Next, we will consider digital signatures. To do so, we first introduce a "dual" of the LWE problem:

<u>Short Integer Solutions (SIS)</u>: The SIS problem is defined with respect to lattice parameters n, m, q and a norm bound β. The $\mathrm{SIS}_{n,m,q,\beta}$ problem says that for $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, no efficient adversary can find a non-zero vector $x \in \mathbb{Z}^m$ where

$$Ax = 0 \in \mathbb{Z}_q^n \quad \text{and} \quad \|x\| \leq \beta$$

[In this course, we will always use the ℓ_∞-norm]

In lattice-based cryptography, the lattice dimension n will be the primary security parameter.

<u>Notes</u>:
- The norm bound β should satisfy $\beta \leq q$. Otherwise, a trivial solution is to set $x = (q, 0, 0, \dots, 0)^T$.
- We need to choose m, β to be large enough so that a solution does exist.
  - ↳ When $m = \Omega(n \log q)$ and $\beta > \sqrt{m}$ a solution always exists. In particular, when $m \geq \lceil n \log q \rceil$, there always exists $x \in \{-1, 0, 1\}^m$ such that $Ax = 0$:
    - There are $2^m \geq 2^{n \log q} = q^n$ vectors $y \in \{0,1\}^m$
    - Since $Ay \in \mathbb{Z}_q^n$, there are at <u>most</u> q^n possible outputs of Ay

    By a counting argument, there exist $y_1 \neq y_2 \in \{0,1\}^m$ such that $Ay_1 = Ay_2$
    - Thus, if we set $x = y_1 - y_2 \in \{-1, 0, 1\}^m$, then $Ax = A(y_1 - y_2) = Ay_1 - Ay_2 = 0 \in \mathbb{Z}_q^n$ and $\|y_1 - y_2\| \leq \sqrt{m}$

Observe that LWE implies SIS. Namely, an algorithm for SIS can be used to break LWE:

1. On input an LWE challenge (A, b^T), use the SIS solver to obtain a low-norm $x \in \mathbb{Z}_q^m$ where $Ax = 0$.
2. Output 1 if $|b^T x|$ is small and 0 otherwise.

If $b^T = s^T A + e^T$, then $b^T x = s^T A x + e^T x = e^T x$, which is small.
If $b \xleftarrow{R} \mathbb{Z}_q^m$, then $b^T x$ is uniform over $\mathbb{Z}_q$ (since $x \neq 0$), so $|b^T x|$ will not be small.

We can directly appeal to SIS to obtain a CRHF: $H: \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \longrightarrow \mathbb{Z}_q^n$ where we set $m > \lceil n \log q \rceil$.

In this case, domain has size $2^m > 2^{n \log q} = q^n$, which is the size of the output space. Collision resistance follows assuming $\mathrm{SIS}_{n,m,q,\beta}$ for any $\beta \geq 1$

The SIS hash function supports efficient local updates:

Suppose you have a public hash $h = H(x)$ of a bit-string $x \in \{0,1\}^m$. Later, you want to update $x \mapsto x'$ where x and x' only differ on a <u>few</u> indices (e.g., updating an entry in an address book). For instance, suppose x and x' differ only on the first bit (e.g., $x_1 = 0$ and $x_1' = 1$). Then observe the following

$$h = H(k, x) = A \cdot x$$

$$= \begin{pmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_m \\ | & | & & | \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \sum_{i \in [m]} x_i a_i = \sum_{i=2}^m x_i a_i \quad \text{since } x_1 = 0$$

$$h' = H(k, x') = A \cdot x'$$

$$= \sum_{i \in [m]} x_i' a_i = x_1' a_1 + \sum_{i=2}^m x_i' a_i = a_1 + \sum_{i=2}^m x_i' a_i = a_1 + h \quad \text{since } x_i' = x_i \text{ for all } i \geq 2$$

Thus, we can easily update h to h' by just adding to it the first column of A without (re)computing the <u>full</u> hash function.

