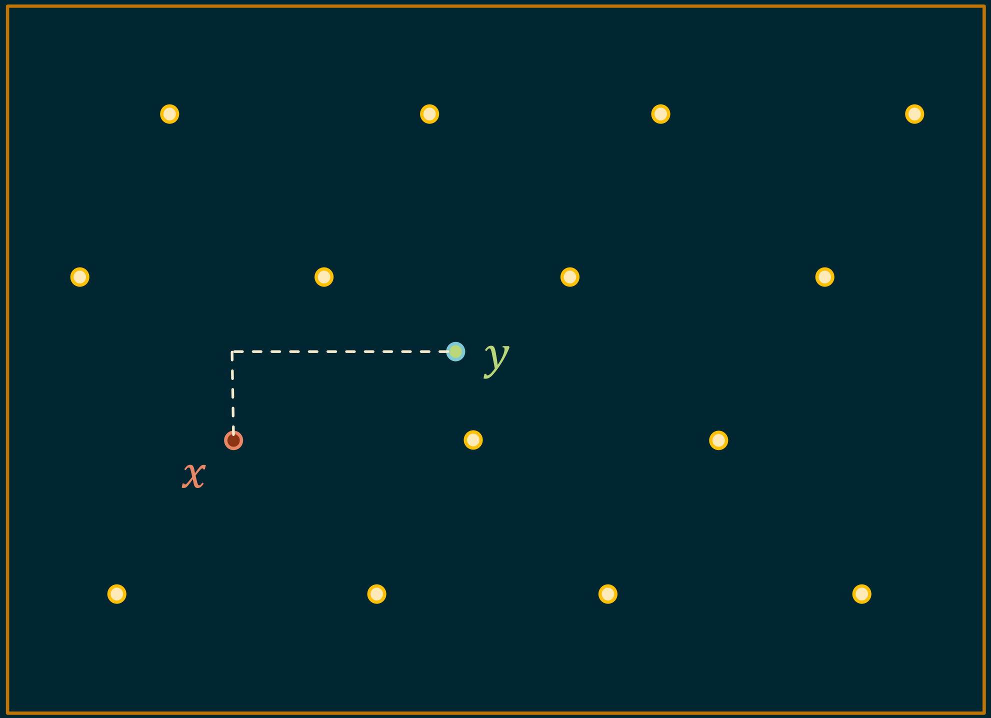


Measuring Distance



ℓ_2 norm (Euclidean distance)

Measuring Distance



ℓ_1 norm (Manhattan distance)

General (Quasi)Norms

ℓ_p(Quasi)Norm: $p > 0$

For any vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, its length in the ℓ_p (quasi)norm is

$$\|x\|_p := \left(x_1^p + \cdots + x_n^p\right)^{1/p}.$$

Our Results

Theorem: *(informal)* There is an efficient algorithm that list-decodes GRS codes

from both worst-case and average-case errors in the ℓ_p (quasi)norm for any $0 < p \leq 2$.

Our Results

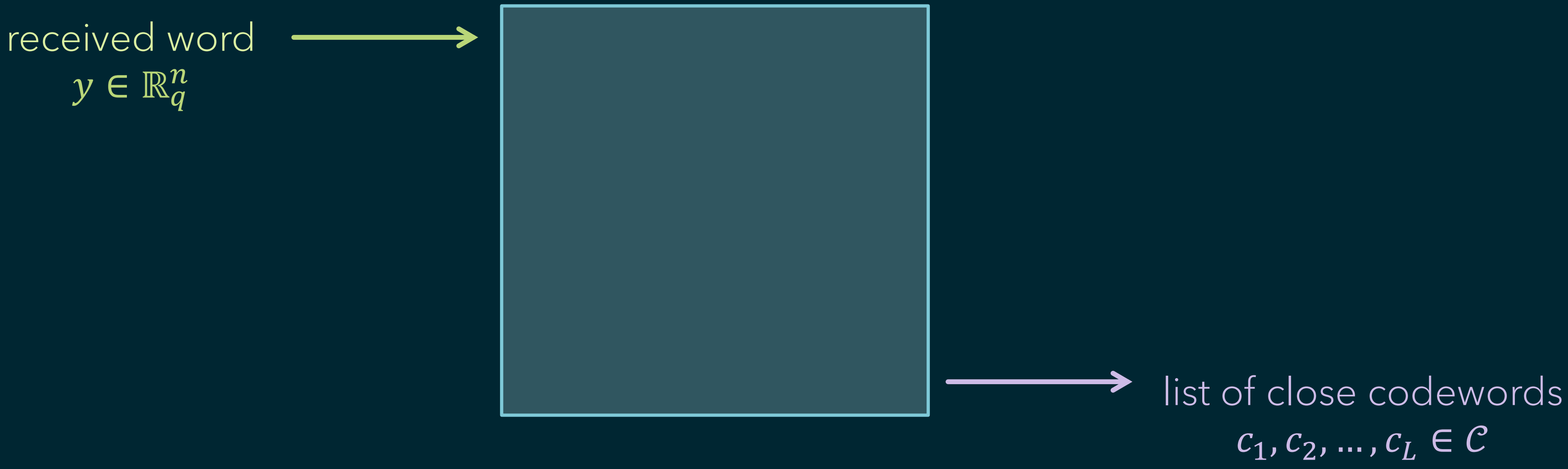
Theorem: *(informal)* There is an efficient algorithm that list-decodes GRS codes from both worst-case and average-case errors in the ℓ_p (quasi)norm for any $0 < p \leq 2$.

