MIT 6.875 & Berkeley CS276

Foundations of Cryptography
Lecture 20

TODAY: Lattice-based Cryptography

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**

  (computing on encrypted data)

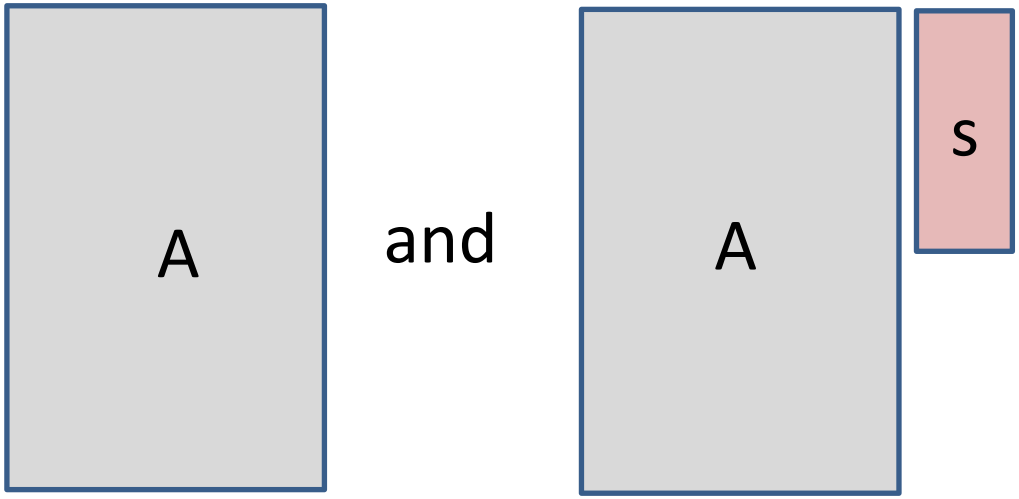
Solving Linear Equations

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

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

Solving Linear Equations

Given:

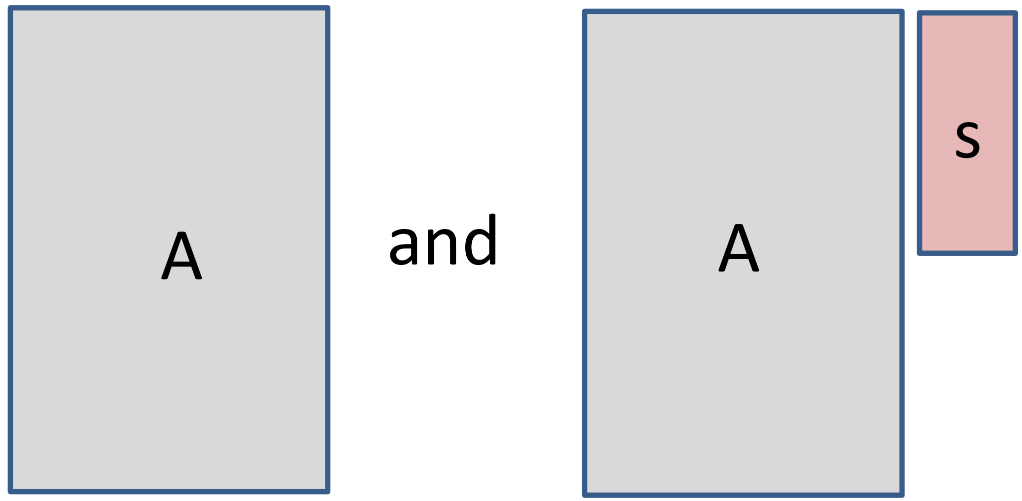


GOAL: Find s.

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

Solving Linear Equations

Given:

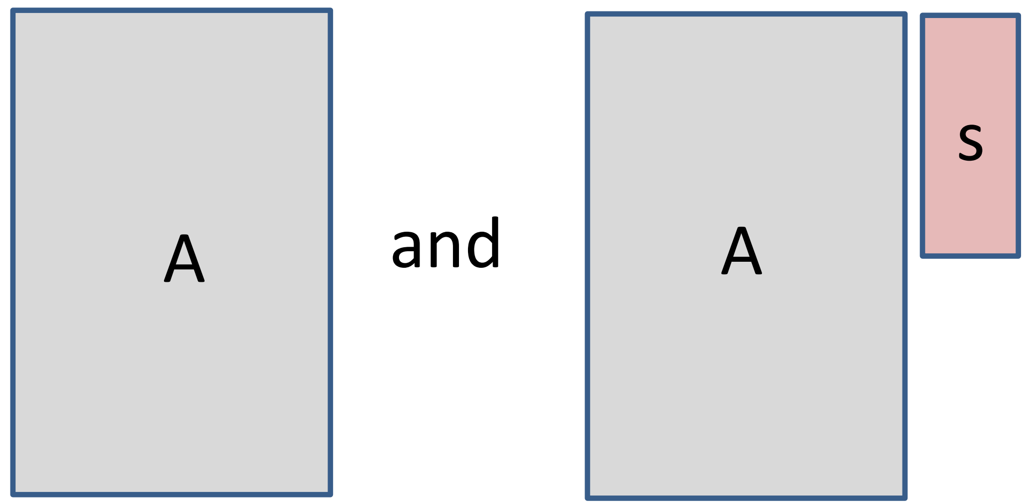


GOAL: Find s.

