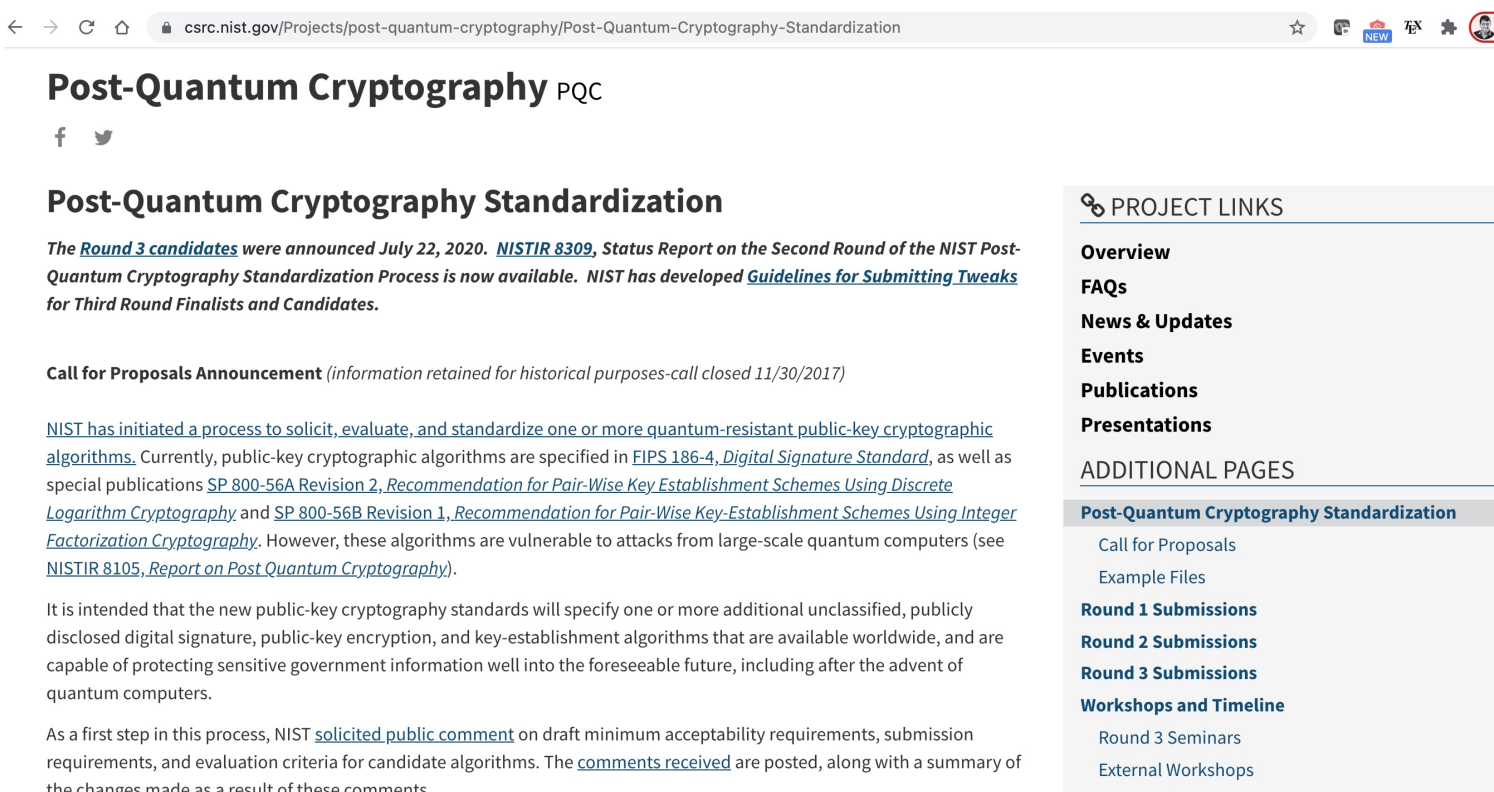
MIT 6.875

Foundations of Cryptography
Lecture 18

TODAY (and next few lectures): Lattice-based Cryptography

Why Lattice-based Crypto?

- ☐ **Exponentially Hard** (so far)
- ☐ **Quantum-Resistant** (so far)



csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization

Post-Quantum Cryptography PQC

Post-Quantum Cryptography Standardization

The Round 3 candidates were announced July 22, 2020. NISTIR 8309, Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process is now available. NIST has developed Guidelines for Submitting Tweaks for Third Round Finalists and Candidates.

Call for Proposals Announcement *(information retained for historical purposes-call closed 11/30/2017)*

NIST has initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms. Currently, public-key cryptographic algorithms are specified in FIPS 186-4, *Digital Signature Standard*, as well as special publications SP 800-56A Revision 2, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography* and SP 800-56B Revision 1, *Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography*. However, these algorithms are vulnerable to attacks from large-scale quantum computers (see NISTIR 8105, *Report on Post Quantum Cryptography*).

It is intended that the new public-key cryptography standards will specify one or more additional unclassified, publicly disclosed digital signature, public-key encryption, and key-establishment algorithms that are available worldwide, and are capable of protecting sensitive government information well into the foreseeable future, including after the advent of quantum computers.

As a first step in this process, NIST solicited public comment on draft minimum acceptability requirements, submission requirements, and evaluation criteria for candidate algorithms. The comments received are posted, along with a summary of the changes made as a result of these comments.

PROJECT LINKS

- **Overview**
- **FAQs**
- **News & Updates**
- **Events**
- **Publications**
- **Presentations**

ADDITIONAL PAGES

- **Post-Quantum Cryptography Standardization**
  - Call for Proposals
  - Example Files
- **Round 1 Submissions**
- **Round 2 Submissions**
- **Round 3 Submissions**
- **Workshops and Timeline**
  - Round 3 Seminars
  - External Workshops

Why Lattice-based Crypto?

