**Short Integer Solutions (SIS):** The SIS problem is defined with respect to lattice parameters $n, m, q$ and a norm bound $\beta$. The $\text{SIS}_{n,m,q,\beta}$ 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

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

[We will use $\ell_\infty$ norm: $\|v\| = \|v\|_\infty = \max_i |v_i|$]

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

**Notes:** - The norm bound $\beta$ should satisfy $\beta \leq q$.

- We need to choose $m, \beta$ to be large enough so that a solution does exist.
  - ↳ When $m = \Omega(n \log q)$ and $\beta \geq 1$, 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 $Ax = 0$: (← recall that we are using the $\ell_\infty$ norm (unless otherwise noted))
    - 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 most $q^n$ 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 $Ax = A(y_1 - y_2) = Ay_1 - Ay_2 = 0 \in \mathbb{Z}_q^n$

**SIS as a lattice problem:** given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, find non-zero $x \in \mathbb{Z}_q^m$ such that $Ax = 0 \in \mathbb{Z}_q^n$ and $\|x\| \leq \beta$.

↳ Can be viewed as an average-case version of finding short vectors in a "q-ary" lattice:

$$\mathcal{L}^\perp(A) = \{ z \in \mathbb{Z}^m : Az = 0 \pmod{q} \}$$

Notice that by construction, $q\mathbb{Z}^m \subseteq \mathcal{L}^\perp(A)$

← "q-ary" lattice (e.g., vectors where all entries are integer multiples of $q$)

**Hardness of SIS:** Ajtai first showed (in 1996) that average-case hardness of SIS can be based on worst-case hardness of certain lattice problems ⟹ long sequence of works understanding and improving the worst-case to average-case reductions

**Typical statement:** Let $n$ be the lattice dimension. For any $m = \text{poly}(n)$, norm bound $\beta > 0$, and sufficiently large $q \geq \beta \cdot \text{poly}(n)$, Then, the $\text{SIS}_{n,m,q,\beta}$ problem is at least as hard as solving $\text{GapSVP}_\gamma$ on an arbitrary $n$-dimensional lattice for $\gamma = \beta \cdot \text{poly}(n)$.

↳ i.e., solving SIS is as hard as approximating GapSVP in the worst case!

We can use SIS to construct a collision-resistant hash function (CRHF).

**Definition.** A keyed hash family $H: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is collision-resistant if the following properties hold:

- Compressing: $|\mathcal{Y}| < |\mathcal{X}|$
- Collision-Resistant: For all efficient adversaries $\mathcal{A}$:

$$\Pr[k \xleftarrow{R} \mathcal{K}; (x, x') \leftarrow \mathcal{A}(1^\lambda, k) : H(k,x) = H(k,x') \text{ and } x \neq x'] = \text{negl}(\lambda).$$

We can directly appeal to SIS to obtain a CRHF:

$$H: \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \to \mathbb{Z}_q^n$$

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 $\text{SIS}_{n,m,q,\beta}$ for any $\beta \geq \sqrt{\lceil n \log q \rceil}$

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 few 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 full hash function.

The SIS assumption can also serve as the basis for digital signatures — to develop this, we will first need to introduce lattice trapdoors. Will define them first and construct them later.

Inhomogeneous SIS: given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $y \xleftarrow{R} \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ such that $Ax = y \in \mathbb{Z}_q^n$ and $\|x\| \leq \beta$

It turns out that this can actually be used as a trapdoor function. Namely, there exist efficient algorithms

- $\text{TrapGen}(n,m,q,\beta) \to (A, td_A)$ : 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 $n, m, q, \beta$, 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 $Ax = y$ ← in fact, more general: can sample a solution to this system from a discrete Gaussian distribution (sampling needed to ensure solution does not leak trapdoor)
- The matrix $A$ output by TrapGen is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$

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: $(A, td_A) \leftarrow \text{TrapGen}(n,m,q,\beta)$
  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 (ideal hash function) ← just as in RSA-FDH
- Verify$(vk, m, \sigma)$ : Check that $\|\sigma\| \leq \beta$ and that $f_A(\sigma) = H(m)$.