EASY! For example, by Gaussian Elimination

Solving Linear Equations

Given:



GOAL: Find s.

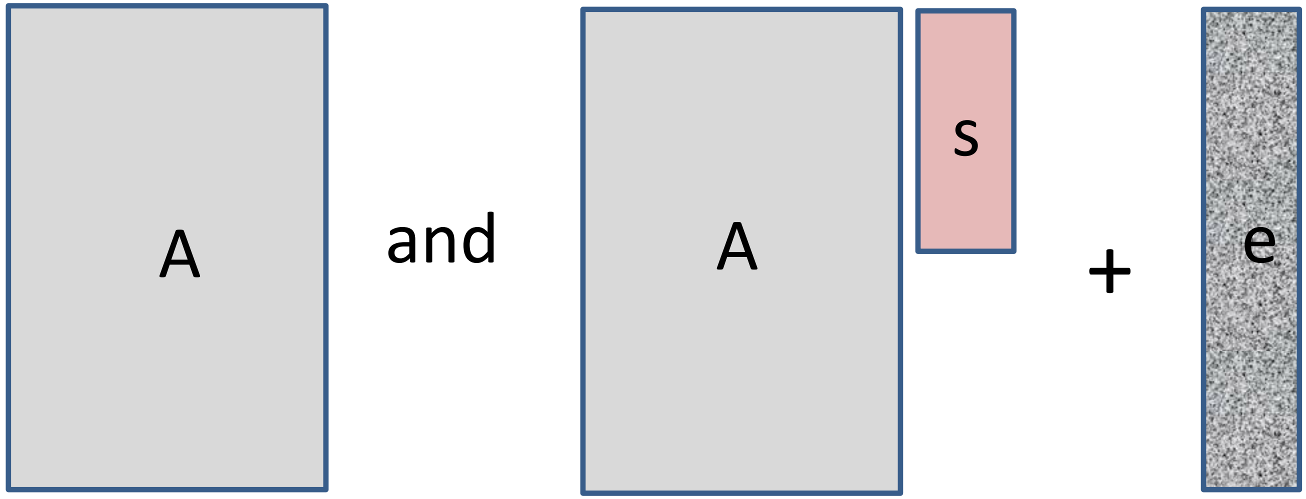
How to make it hard: Chop the head?

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

Still EASY! Gaussian Elimination mod q

Solving Linear Equations

Given:



GOAL: Find s.

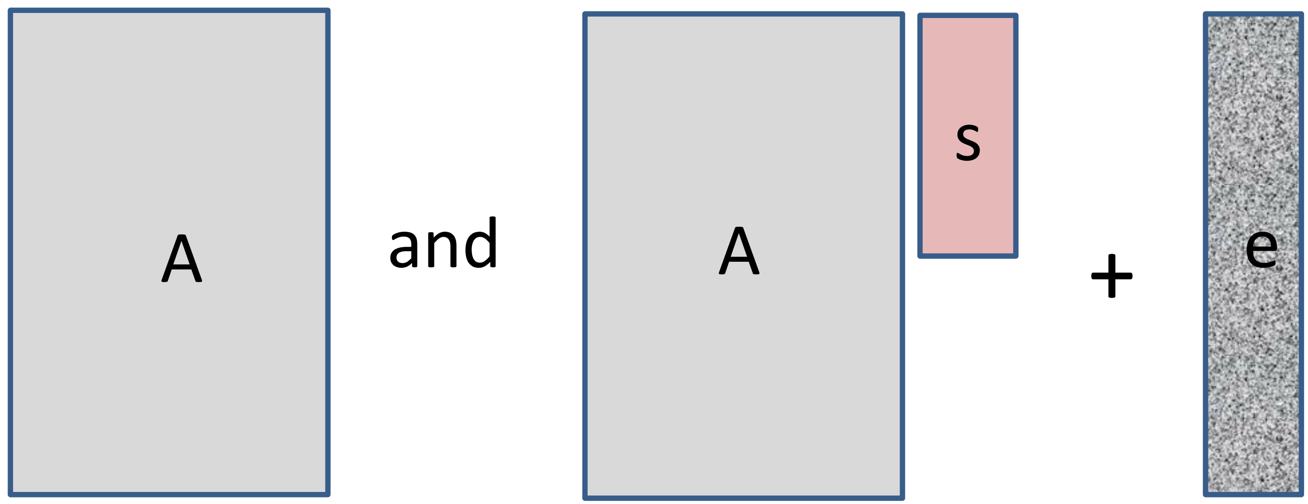
How to make it hard: Chop the tail?

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?