- ☐ **Exponentially Hard** (so far)
- ☐ **Quantum-Resistant** (so far)
- ☐ **Worst-case hardness**

  (unique feature of lattice-based crypto)
- ☐ **Simple and Efficient**
- ☐ **Enabler of Surprising Capabilities**

  (Fully Homomorphic Encryption)

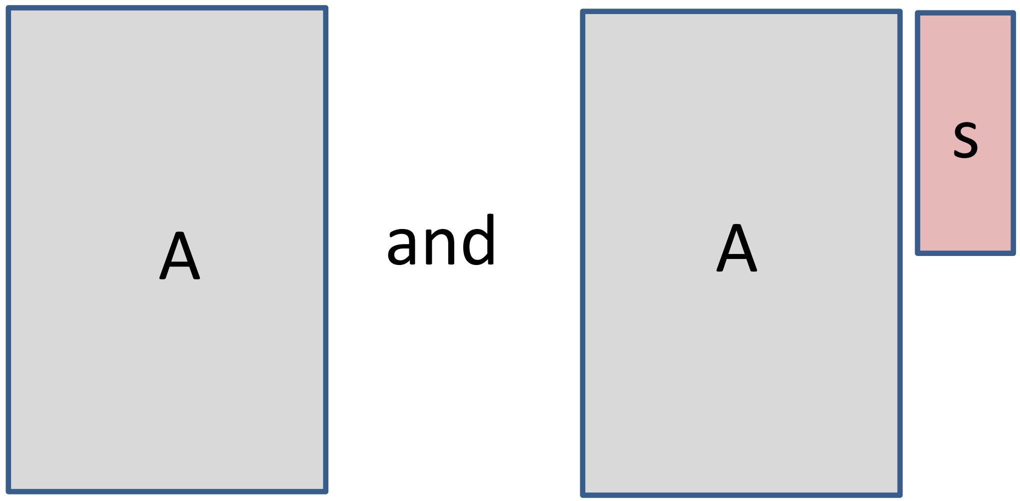
Solving Linear Equations

$$\begin{aligned} 5s_1 + 11s_2 &= 2 \\ 2s_1 + s_2 &= 6 \\ 7s_1 + s_2 &= 26 \end{aligned}$$

where all equations are over $\mathbb{Z}$, the integers

Solving Linear Equations

Given:

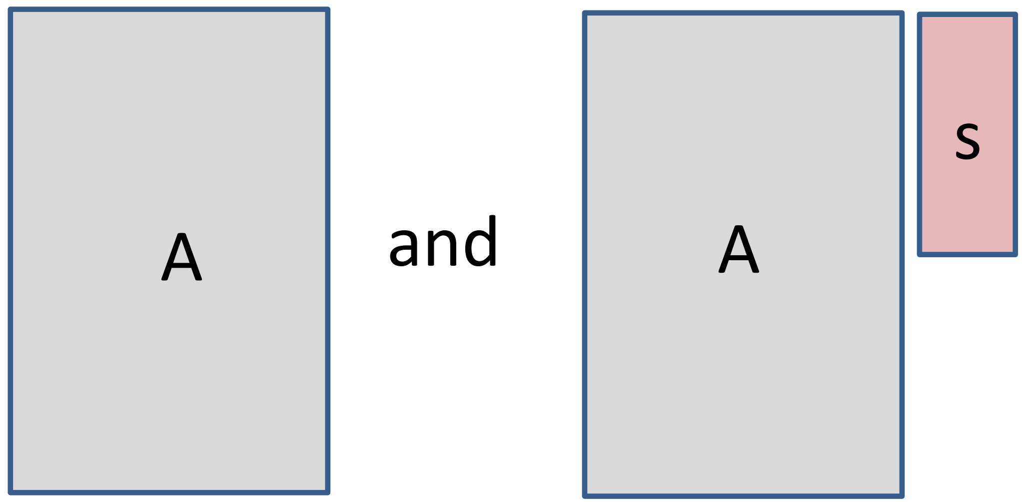


GOAL: Find s.

More generally, n variables and $m \gg n$ equations.

Solving Linear Equations

Given:

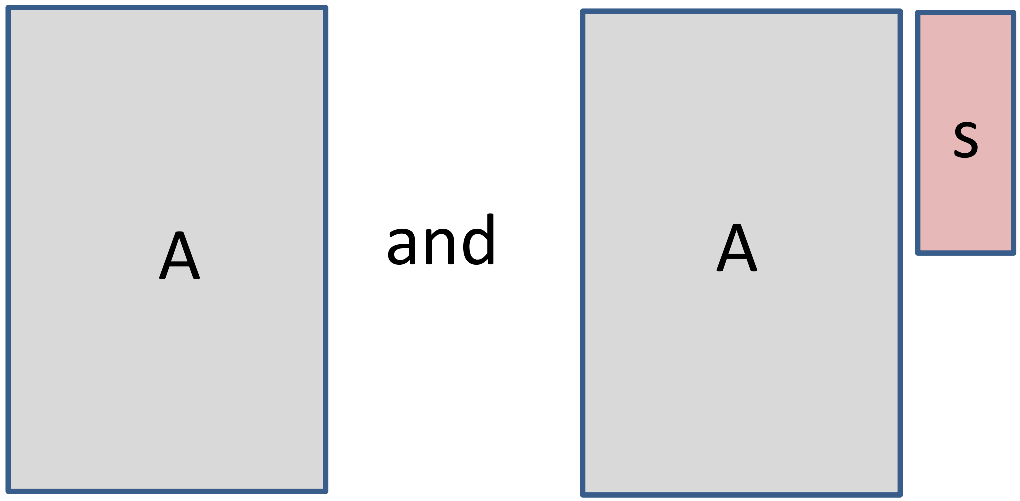


GOAL: Find s.