Prior algorithms: Hamming metric (many works),

ℓ_2 norm [Mook-Peikert, 2022],

ℓ_1 norm [Roth-Siegel, 1994]

List-decoding Algorithm



Soft-decision Decoding Algorithm

weight vector

$W = (\overrightarrow{w_1}, \dots, \overrightarrow{w_i}, \dots, \overrightarrow{w_n}) \in [0,1]^{qn}$

list of close codewords

$c_1, c_2, \dots, c_L \in \mathcal{C}$

Soft-decision Decoding Algorithm

weight vector

$W = (\overrightarrow{w_1}, \dots, \overrightarrow{w_i}, \dots, \overrightarrow{w_n}) \in [0,1]^{qn}$

$w_i(x_1)$	$w_i(x_2)$	…	$w_i(x_q)$

$w_i(x)$ specifies the ``likelihood'' that

x was the i-th transmitted symbol

list of close codewords

$c_1, c_2, \dots, c_L \in \mathcal{C}$

Guruswami-Sudan Algorithm

[Guruswami-Sudan, 1998], [Koetter-Vardy, 2003], [Guruswami, 2001]

There is a deterministic *soft-decoding* algorithm for (Generalized) Reed-Solomon codes

$\mathcal{C} \subseteq \mathbb{F}_q^n$ with prime field size q, dimension k, adjusted rate $R^* = \frac{k-1}{n}$, with

Input: weight vector $\mathrm{W} = (\overrightarrow{\mathrm{w}_1}, \ldots, \overrightarrow{\mathrm{w}_n}) \in [0,1]^{qn}$,

tolerance parameter $\tau > 0$

Output: list of all codewords $\boldsymbol{c} \in \mathcal{C}$ that are "closely correlated" with W

$$\mathrm{corr}(\mathrm{W}, \boldsymbol{c}) \gtrsim \sqrt{R^*}. + \tau.$$

running in poly $\left(n, q, \frac{1}{\tau \|W\|}\right)$ time.

Guruswami-Sudan Algorithm

[Guruswami-Sudan, 1998], [Koetter-Vardy, 2003], [Guruswami, 2001]

There is a deterministic *soft-decoding* algorithm for (Generalized) Reed-Solomon codes

$\mathcal{C} \subseteq \mathbb{F}_q^n$ with prime field size q, dimension k, adjusted rate $R^* = \frac{k-1}{n}$, with

Input: weight vector $\mathrm{W} = (\overrightarrow{\mathrm{w}_1}, \dots, \overrightarrow{\mathrm{w}_n}) \in [0,1]^{qn}$,

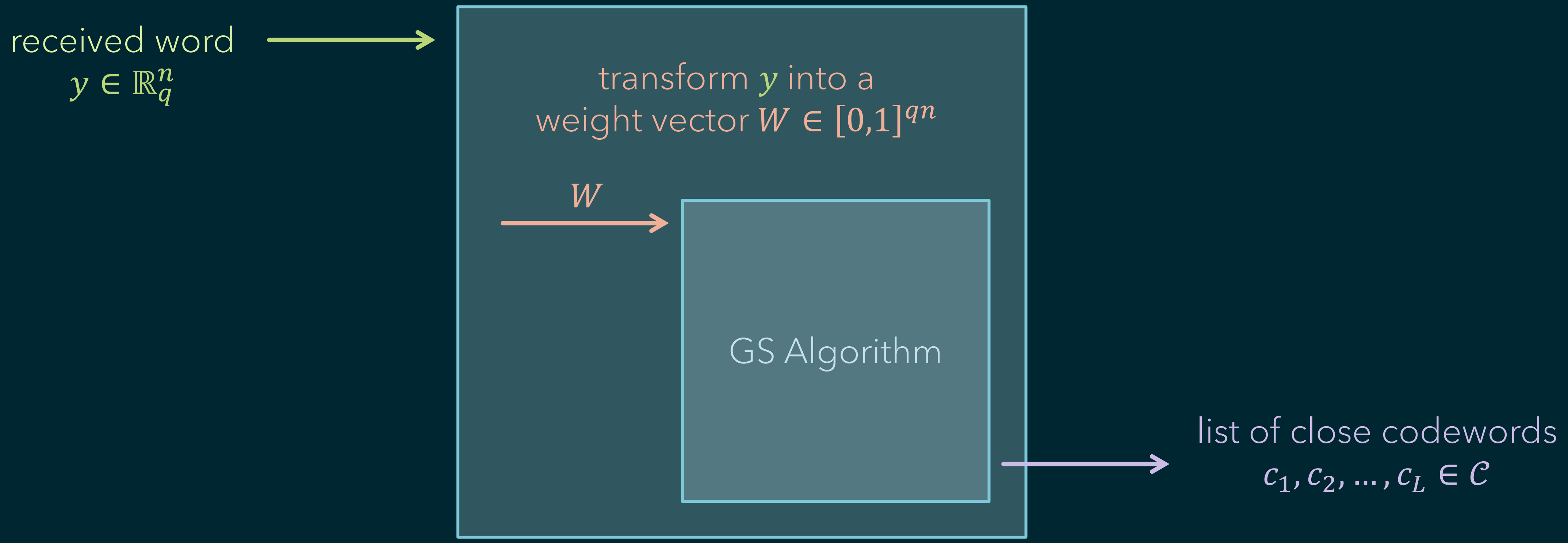
tolerance parameter $\boldsymbol{\tau} > 0$