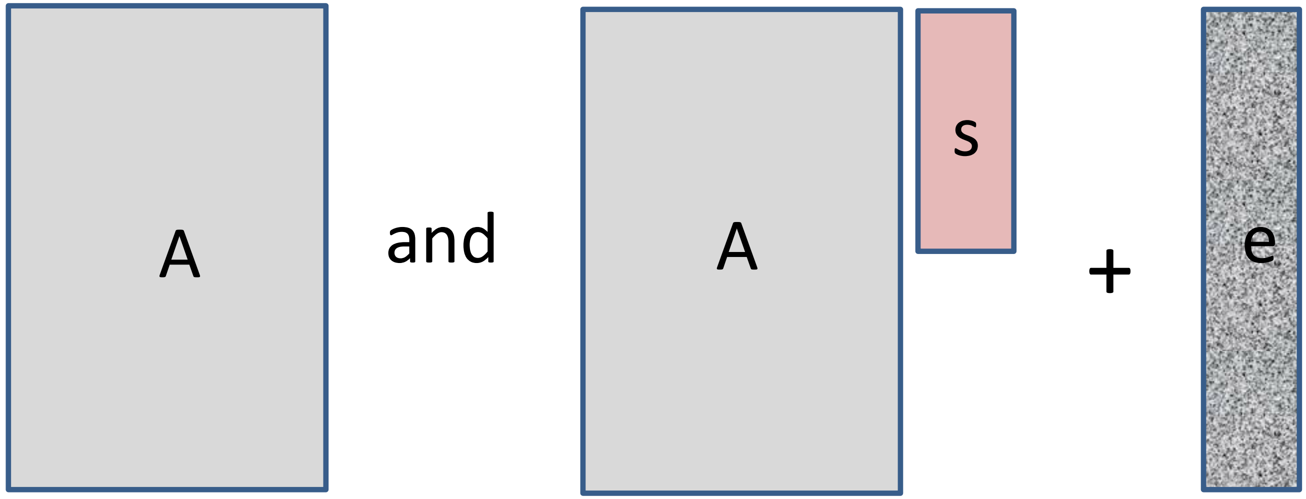
Add a small error to each equation and work mod q.

Turns out to be very HARD!

Learning with Errors (LWE)

Solving Modular Linear Equations

Given:



GOAL: Find s.

<u>Parameters</u>: 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*

<u>Given $\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$, find $\boldsymbol{s}$</u>.

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

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

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

So, here is a noiseless polynomial equation on s_i:

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

Attack 1: *Linearization*

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

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

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

Attack 1: *Linearization*

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

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. $\mathrm{s_i s}$

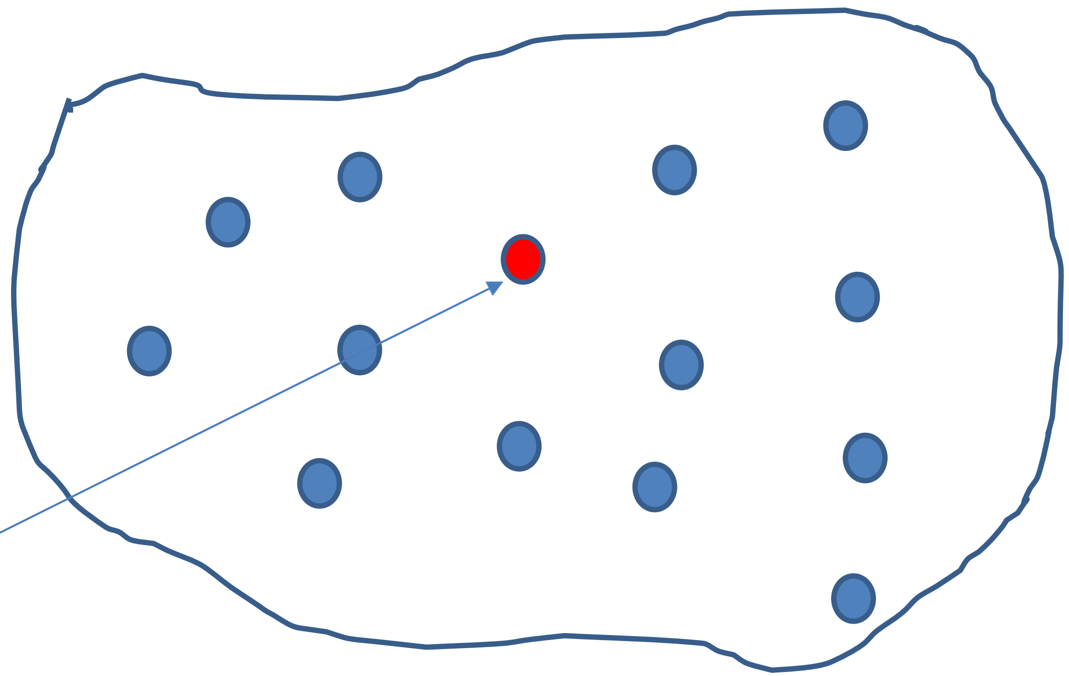
variable, e.g. $\mathrm{t_{ijk}}$.