EASY! For example, by Gaussian Elimination

Solving Linear Equations

Given:



GOAL: Find s.

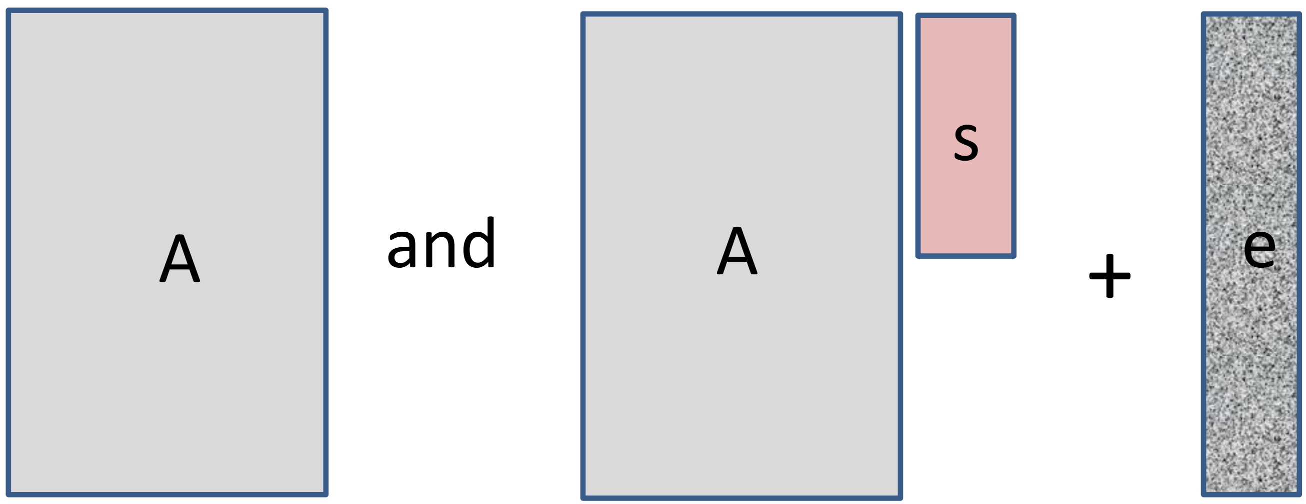
How to make it hard: Chop the head?

That is, work modulo some q. $(1121 \; mod \; 100 = 21)$

Still EASY! Gaussian Elimination mod q

Solving Linear Equations

Given:



GOAL: Find s.

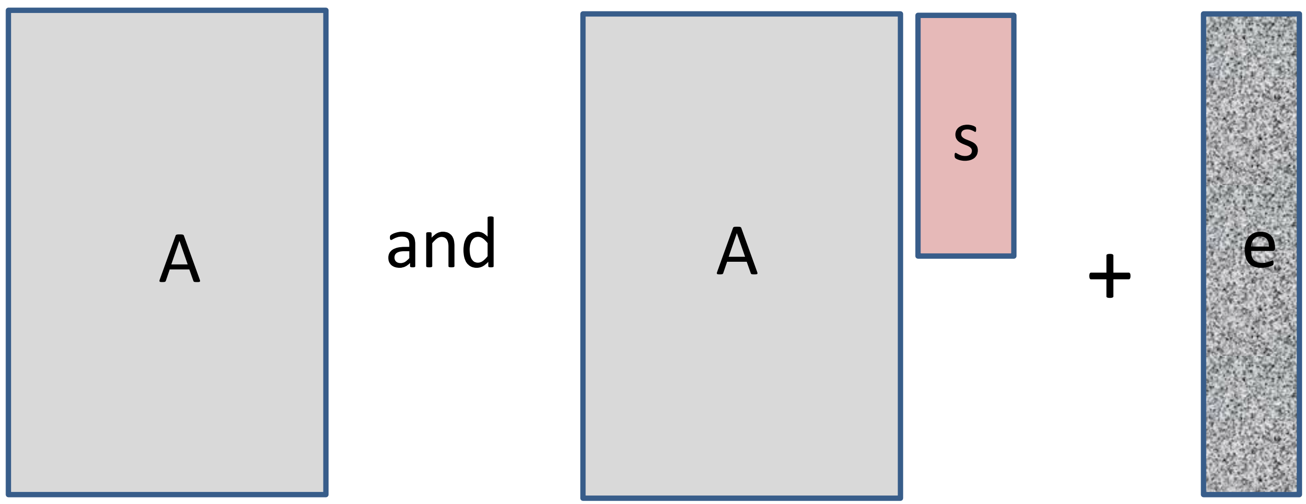
How to make it hard: Chop the tail?

Add a small error to each equation.

Still EASY! Linear regression.

Solving Linear Equations

Given:



GOAL: Find s.

How to make it hard: Chop the head *and* the tail?