We will now show how to construct digital signatures from SIS in the random oracle model.

We first introduce the inhomogeneous SIS (ISIS) problem.

<u>Inhomogeneous SIS</u>: The inhomogeneous SIS problem is defined with respect to lattice parameters n, m, q and a norm bound β. The $\mathrm{ISIS}_{n,m,q,\beta}$ problem says that for $A \xleftarrow{R} \mathbb{Z}_q^{n\times m}$, $u \xleftarrow{R} \mathbb{Z}_q^n$, no efficient adversary can find a non-zero vector $x \in \mathbb{Z}^m$ where

$$Ax = u \in \mathbb{Z}_q^n \quad \text{and} \quad \|x\| \leq \beta$$

For many choices of parameters, hardness of SIS $\Rightarrow$ hardness of inhomogeneous SIS

The SIS and ISIS problems can be leveraged to construct <u>lattice trapdoors</u>. We define the syntax here:

- $\mathrm{TrapGen}(n,m,q,\beta) \to (A, td_A)$: On input the lattice parameters n, m, q, the trapdoor-generation algorithm outputs a matrix $A \in \mathbb{Z}_q^{n\times m}$ and a trapdoor td_A
- $f_A(x) \to y$: On input $x \in \mathbb{Z}_q^m$, computes $y = Ax \in \mathbb{Z}_q^n$
- $f_A^{-1}(td_A, y) \to x$: On input the trapdoor td_A and an element $y \in \mathbb{Z}_q^n$, the inversion algorithm outputs a value $\|x\| \leq \beta$

Moreover, for a suitable choice of n, m, q, β, these algorithms satisfy the following properties:

- For all $y \in \mathbb{Z}_q^n$, $f_A^{-1}(td_A, y)$ outputs $x \in \mathbb{Z}_q^n$ such that $\|x\| \leq \beta$ and $Ax = y$
- The matrix A output by TrapGen is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$

Lattice trapdoors have received significant amount of study and we will not have time to study it extensively. Here, we will describe the high-level idea behind a very useful and versatile trapdoor known as a "gadget" trapdoor

<u>Observation</u>: SIS is easy with respect to G:

$$G \cdot \begin{pmatrix} 2 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 0 \in \mathbb{Z}_q^n \quad \Rightarrow \text{norm of this vector is } 2$$

Inhomogenous SIS is also easy with respect to G: take any target vector $y \in \mathbb{Z}_q^n$ and output $G^{-1}(y) \in \{0,1\}^m$.

We now have a matrix with a "public" trapdoor. To construct a <u>secret</u> trapdoor function (useful for cryptographic applications), we will "hide" the gadget matrix in the matrix A, and the trapdoor will be a "short" matrix (i.e., matrix with small entries) that recovers the gadget.

More precisely, a gadget trapdoor for a matrix $A \in \mathbb{Z}_q^{n \times k}$ is a short matrix $R \in \mathbb{Z}_q^{k \times m}$ such that

$$A \cdot R = G \in \mathbb{Z}_q^{n \times m}$$

We say that R is "short" if all values are small. [We will write $\|R\|$ to refer to the largest value in R].

Suppose we know $R \in \mathbb{Z}_q^{m \times m}$ such that $AR = G$. We can then define the inversion algorithm as follows:

- $f_A^{-1}(td_A = R, y \in \mathbb{Z}_q^n)$: Output $x = R \cdot G^{-1}(y)$.

<u>Important note:</u> When using trapdoor functions in a setting where the adversary can see trapdoor evaluations, we actually need to <u>randomize</u> the computation of f_A^{-1}. Otherwise, we <u>leak</u> the trapdoor. (We will revisit this later.)

We check two properties.