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

Attack 1: *Linearization*

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

Solution space (with some eqns):



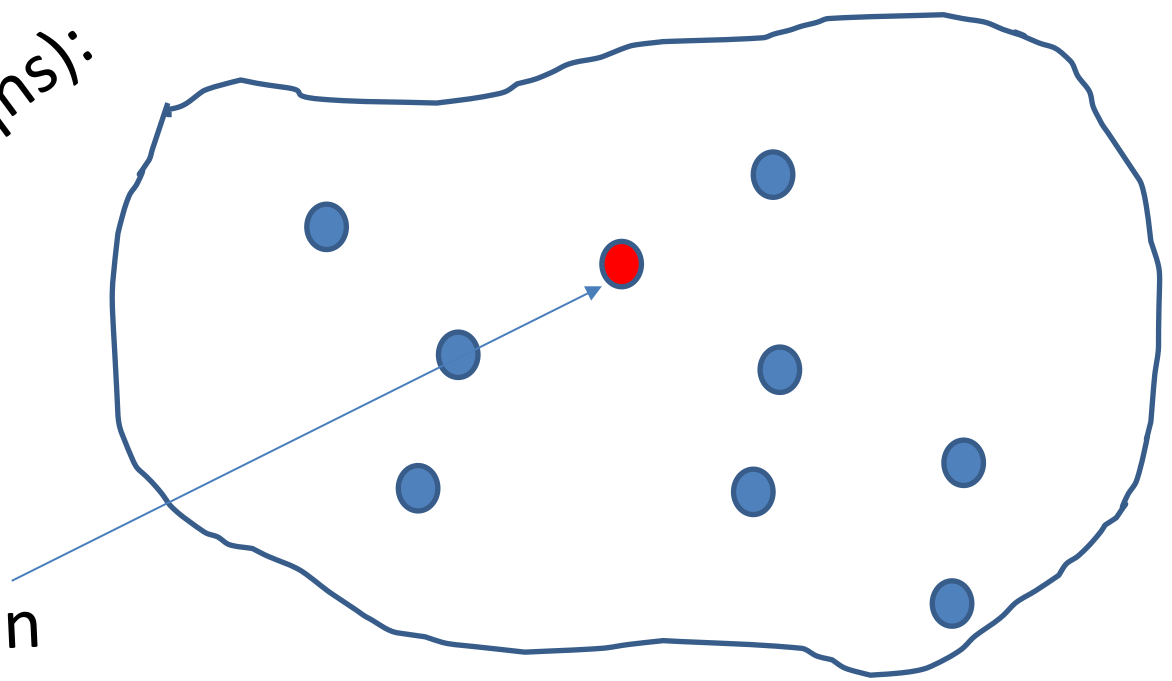
The real solution

$t_{ijk} = s_i s_j s_k$ 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 more eqns):

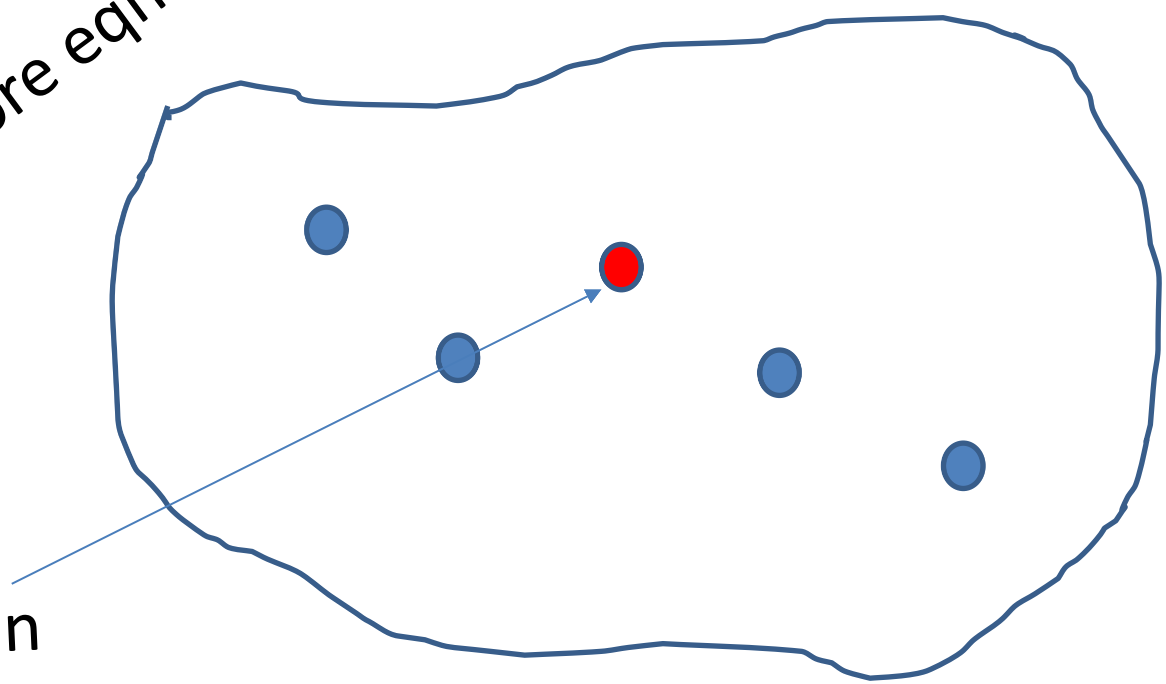


The real solution $t_{ijk} = s_i s_j s_k$ etc.