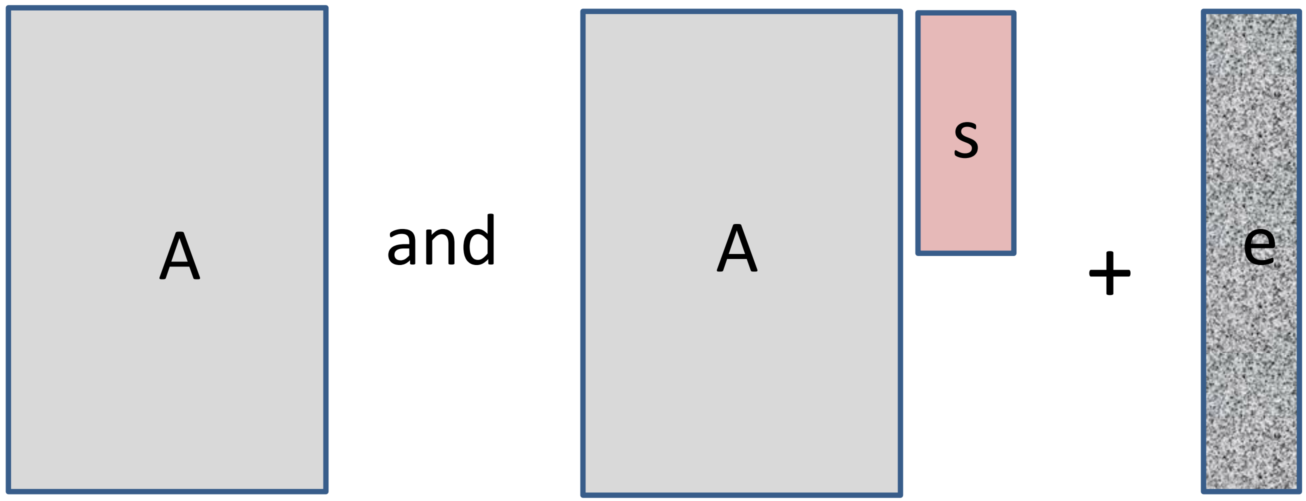
Add a small error to each equation and work mod q.

Turns out to be very HARD!

Solving Noisy Modular Linear Equations

Learning with Errors (LWE)

Given:



GOAL: Find s.

Parameters: dimensions $\boldsymbol{n}$ and m, modulus $\boldsymbol{q}$, error distribution χ = uniform in some interval $[-\boldsymbol{B}, \ldots, \boldsymbol{B}]$.

A is chosen at random from $\mathbb{Z}_q^{m \times n}$, **s** from $\mathbb{Z}_q^n$ and **e** from χ^m.

Learning with Errors (LWE)

=

- **Decoding Random Linear Codes**

  (over F_q with L_1 errors)

- **Learning Noisy Linear Functions**

- **Worst-case hard Lattice Problems**

  [Regev'05, Peikert'09]

Attack 1: *Linearization*

Given $\boldsymbol{A}, \boldsymbol{As} + \boldsymbol{e}$, find $\boldsymbol{s}$.

Idea (a) *Each noisy linear equation is an exact polynomial eqn.*

Consider $b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e = \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i s_i + e$.

Imagine for now that the error bound $B = 1$. So, $e \in \{-1,0,1\}$. In other words, $\mathrm{b} - \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i s_i \in \{-1,0,1\}$.

So, here is a noiseless polynomial equation on s_i:

$$(\mathrm{b} - \textstyle\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i s_i - 1)\,(\mathrm{b} - \textstyle\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i s_i)(\mathrm{b} - \textstyle\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} a_i s_i + 1) = 0$$

Attack 1: *Linearization*

Given $\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}$, find $\boldsymbol{s}$.

BUT: *Solving (even degree 2) polynomial equations is NP-hard.*

$$(\mathrm{b}-\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} a_i s_i-1)\,(\mathrm{b}-\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} a_i s_i)(\mathrm{b}-\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} a_i s_i+1)=0$$

Attack 1: *Linearization*

$$(\mathrm{b} - \textstyle\sum_{i=1}^{n} a_i s_i - 1)\,(\mathrm{b} - \sum_{i=1}^{n} a_i s_i)(\mathrm{b} - \sum_{i=1}^{n} a_i s_i + 1) = 0$$

Idea (b) *Easy to solve given <u>sufficiently many</u> equations.*

(using a technique called '

$$\sum a_{ijk} s_i s_j s_k + \sum a_{ij} s_i s_j + \sum a_{i}$$

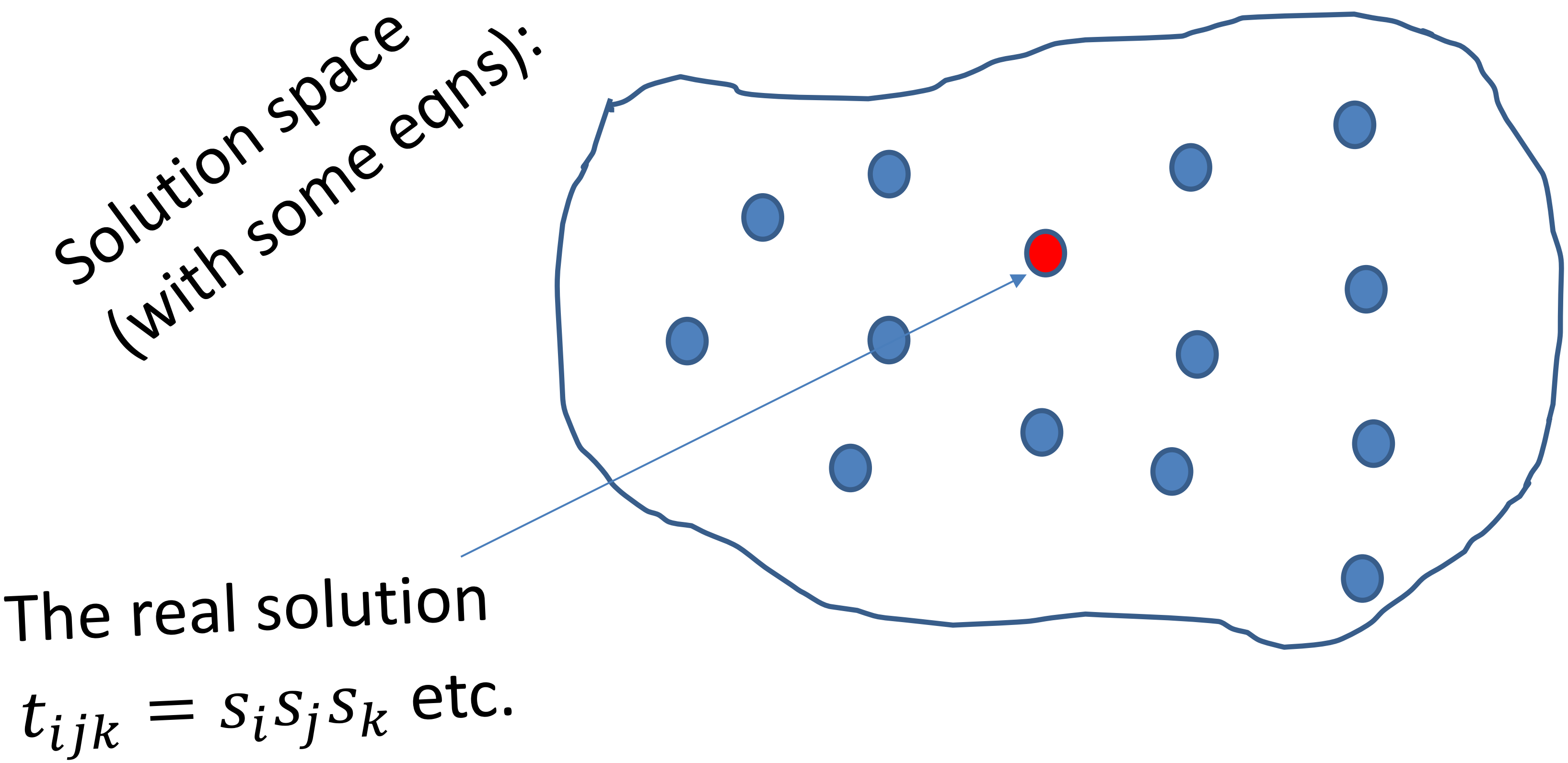
Treat each "monomial", e.g. $\mathrm{s_i s}$