Output: list of all codewords $\boldsymbol{c} \in \mathcal{C}$ that are "closely correlated" with W

$$\mathrm{corr}(\mathrm{W}, \boldsymbol{c}) \geq \sqrt{R^*} + \boldsymbol{\tau}.$$

running in $\mathrm{poly}\left(n, q, \frac{1}{\tau \|W\|}\right)$ time.

Our List-decoding Algorithm



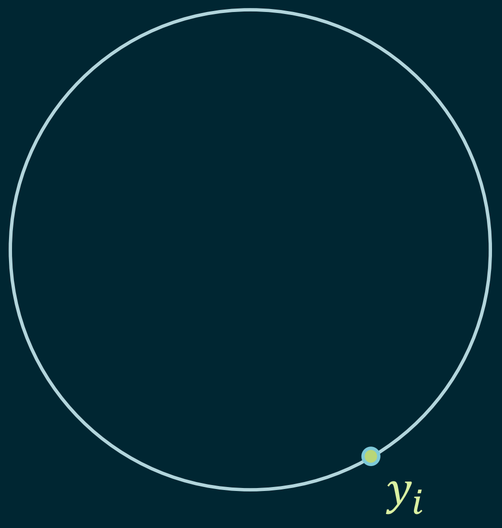
Transforming into Weights

received word $y = \begin{array}{|c|c|c|c|}\hline y_1 & y_2 & \dots & y_n \\ \hline\end{array} \in \mathbb{R}_q^n$

Transforming into Weights

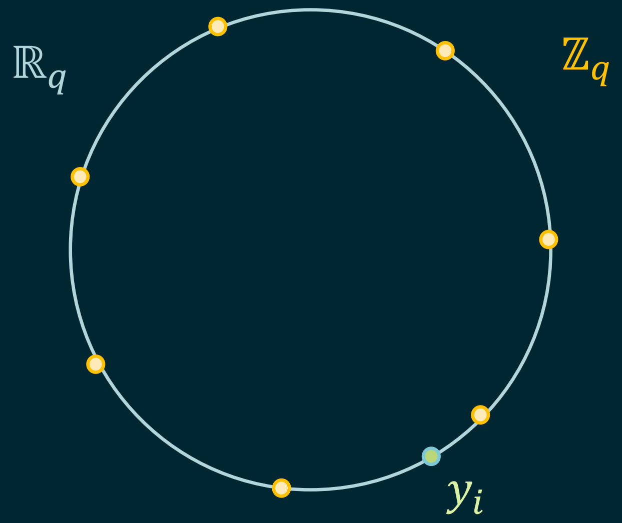
received word $y = \begin{array}{|c|c|c|c|}\hline y_1 & y_2 & \dots & y_n \\ \hline\end{array} \in \mathbb{R}_q^n$