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):

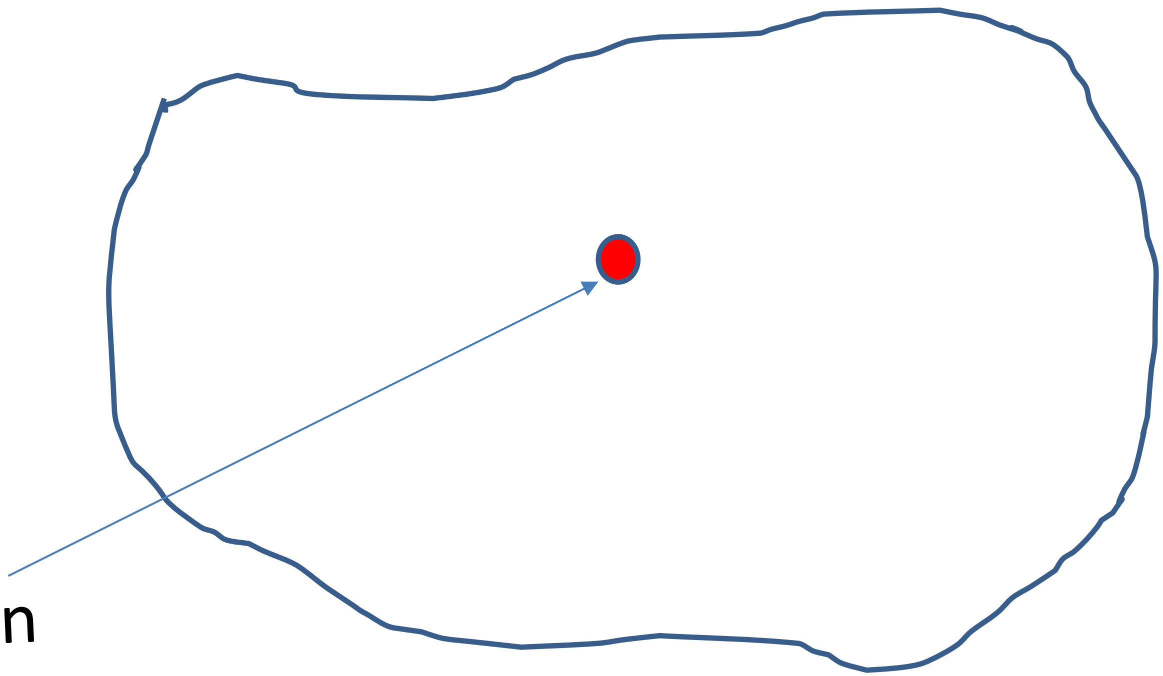


The real solution
$t_{ijk} = s_i s_j s_k$ etc.

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):