variable, e.g. $\mathrm{t_{ijk}}$.

Now, you have a noiseless linear equation in $\mathrm{t_{ijk}}$!!!

Attack 1: *Linearization*

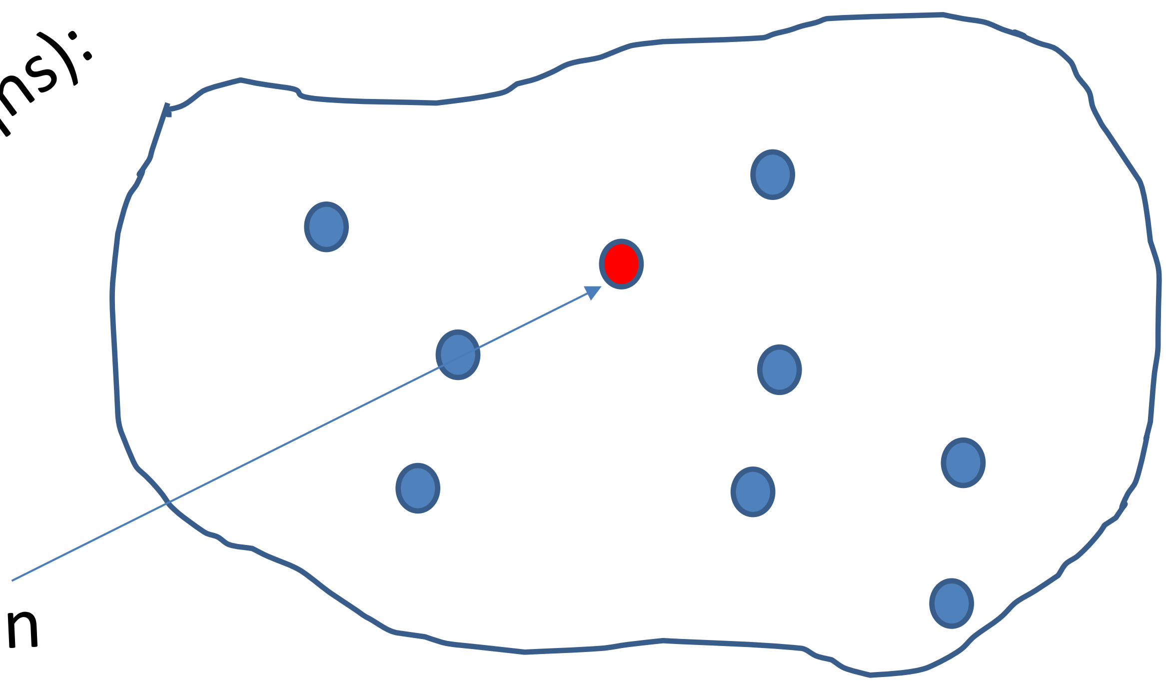
$$\sum a_{ijk} t_{ijk} + \sum a_{ij} t_{ij} + \sum a_i t_i + (b-1)b(b+1) = 0$$



Attack 1: *Linearization*

$$\sum a_{ijk}t_{ijk} + \sum a_{ij}t_{ij} + \sum a_i t_i + (b-1)b(b+1) = 0$$

Solution space
(with more eqns):