$\mathbb{R}_q = \mathbb{R}/q\mathbb{Z}$



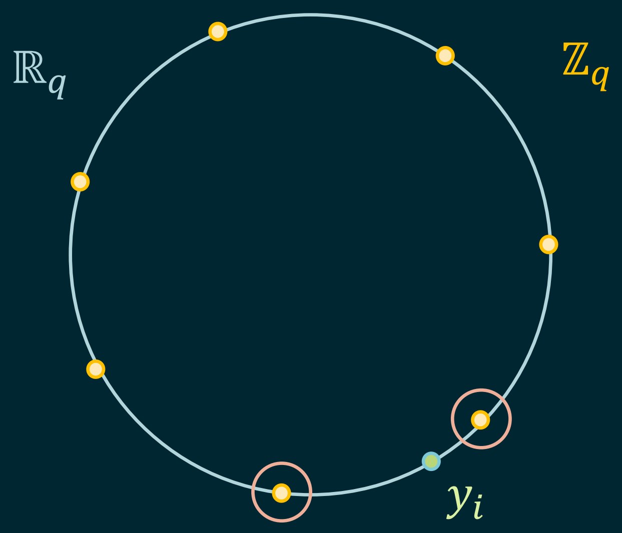
Transforming into Weights

received word $y = \boxed{y_1 \mid y_2 \mid \dots \mid y_n} \in \mathbb{R}_q^n$



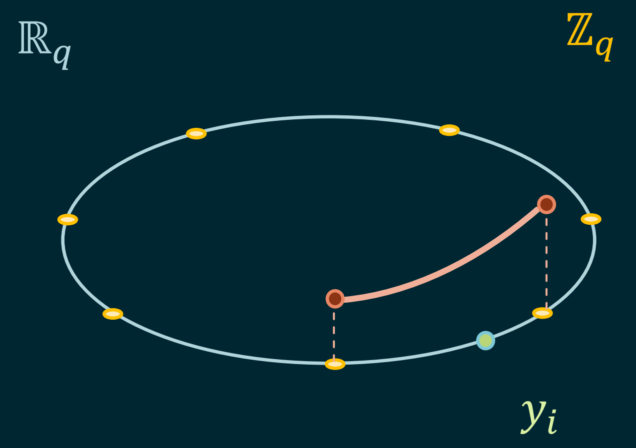
Transforming into Weights

received word $y = \begin{array}{|c|c|c|c|} \hline y_1 & y_2 & \dots & y_n \\ \hline \end{array} \in \mathbb{R}_q^n$



Transforming into Weights

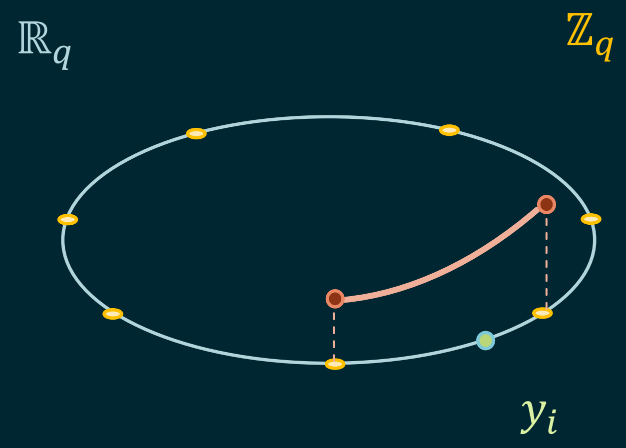
received word $y = \begin{array}{|c|c|c|c|} \hline y_1 & y_2 & \dots & y_n \\ \hline \end{array} \in \mathbb{R}_q^n$



Transforming into Weights

received word $y = \boxed{y_1 \mid y_2 \mid \dots \mid y_n} \in \mathbb{R}_q^n$

[Mook-Peikert, 2022] :



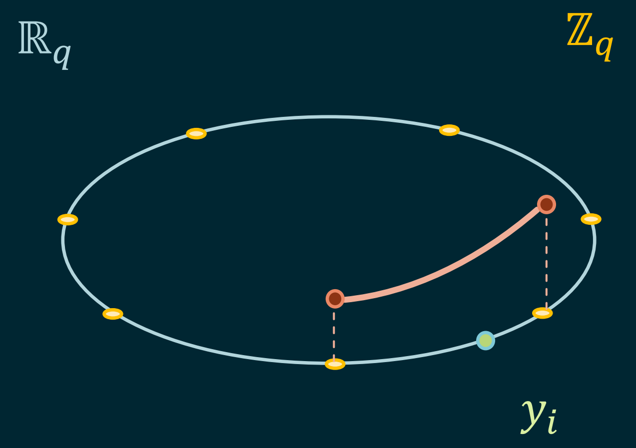
i-th weight vector

$\vec{w_i} = \boxed{0 \mid 0 \mid w_i \mid w_i' \mid 0 \mid 0 \mid 0}$

Transforming into Weights

received word $y =$ | y_1 | y_2 | ... | y_n | $\in \mathbb{R}_q^n$

[Mook-Peikert, 2022] :