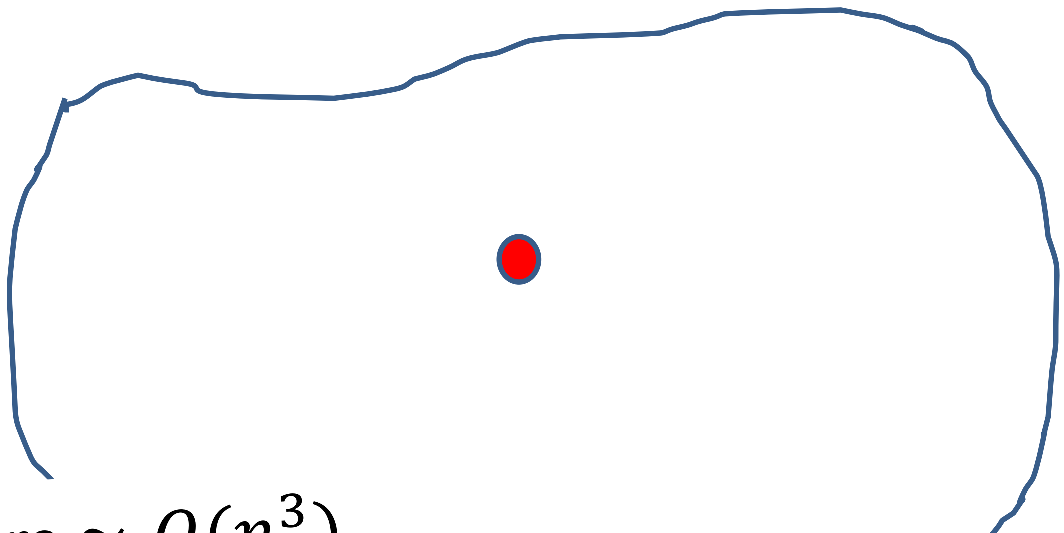


The real solution

$t_{ijk} = s_i s_j s_k$ 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*

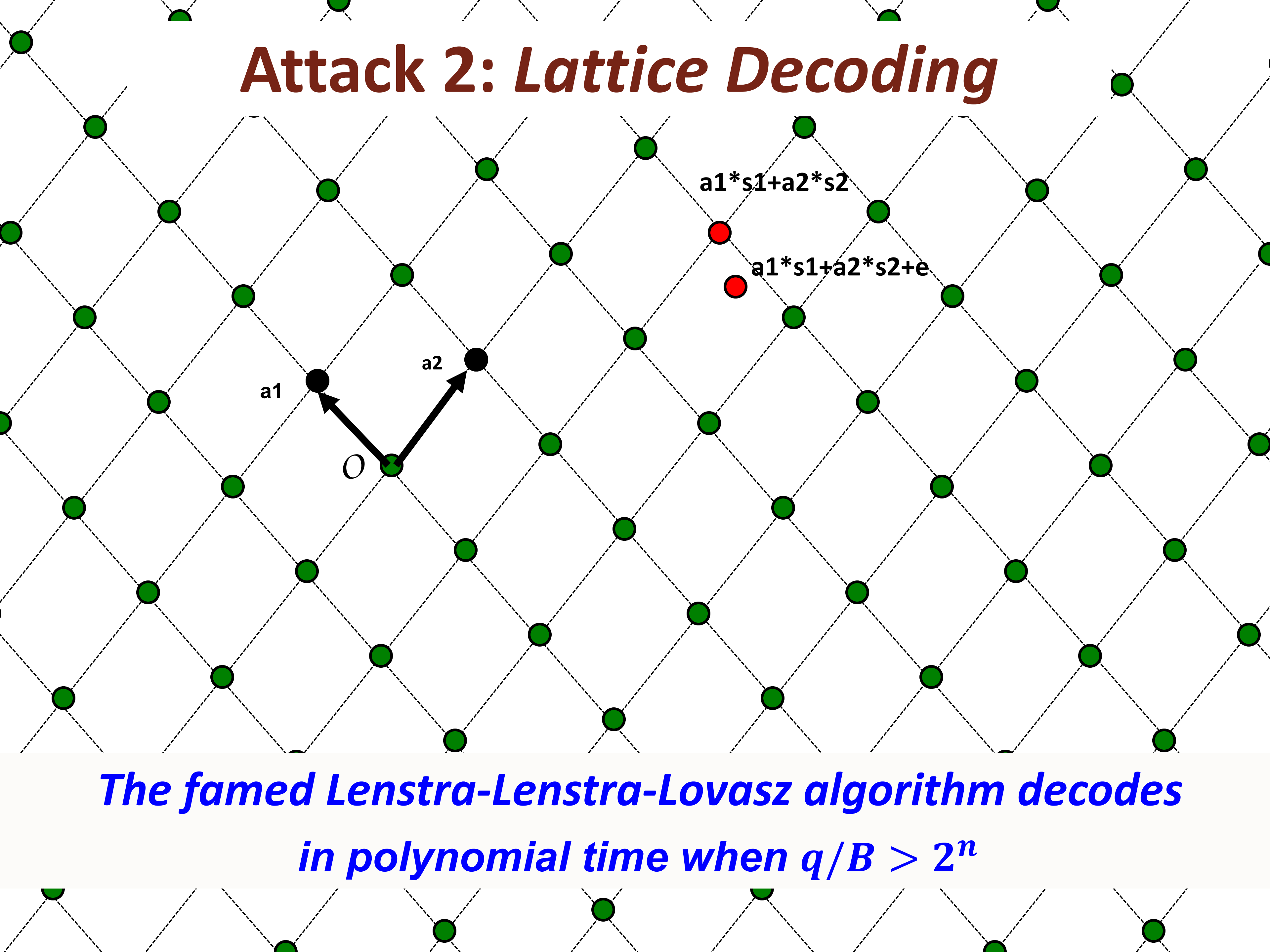
Given $\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}$, find $\boldsymbol{s}$.

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:

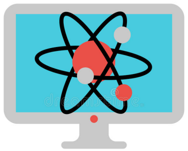
$n =$ security parameter ($\approx 1 - 10\text{K}$)