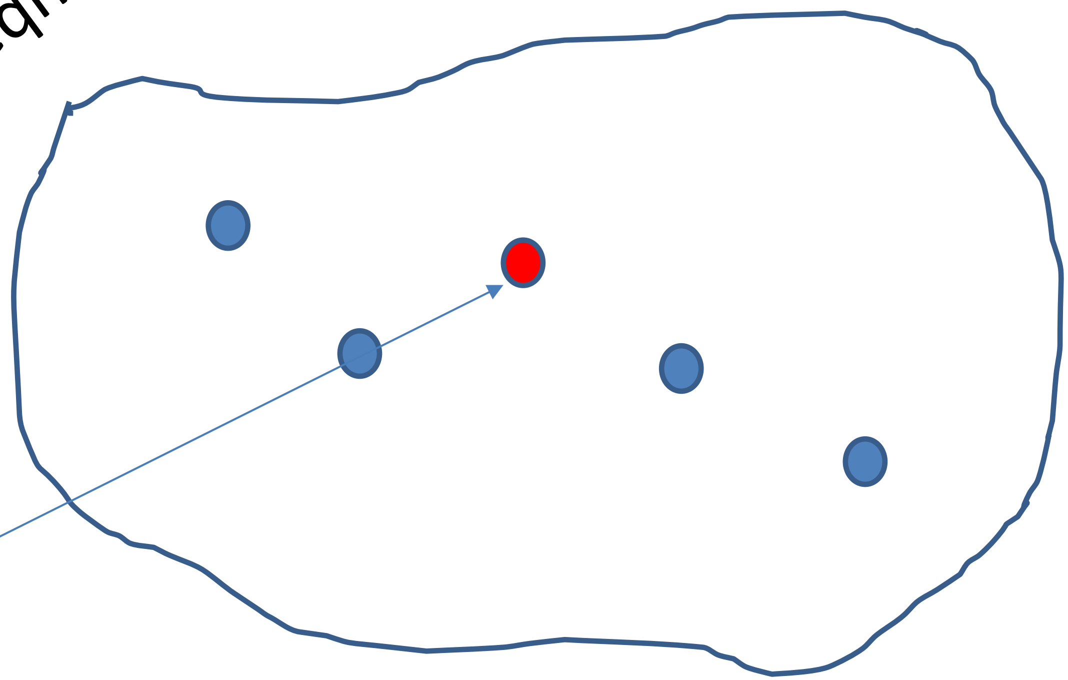


The real solution
$t_{ijk} = s_i s_j s_k$ etc.

Attack 1: *Linearization*

$$\sum a_{ijk} t_{ijk} + \sum a_{ij} t_{ij} + \sum a_i t_i + (b-1)b(b+1) = 0$$

Solution space
(with even more eqns):

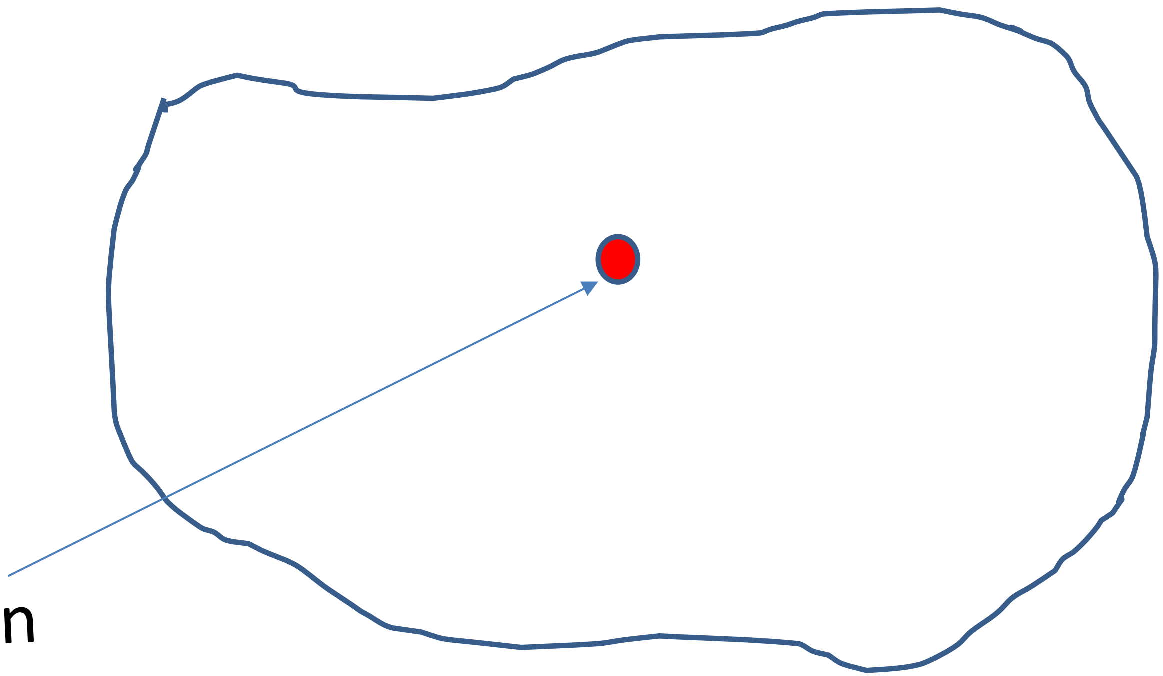


The real solution
$t_{ijk} = s_i s_j s_k$ etc.

Attack 1: *Linearization*

$$\sum a_{ijk} t_{ijk} + \sum a_{ij} t_{ij} + \sum a_i t_i + (b-1)b(b+1) = 0$$

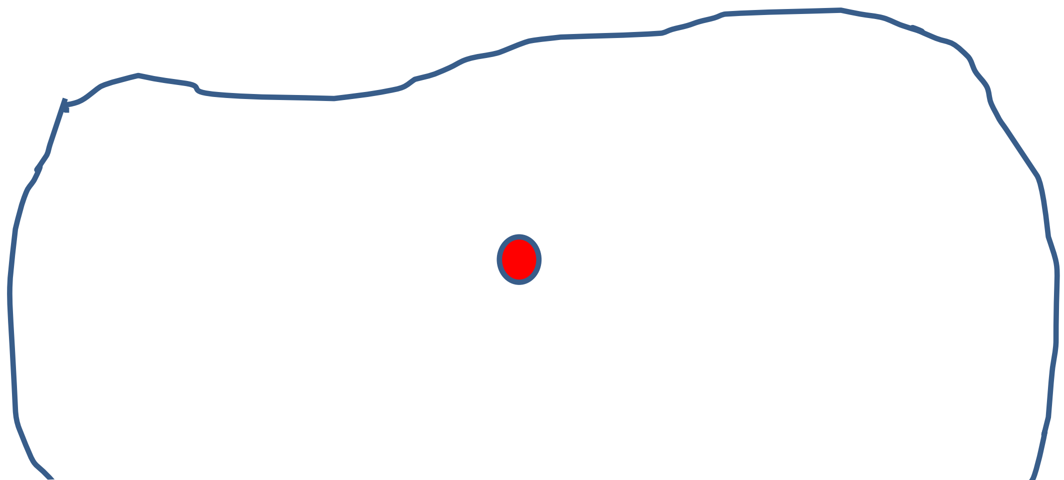
Solution space
(keep going):



The real solution
$t_{ijk} = s_i s_j s_k$ etc.