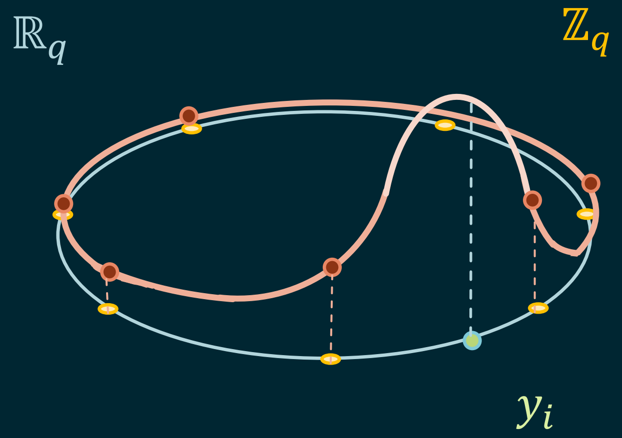
weight vector

$\vec{w_1} =$	0	0	w_1	w_1'	0	0	0
$\vec{w_2} =$	w_2'	0	0	0	0	0	w_2
$\vdots$							
$\vec{w_n} =$	0	0	0	0	w_n	w_n'	0

Transforming into Weights

received word $y = \begin{array}{|c|c|c|c|}\hline y_1 & y_2 & \dots & y_n \\ \hline\end{array} \in \mathbb{R}_q^n$

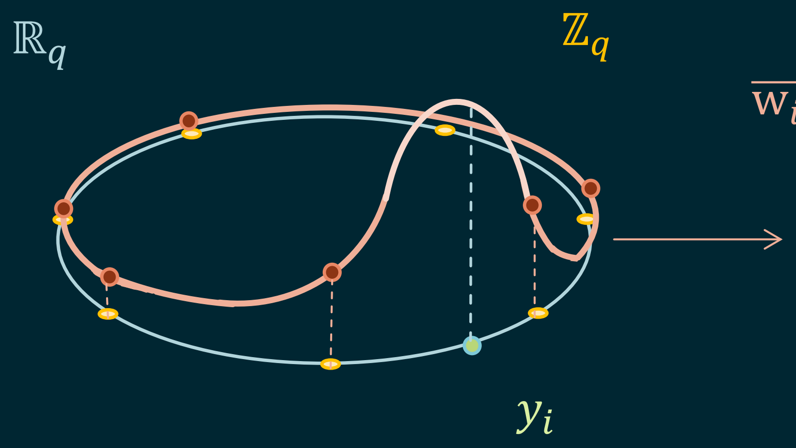
Our weight vector :



Transforming into Weights

received word $y = \boxed{y_1 \mid y_2 \mid \ldots \mid y_n} \in \mathbb{R}_q^n$

Our weight vector :



i-th weight vector

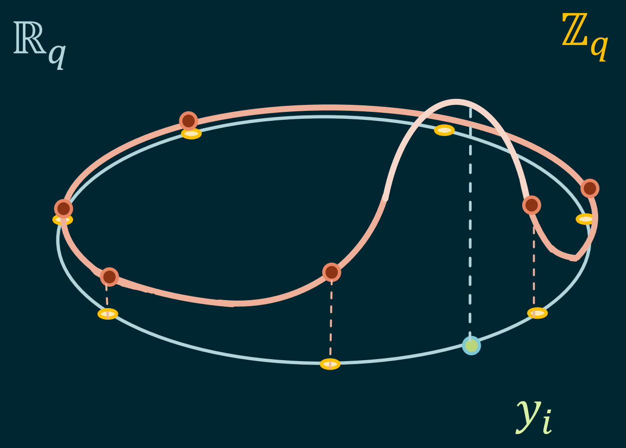
$\overrightarrow{w_i} = \boxed{w_{y_i}(x_1) \mid \ldots \mid w_{y_i}(x_q)}$

weights given by a function f_s of width $s > 0$

Transforming into Weights

received word $y = \boxed{y_1 \mid y_2 \mid \dots \mid y_n} \in \mathbb{R}_q^n$

Our weight vector :



i-th weight vector

$\vec{w_i} = \boxed{w_{s,y_i}(x_1) \mid \dots \mid w_{s,y_i}(x_q)}$

$$w_{s,y_i}(x) = f_s(y_i - x + q\mathbb{Z})$$

determined by the distance between y_i and symbol x

Choosing the Weight Function

We can choose any nicely behaved function f that satisfies certain properties.