$m =$ arbitrary poly in n

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

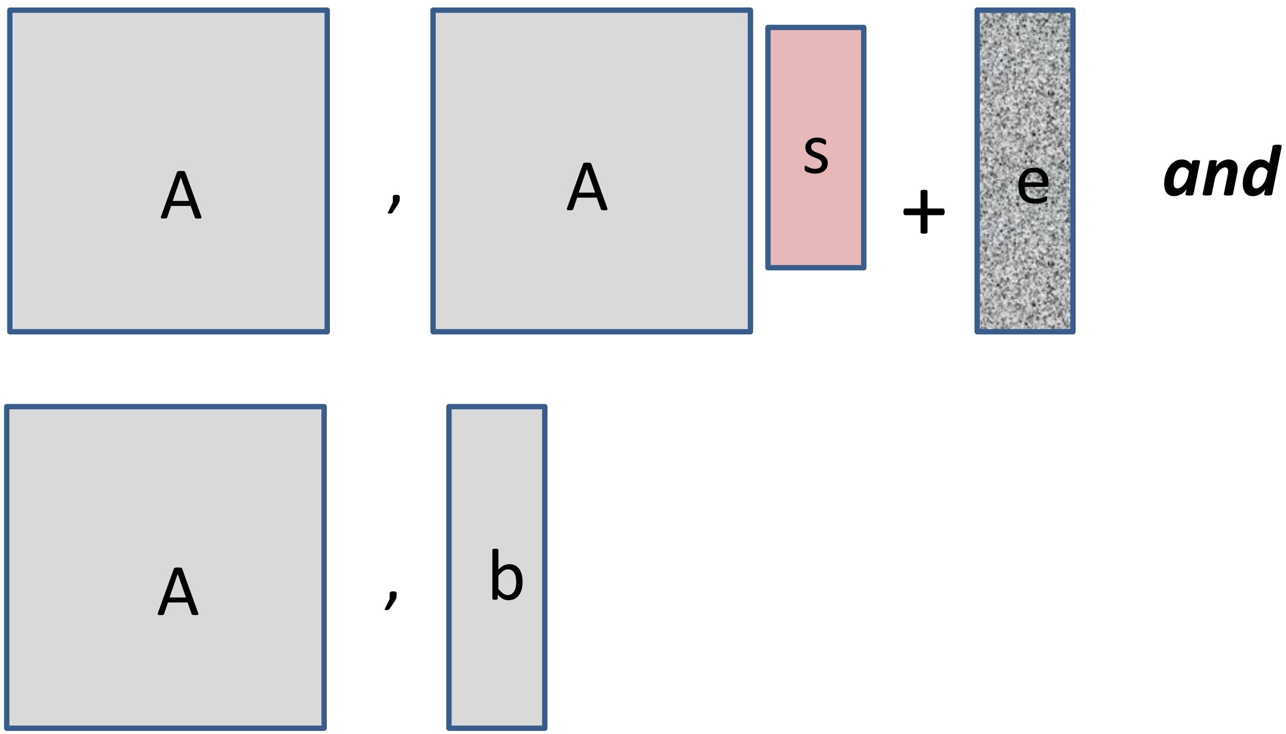
$q =$ poly in n, larger than B, and could be as large as sub-exponential, *say* $2^{n^{0.99}}$

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 +$\mu \lfloor q/2 \rfloor$**)**
- Decryption Dec$_{sk}$(**c**): Output Round$_{q/2}$(b − ⟨**a, s**⟩ mod q)

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

Basic (Secret-key) Encryption

[Regev05]

We already saw that this scheme is additively homomorphic.

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

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

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

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 lec)

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.

Public-key Encryption

[Regev05]

- Secret key sk = Uniformly random vector **s** $\in Z_q^n$
- Public key pk: for $i \ from \ 1 \ to \ m = poly(n)$ **TBD**

$$\boldsymbol{c_i} = (\boldsymbol{a_i}, \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i)$$

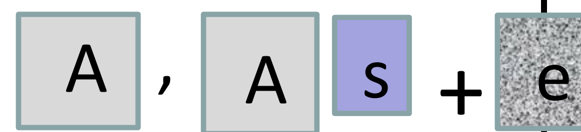
Public-key Encryption

[Regev05]

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$

- Public key pk: for $i\ from\ 1\ to\ m = poly(n)$

$$(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$$



- Encrypting a message bit μ: pick a random vector $\boldsymbol{r} \in \{0,1\}^m$

$$(\boldsymbol{r}\boldsymbol{A}, \boldsymbol{r}\boldsymbol{b} + \mu \lfloor q/2 \rfloor)$$

- Decryption: compute

$$\boldsymbol{r}\boldsymbol{b} + \mu \lfloor q/2 \rfloor - (\boldsymbol{r}\boldsymbol{A})\boldsymbol{s}$$

and round to nearest multiple of q/2.

Correctness

- Encrypting a message bit μ: pick a random vector $\boldsymbol{r} \in \{0,1\}^m$

$$(\boldsymbol{rA}, \boldsymbol{rb} + \mu \lfloor q/2 \rfloor)$$

- Decryption:

$$\boldsymbol{rb} + \mu \lfloor q/2 \rfloor - (\boldsymbol{rA})\mathbf{s} = \boldsymbol{r}(\boldsymbol{As} + \boldsymbol{e}) + \mu \lfloor q/2 \rfloor - (\boldsymbol{rA})\mathbf{s}$$

Decryption works as long as $|\boldsymbol{re}| < \boldsymbol{q}/\mathbf{4}$ or in other words, if the LWE error bound $\mathrm{B} < \boldsymbol{q}/\mathbf{4}\boldsymbol{m} \approx \mathrm{q/poly(n)}$.

Security

Theorem: under decisional LWE, the scheme is IND-secure. In fact, even more: a ciphertext together with the public key is pseudorandom.

We show this by a hybrid argument.

Let's stare at a public key, ciphertext pair.

$$\boldsymbol{pk} = \ (\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{As} + \boldsymbol{e}), \boldsymbol{c} = \boldsymbol{Enc}(\boldsymbol{pk}, \mu) = \boldsymbol{rA}, \boldsymbol{rb} + \mu\, \lfloor q/2 \rfloor)$$

Call this distribution **Hybrid 0**.