Attack 1: *Linearization*

$$\sum a_{ijk}t_{ijk} + \sum a_{ij}t_{ij} + \sum a_i t_i + (b-1)b(b+1) = 0$$



When #eqns = #vars $\approx O(n^3)$
the only surviving solution to the linear system is the real solution.

Attack 1: *Linearization*

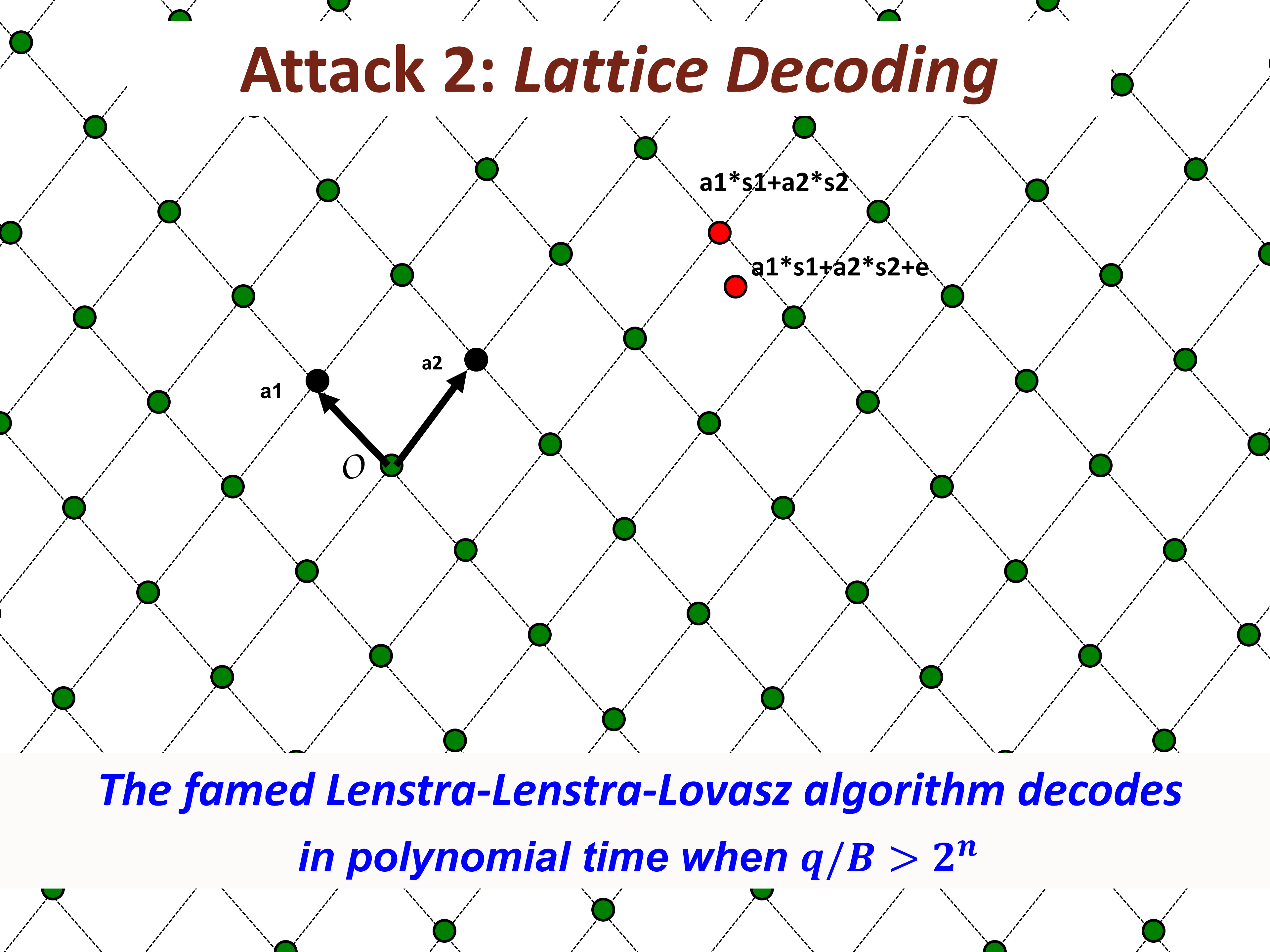
Given $\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$, find $\boldsymbol{s}$.

Can solve/break as long as

$$\boldsymbol{m} \gg \boldsymbol{n}^{\boldsymbol{2B+1}}$$

We will set $B = n^{\Omega(1)}$, in other words polynomial in n so as to blunt this attack.