1. $Ax = AR \cdot G^{-1}(y) = G \cdot G^{-1}(y) = y$ so x is indeed a valid pre-image
2. $\|x\| = \|R \cdot G^{-1}(y)\| \leq m \cdot \|R\| \|G^{-1}(y)\| = m \cdot \|R\|$

   Thus, if $\|R\|$ is small, then $\|x\|$ is also small (think of β as a large polynomial in n).

   (Recall we are using ℓ_∞-norm now)

<u>Remaining question</u>: How do we generate A together with a trapdoor (and so that A is statistically close to uniform)?

Many techniques to do so; we will look at one approach using the LHL:

Sample $\bar{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $\bar{R} \xleftarrow{R} \{0,1\}^{m \times m}$.

Set $A = [\bar{A} \mid \bar{A}\bar{R} + G] \in \mathbb{Z}_q^{n \times 2m}$

Output $A \in \mathbb{Z}_q^{n \times 2m}$, $td_A = R = \begin{bmatrix} -\bar{R} \\ I \end{bmatrix} \in \mathbb{Z}_q^{2m \times m}$

First, we have by construction that $AR = -\bar{A}\bar{R} + \bar{A}\bar{R} + G = G$, and moreover $\|R\| = 1$. It suffices to argue that A is statistically close to uniform (without the trapdoor R). This boils down to showing that $\bar{A}\bar{R} + G$ is statistically close to uniform given $\bar{A}$. We appeal to the LHL:

1. From the previous lecture, the function $f_A(x) = Ax$ is **universal**
2. Thus, by the LHL, if $m \geq 3n \log q$, then Ar is statistically close to uniform in $\mathbb{Z}_q^n$ when $r \xleftarrow{R} \{0,1\}^m$.
3. Claim now follows by a hybrid argument (applied to each column of R)

Thus, given $\bar{A}$, the matrix $\bar{A}\bar{R}$ is still statistically close to uniform. Corresponding, A is statistically close to uniform.

Digital signatures from lattice trapdoors: We can use lattice trapdoors to obtain a digital signature scheme in the random oracle model (this is essentially an analog of RSA signatures):

- KeyGen: $(A, td_A) \leftarrow \text{TrapGen}(n, m, q, \beta)$ [lattice parameters n, m, q, β are based on security parameter λ]
  Output $vk = A$ and $sk = td_A$
- Sign(sk, m): Output $\sigma \leftarrow f_A^{-1}(td_A, H(m))$. Here, $H : \{0,1\}^* \to \mathbb{Z}_q^n$ is modeled as a random oracle.
- Verify(vk, m, σ): Check that $\|\sigma\| \leq \beta$ and that $f_A(\sigma) = H(m)$.

Consider instantiation with gadget trapdoors:

- verification key: $A \in \mathbb{Z}_q^{n \times m}$
  signing key: $R \in \{0,1\}^{m \times m}$ such that $AR = G$
- signature on m: $y \leftarrow H(m) \in \mathbb{Z}_q^n$
  output $\sigma = v = \boxed{R \cdot G^{-1}(y)}$
- verification: check that
  $$A \cdot v = ARG^{-1}(y) = G \cdot G^{-1}(y) = y$$
  and v is short

Rationale for security:

- To forge a signature on m, adversary has to find v such that $Av = H(m)$
- Matrix A is statistically close to uniform and v is uniform, so this corresponds to solving the ISIS problem

Problem: Signing queries leak information about R.
Adversary can compute $H(m) = y$ and $G^{-1}(y)$, so signing becomes a linear function!

Early approach of Goldreich-Goldwasser-Halevi was insecure — explicit key-recovery attack by Nguyen, Regev

In the context of the security proof, simulator needs a way to answer signing queries (without a trapdoor for A).

Requirement: Randomize the signing algorithm to hide trapdoor R

Approach: Instead of outputting a fixed (deterministic) preimage, use the trapdoor to sample a preimage where $Av = H(m)$.
↳ Can show that sample does not leak the trapdoor used [see Gentry - Peikert - Vaikuntanathan]

This is the basis of the Falcon post-quantum signature scheme standardized by NIST.