Hardness reduces to hardness of inhomogeneous SIS (similar proof as RSA-FDH). Intuition:

- Matrix $A$ output by KeyGen is uniformly random (property of KeyGen)
- To forge signature on message $m^*$, adversary needs to find short $x$ such that $Ax = y$ where $y = H(m^*)$, which is uniform (essentially solving inhomogeneous SIS instance)
- Formally: Need to rely on random oracle to embed inhomogeneous SIS challenge + respond to signing queries
  (ask in office hours for more details)

# Constructing lattice trapdoors: "gadget trapdoors"

First, we define the "gadget" matrix (there are actually many possible gadget matrices — here, we use a common one sometimes called the "powers-of-two" matrix):

$$G = \begin{pmatrix} 1 \; 2 \; 4 \; 8 \; \cdots \; 2^{\lfloor \log q \rfloor} & & & \\ & 1 \; 2 \; 4 \; \cdots \; 2^{\lfloor \log q \rfloor} & & \\ & & \ddots & \\ & & & 1 \; 2 \; 4 \; \cdots \; 2^{\lfloor \log q \rfloor} \end{pmatrix}$$

Each row of $G$ consists of the powers of two (up to $2^{\lfloor \log q \rfloor}$). Thus, $G \in \mathbb{Z}_q^{n \times n \lfloor \log q \rfloor}$. Oftentimes, we will just write $G \in \mathbb{Z}_q^{n \times m}$ where $m > n \lfloor \log q \rfloor$. Note that we can always pad $G$ with all-zero columns to obtain the desired dimension.

Observation: SIS is easy with respect to $G$:

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

Inhomogenous SIS is also easy with respect to $G$: take any target vector $y \in \mathbb{Z}_q^n$.
Let $y_{i,\lfloor \log q \rfloor}, \ldots, y_{i,1}$ be the binary decomposition of $y_i$ (the $i^{th}$ component of $y$). Then,

$$G \cdot \begin{pmatrix} y_{1,1} \\ y_{1,2} \\ \vdots \\ y_{1,\lfloor \log q \rfloor} \\ y_{2,1} \\ \vdots \\ y_{2,\lfloor \log q \rfloor} \\ \vdots \\ y_{n,1} \\ \vdots \\ y_{n,\lfloor \log q \rfloor} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{\lfloor \log q \rfloor} 2^j y_{1,j} \\ \vdots \\ \sum_{j=1}^{\lfloor \log q \rfloor} 2^j y_{n,j} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = y$$

↑ Observe that this is a 0/1 vector (binary valued vector), so the $\ell_\infty$-norm is exactly 1

We will denote this "bit-decomposition" operation by the function $G^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$

↑ important: $G^{-1}$ is not a matrix (even though $G$ is)!

Then, for all $y \in \mathbb{Z}_q^n$, $G \cdot G^{-1}(y) = y$ and $\|G^{-1}(y)\| = 1$. Thus, both SIS and inhomogeneous SIS are easy with respect to the matrix $G$.

We now have a matrix with a public trapdoor. To construct a secret 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 $x = R \cdot G^{-1}(y)$.

**Important note:** When using trapdoor functions in a setting where the adversary can see trapdoor evaluations, we actually need to randomize the computation of $f_A^{-1}$. Otherwise, we leak the trapdoor. But this basic scheme illustrates the main ideas...

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 $\beta$ as a large polynomial in $n$).

Remaining question: 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 "leftover hash lemma" (also used when arguing security of Regev's PKE scheme)

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

By construction that $AR = -\bar{A}\bar{R} + \bar{A}\bar{R} + G = G$, and moreover $\|R\| = 1$.

By leftover hash lemma, for $m = O(n \log q)$, $(\bar{A}, \bar{A}\bar{R})$ is indistinguishable from $(\bar{A}, U)$ where $U \xleftarrow{R} \mathbb{Z}_q^{n \times m}$