Attack 2: *Lattice Decoding*



The famed Lenstra-Lenstra-Lovasz algorithm decodes in polynomial time when $q/B > 2^n$

Setting Parameters

Put together, we are safe with:

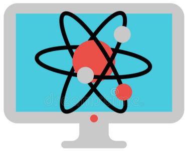
$$n = \text{security parameter } (\approx 1 - 10\text{K})$$

$$m = \text{arbitrary poly in } n$$

$$B = \text{small poly in } n, \text{say } \sqrt{n}$$

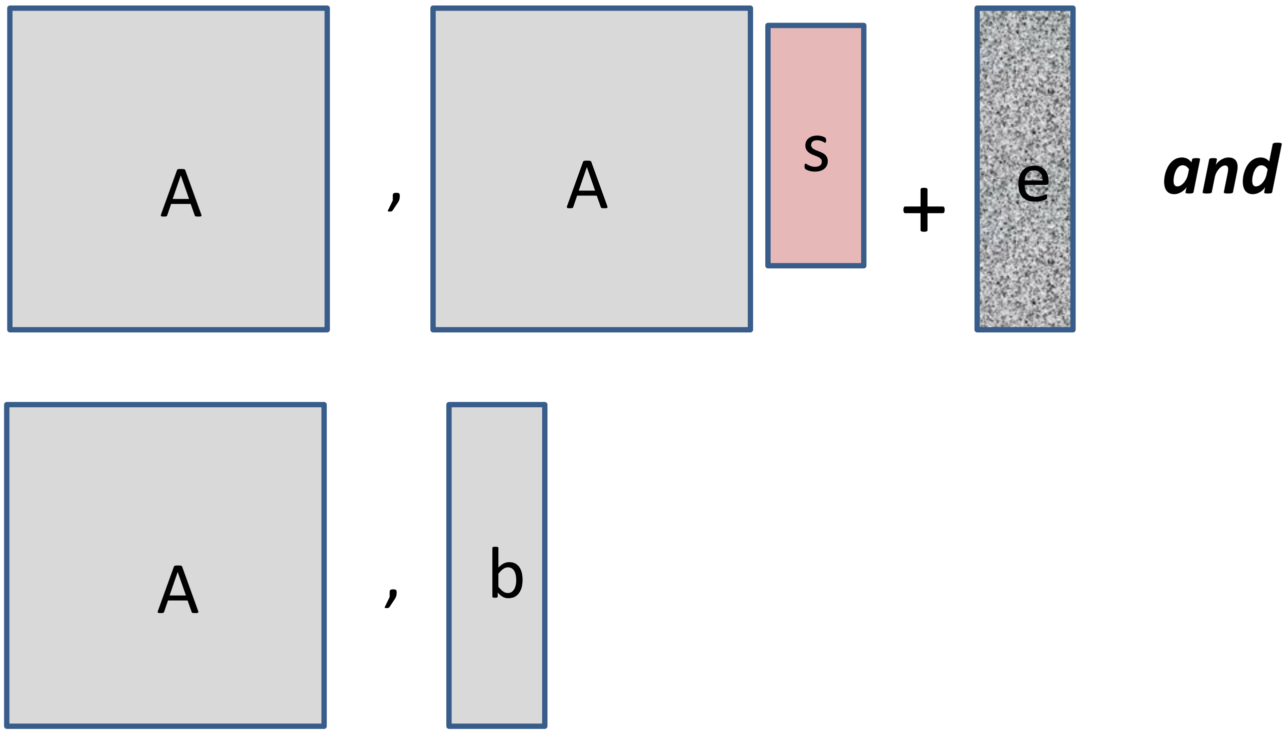
$q =$ poly in n, larger than B, and could be as large as sub-exponential, *say* $2^{n^{0.99}}$

even from quantum computers, AFAWK!



Decisional LWE

Can you distinguish between:



Theorem: "Decisional LWE is as hard as LWE".

OWF and PRG

$$g_A(s,e) = \mathbf{As}+e$$

($\mathbf{A} \in Z_q^{nXm}$
$\mathbf{s} \in Z_q^n$ random "small" secret vector
$\boldsymbol{e} \in Z_q^n$: random "small" error vector)

- g_A is a one-way function (assuming LWE)
- g_A is a pseudo-random generator (decisional LWE)
- g_A is also a trapdoor function…
- also a homomorphic commitment…

Basic (Secret-key) Encryption

[Regev05]

n = security parameter, q = "small" modulus

- Secret key sk = Uniformly random vector **s** $\in Z_q^n$
- Encryption Enc$_{\mathbf{s}}(\mu)$: // $\mu \in \{0,1\}$
  - Sample uniformly random **a** $\in Z_q^n$, "small" noise e $\in Z$
  - The ciphertext **c = (a**, b = ⟨**a, s**⟩ + e + μ)
- Decryption Dec$_{sk}$(**c**): Output (b − ⟨**a, s**⟩ mod q)

  // correctness as long as |e| < q/4

Basic (Secret-key) Encryption

[Regev05]

This scheme is additively homomorphic.

$\boldsymbol{c} = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + \mu \lfloor q/2 \rceil)$ $+$ Enc$_{\mathbf{s}}$(m)

$\boldsymbol{c}' = (\mathbf{a}', b' = \langle \mathbf{a}', \mathbf{s} \rangle + e' + \mu' \lfloor q/2 \rceil)$ Enc$_{\mathbf{s}}$(m')

$\boldsymbol{c} + \boldsymbol{c}' = (\mathbf{a}+\mathbf{a}', b+b') = \langle \mathbf{a}+\mathbf{a}', \mathbf{s} \rangle + (e+e') + (\mu + \mu') \lfloor q/2 \rceil)$

In words: $c + c'$ is an encryption of $\mu + \mu'$ (mod 2)

Basic (Secret-key) Encryption

[Regev05]

You can also negate the encrypted bit easily.

We will see how to make this scheme into a fully homomorphic scheme (in the next few lectures)

For now, note that the error increases when you add two ciphertexts. That is, $|e_{add}| \approx |e_1| + |e_2| \leq 2B$.

Setting $q = n^{\log n}$ and $B = \sqrt{n}$ (for example) lets us support any polynomial number of additions.

Next Lecture:
Fully Homomorphic Encryption