But some functions are more natural for specific norms…

Choosing the Weight Function

For distances measured in the ℓ_p norm:

$$f_s^{(p)}(x) \coloneqq \exp(-(c_p \cdot |x/s|)^p)$$

↓

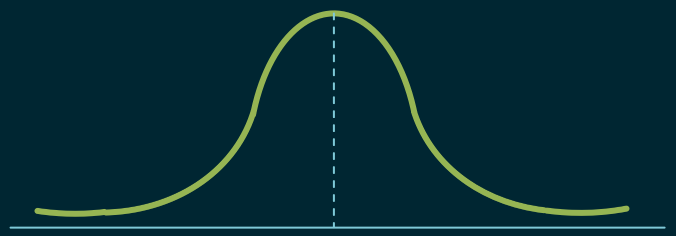
normalizing constant

Choosing the Weight Function

For distances measured in the ℓ_2 norm:

$$f_s^{(2)}(x) := \exp(-(\pi \cdot |x/s|)^2)$$

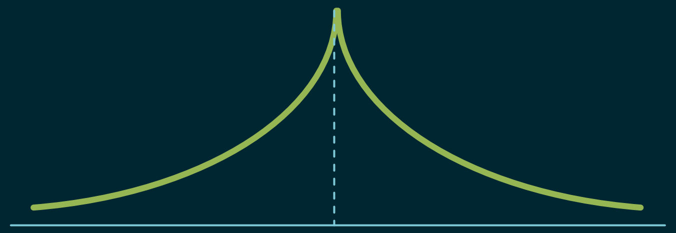
Gaussian function



For distances measured in the ℓ_1 norm:

$$f_s^{(1)}(x) := \exp(-(2 \cdot |x/s|)^1)$$

Laplacian function



Our Main Result

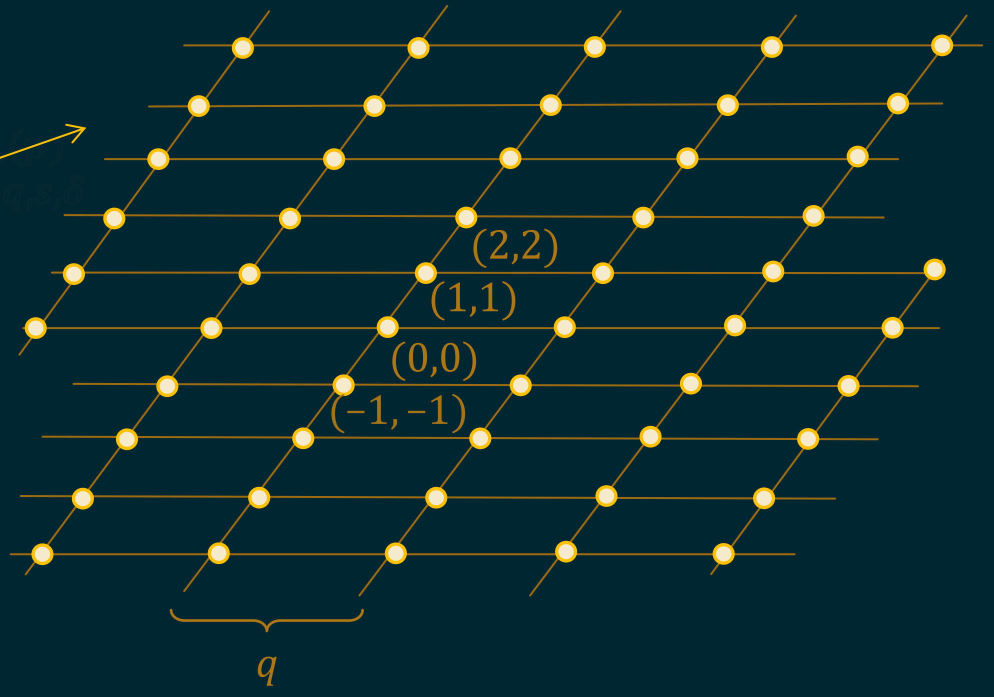
Theorem: For any $0 < p \leq 2$, prime q, and $\delta > 0$, the GS soft-decision algorithm using weight vectors defined by $f_s^{(p)}$ for any $s > 0$, list-decodes up to ℓ_p distance $d = \delta \cdot n^{1/p}$ any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate

$$R^* < \frac{f_s(\delta)^2}{f_s(\mathcal{L}_q)}. =: B_{q,s,\delta}^{(p)}$$

Our Main Result

Theorem: For any $0 < p \leq 2$, prime q, and $\delta > 0$, the GS soft-decision algorithm using weight vectors defined by $f_s^{(p)}$ for any $s > 0$, list-decodes up to ℓ_p distance $d = \delta \cdot n^{1/p}$ any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate

$$R^* < \frac{f_s(\delta)^2}{f_s(\mathcal{L}_q)}.$$



Our Main Result

Theorem: For any $0 < p \leq 2$, prime q, and $\delta > 0$, the GS soft-decision algorithm using weight vectors defined by $f_s^{(p)}$ for any $s > 0$, list-decodes up to ℓ_p distance $d = \delta \cdot n^{1/p}$ any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate

$$R^* < \frac{f_s(\delta)^2}{f_s(\mathcal{L}_q)} =: B_{q,s,\delta}^{(p)}$$

in time $\mathrm{poly}(n, q, \exp(1/s^p)/(B_{q,s,\delta}^{(p)} - \sqrt{R^*}))$.