Security

Theorem: under decisional LWE, the scheme is IND-secure. In fact, even more: a ciphertext together with the public key is pseudorandom.

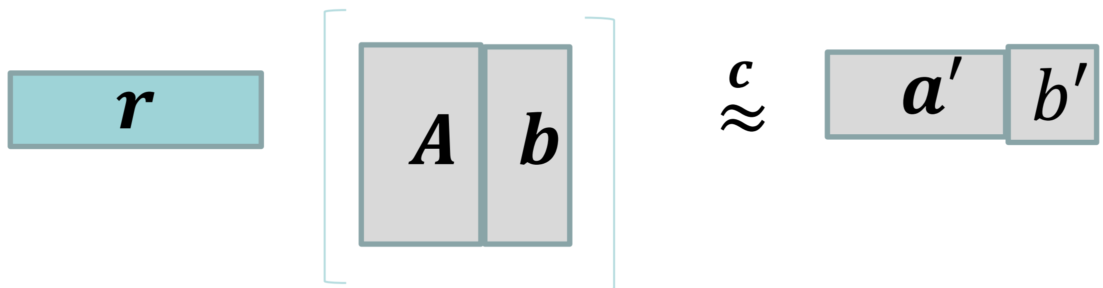
Hybrid 1. Change the public key to random (from LWE).

$$\widetilde{\boldsymbol{pk}} = (\boldsymbol{A}, \boldsymbol{b}), \tilde{\boldsymbol{c}} = \boldsymbol{Enc}\left(\widetilde{\boldsymbol{pk}}, \mu\right) = \boldsymbol{rA}, \boldsymbol{rb} + \mu \lfloor q/2 \rfloor)$$

Hybrids 0 and 1 are comp. indist. by decisional LWE.

Detour: Leftover Hash Lemma
[Impagliazzo-Levin-Luby'90]

We want to understand how $\boldsymbol{rA}, \boldsymbol{rb} = \boldsymbol{r}[\boldsymbol{A}\ |\boldsymbol{b}]$ is distributed when A, b is random (and public).



If $\boldsymbol{r}$ is truly random, so is $\boldsymbol{r}[\boldsymbol{A}\ |\boldsymbol{b}]$.

But r is NOT truly random! It has small entries.

Nevertheless, $\boldsymbol{r}$ has entropy. Leftover hash lemma tells us that matrix multiplication turns (sufficient) entropy into true randomness. We need $m \gg (n+1)\log q$.

Security

Theorem: under decisional LWE, the scheme is IND-secure. In fact, even more: a ciphertext together with the public key is pseudorandom.

Hybrid 1. Change the public key to random (from LWE).

$$\widetilde{\boldsymbol{pk}} = (\boldsymbol{A}, \boldsymbol{b}), \tilde{\boldsymbol{c}} = \boldsymbol{Enc}(\widetilde{\boldsymbol{pk}}, \mu) = \boldsymbol{rA}, \boldsymbol{rb} + \mu \lfloor q/2 \rfloor)$$

Hybrids 0 and 1 are comp. indist. by decisional LWE.

Security

Theorem: under decisional LWE, the scheme is IND-secure. In fact, even more: a ciphertext together with the public key is pseudorandom.

Hybrid 2. Change $\boldsymbol{rA}, \boldsymbol{rb}$ into random.

$$\widetilde{\boldsymbol{pk}} = (\boldsymbol{A}, \boldsymbol{b}), \tilde{\boldsymbol{c}} = \boldsymbol{Enc}\left(\widetilde{\boldsymbol{pk}}, \mu\right) = \boldsymbol{a}', b' + \mu\,\lfloor q/2 \rceil)$$

Hybrids 1 and 2 are stat. indist. by leftover hash lemma.

Security

Theorem: under decisional LWE, the scheme is IND-secure. In fact, even more: a ciphertext together with the public key is pseudorandom.

Hybrid 2. Change $\boldsymbol{rA}, \boldsymbol{rb}$ into random.

$$\widetilde{\boldsymbol{pk}} = \ (\boldsymbol{A}, \boldsymbol{b}), \tilde{\boldsymbol{c}} = \boldsymbol{Enc}\left(\widetilde{\boldsymbol{pk}}, \mu\right) = \boldsymbol{a}', b' + \mu \lfloor q/2 \rfloor)$$

Now, we have the message μ encrypted with a one-time pad which perfectly hides μ. ∎

Public-key Encryption

[Regev05]

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Public key pk: for $i\ from\ 1\ to\ m = 2(n+1)\log q$

$$(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$$

- Encrypting a message bit μ: pick a random vector $\boldsymbol{r} \in \{0,1\}^m$

$$(\boldsymbol{r}\boldsymbol{A}, \boldsymbol{r}\boldsymbol{b} + \mu \lfloor q/2 \rfloor)$$

- Decryption: compute

$$\boldsymbol{r}\boldsymbol{b} + \mu \lfloor q/2 \rfloor - (\boldsymbol{r}\boldsymbol{A})\boldsymbol{s}$$

and round to nearest multiple of q/2.

Next Lecture:
Fully Homomorphic Encryption