Our Main Result

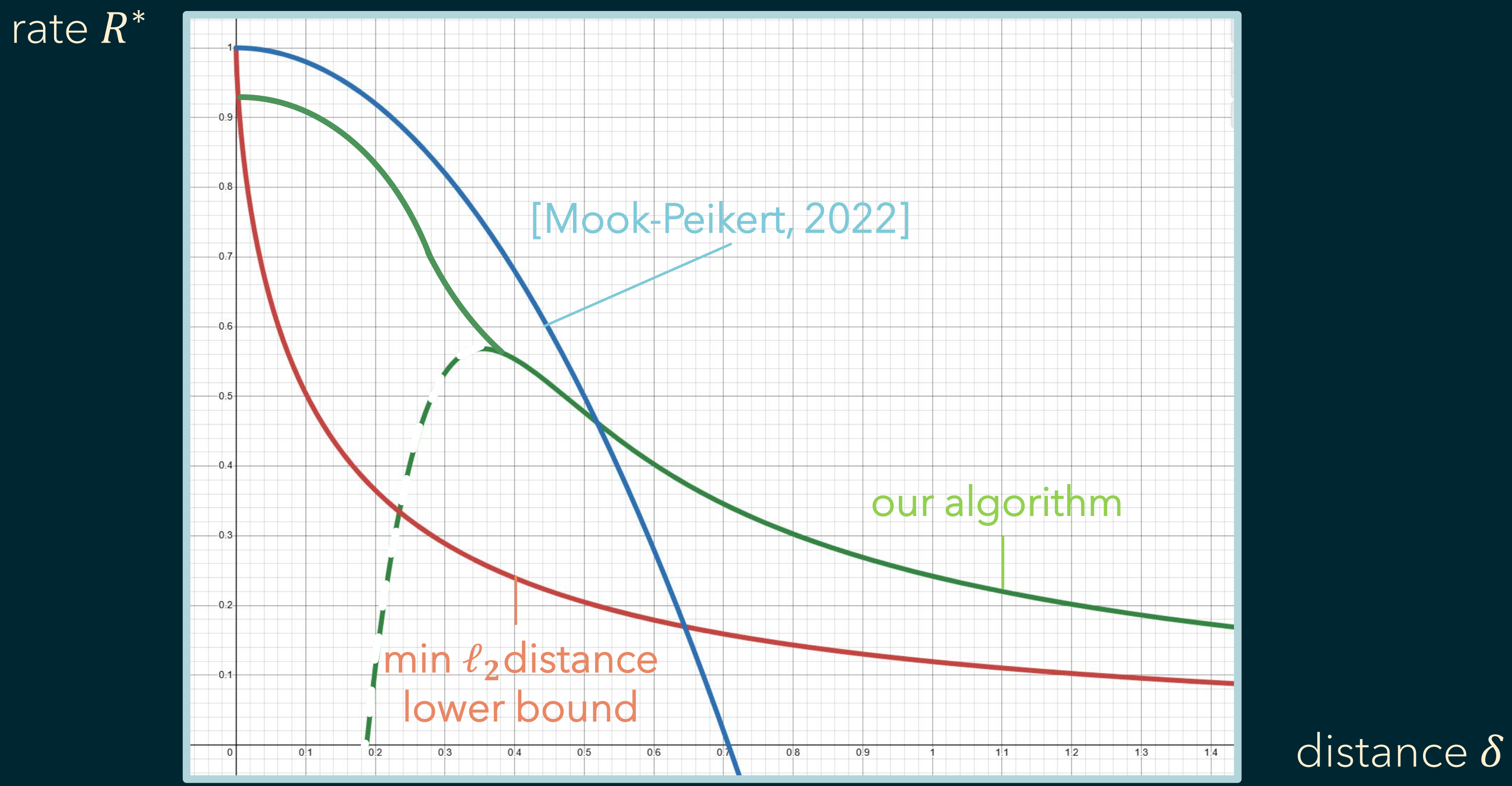
Theorem: For any $0 < p \leq 2$, prime q, and $\delta > 0$, the GS soft-decision algorithm using weight vectors defined by $f_s^{(p)}$ for any $s > 0$, list-decodes up to ℓ_p distance $d = \delta \cdot n^{1/p}$ any GRS code $\mathcal{C} \subseteq \mathbb{F}_q^n$ with adjusted rate

$$R^* < \frac{f_s(\delta)^2}{f_s(\mathcal{L}_q)} =: B_{q,s,\delta}^{(p)} \xrightarrow{s,q/s \to \infty} \frac{1}{\delta \cdot c_p (e \cdot p)^{1/p}}$$

This is the (dimension-normalized) volume of the n-dim. ℓ_p ball of radius $n^{1/p}$!

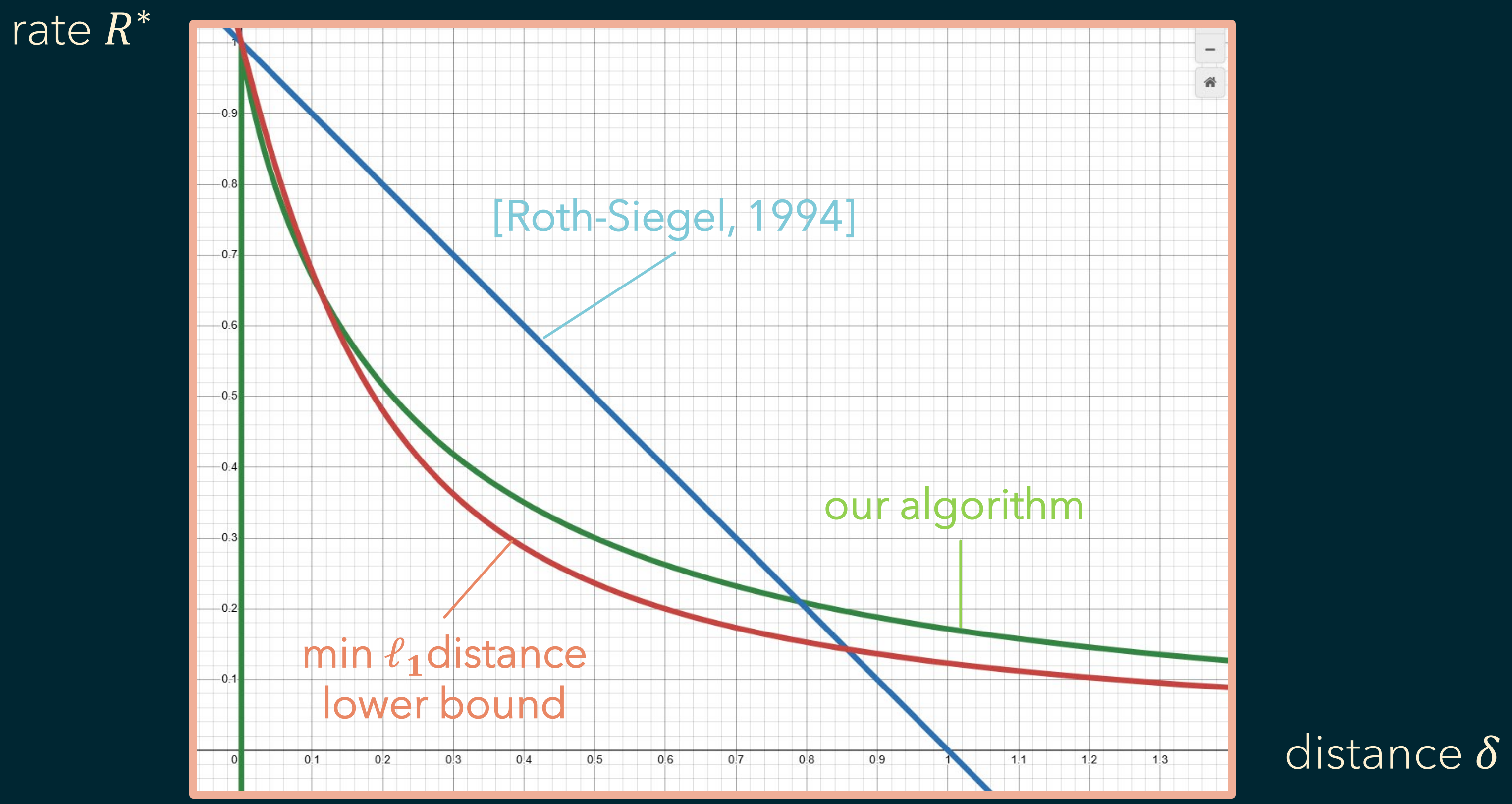
in time $\mathrm{poly}(n, q, \exp(1/s^p)/(B_{q,s,\delta}^{(p)} - \sqrt{R^*}))$.

Comparison to Prior Algorithms



Rate-distance trade-off for ℓ_2

Comparison to Prior Algorithms



Rate-distance trade-off for ℓ_1

Open Directions

- Determine the optimal choice of weights for the GS algorithm for $\delta > 1/2$ for ℓ_2 norm.

  For $\delta < 1/2$, [Mook-Peikert, 2022] proved their weight vector is optimal.

- The product of the rate R^* and distance δ for which our algorithm works approaches

  $R^* \cdot \delta \to 1$ / volume of the n-dim. ℓ_p ball of radius $n^{1/p}$ (dim.-normalized).

  Why should this be the case?

- What is the list-decoding capacity for decoding over general ℓ_p norms?

  How do our algorithmic bounds compare?

Thank you to my collaborators!

Questions?