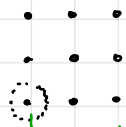
Focus: lattice-based cryptography
- Conjectured post-quantum resilience
  - Number-theoretic assumptions like discrete log and factoring are insecure against quantum computers
- Basis of many NIST post-quantum cryptography standards for post-quantum key agreement and digital signatures
- Security based on worst-case hardness
  - Cryptography has typically relied on average-case hardness (i.e., there exists some <u>distribution</u> of hard instances)
  - Lattice-based cryptography can be based on <u>worst-case</u> hardness (there does not exist an algorithm that solves <u>all</u> instances)
- Enables advanced cryptographic capabilities

<u>Definition</u>: An n-dimensional lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is a discrete additive subspace of $\mathbb{R}^n$
- <u>Discrete</u>: For every $x \in \mathcal{L}$, there exists a neighborhood around x that only contains x:



neighborhood $B_\varepsilon(x) = \{ y \in \mathbb{R}^n : \|x - y\| \leq \varepsilon \}$

discrete means $B_\varepsilon(x) \cap \mathcal{L} = \{x\}$

- <u>Additive subspace</u>: For all $x, y \in \mathcal{L}$: $x + y \in \mathcal{L}$
  $-x \in \mathcal{L}$

<u>Examples</u>: $\mathbb{Z}^n$ (n-dimensional integer-valued vectors)
$q\mathbb{Z}^n$ (n-dimensional integer-valued vectors where each coordinate is multiple of q) "q-ary" lattice

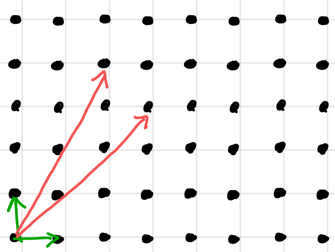
Lattices typically contain infinitely-many points, but are <u>finitely-generated</u> by taking <u>integer</u> linear combinations of a small number of basis vectors:

$$B = [b_1 \mid b_2 \mid \cdots \mid b_k] \in \mathbb{R}^{n \times k}$$ (vectors are linearly independent over $\mathbb{R}$)

← k is the rank of the lattice (full-rank: $k = n$)

$$\mathcal{L}(B) = \Big\{ \sum_{i \in [n]} \alpha_i b_i \;\Big|\; \alpha_i \in \mathbb{Z} \Big\}$$
$$= B \cdot \mathbb{Z}^k$$

A lattice can have many basis:



standard basis for $\mathbb{Z}^2$
alternative basis for $\mathbb{Z}^2$

} Choice of basis makes a big difference in hardness of lattice problems
↳ <u>often</u>: bad basis is public key
good basis is trapdoor

Definition. Let $\mathcal{L}$ be an n-dimensional lattice. Then, the minimum distance $\lambda_1(\mathcal{L})$ is the norm of the shortest non-zero vector in $\mathcal{L}$:

$$\lambda_1(\mathcal{L}) = \min_{v \in \mathcal{L}\setminus\{0\}} \|v\|$$

The i^{th} <u>successive minimum</u> $\lambda_i(\mathcal{L})$ is the smallest $r \in \mathbb{R}$ such that $\mathcal{L}$ contains i <u>linearly independent</u> basis vectors of norm at most r.

n = 2 case is easy

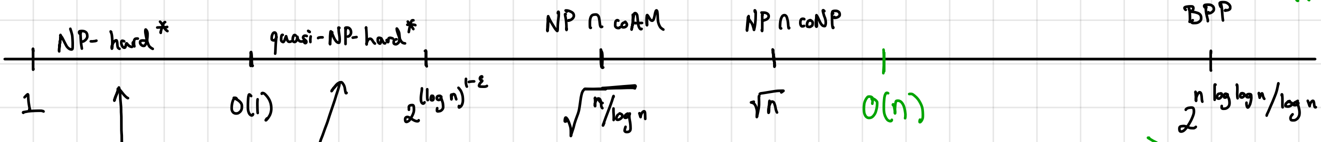
Computational problems on lattices: [problems parameterized by lattice dimension n] (can solve exactly using Gauss' algorithm)

- Shortest vector problem (SVP): Given a basis B of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(B)$, find $v \in \mathcal{L}$ such that $\|v\| = \lambda_1(\mathcal{L})$
- Approximate SVP (SVP_γ): Given a basis B of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(B)$, find $v \in \mathcal{L}$ such that $\|v\| \le \gamma \cdot \lambda_1(\mathcal{L})$
- Decisional approximate SVP ($GapSVP_\gamma$): Given a basis B of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(B)$, decide if

$$\lambda_1(\mathcal{L}) \le 1 \quad \text{or if} \quad \lambda_1(\mathcal{L}) \ge \gamma$$

example language in NP ∩ coAM is graph isomorphism (not known to be in coNP)

Complexity of GapSVP depends on approximation factor γ:



under randomized reductions

"nearly polynomial"

[mapping NO instances to NO instances w.p. 1
and YES instances to YES instances w.p. 2/3]

approximation factor sufficient for cryptography (e.g., OWF / PKE exists)

unlikely to allow basing crypto on NP hardness since for approximation factors bigger than $\sqrt{n}$, $GapSVP_\gamma \in NP \cap coNP$

<u>Algorithms for SVP</u>: Lenstra-Lenstra-Lovász (LLL) algorithm (lattice reduction)

- Polynomial time algorithm for $\gamma = 2^{n \log\log n/\log n}$ approximation
- Known algorithms for poly(n) approx run in time $2^{\Theta(n)}$ (many need similar space as well)
- Can trade-off time for approximation factor: solve $GapSVP_\gamma$ in time $2^{\Theta(n/\log\gamma)}$
- Same asymptotics with quantum algorithms

Main problems we use for cryptography are short integer solutions (SIS) and learning with errors (LWE)

↳ These reduce to $GapSVP_\gamma$ and $SIVP_\gamma$

↳ Currently open: basing crypto on search-SVP (SVP or SVP_γ)

<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 $\text{SIS}_{n,m,q,\beta}$ 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

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

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

<u>Notes</u>:
- The norm bound β should satisfy $\beta \le q$. Otherwise, a trivial solution is to set $x = (q, 0, 0, \ldots, 0)^T$.
- We need to choose m, β to be large enough so that a solution does exist.
  - $\hookrightarrow$ When $m = \Omega(n \log q)$ and $\beta > \sqrt{m}$ a solution always exists. In particular, when $m \ge \lceil n \log q \rceil$, there always exists $x \in \{-1, 0, 1\}^m$ such that $Ax = 0$:
    - There are $2^m \ge 2^{n \log q} = q^n$ 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 $y_1 \ne y_2 \in \{0,1\}^m$ 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 $\|y_1 - y_2\| \le \sqrt{m}$

SIS can be viewed as an <u>average-case</u> SVP on a lattice defined by $A \in \mathbb{Z}_q^{n \times m}$:

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

in coding-theoretic terms, the matrix A is a "parity-check" matrix

called a "q-ary" lattice since $q\mathbb{Z}^m \subseteq \mathcal{L}^{\perp}(A)$

SIS problem is essentially finding short vectors in the lattice $\mathcal{L}^{\perp}(A)$ where $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$

<u>Theorem</u>. For any $m = \text{poly}(n)$, any $\beta > 0$, and sufficiently large $q \ge \beta \cdot \text{poly}(n)$, there is a probabilistic polynomial time (PPT) reduction from solving GapSVP_γ or SIVP_γ in the worst case to solving $\text{SIS}_{n,m,q,\beta}$ with non-negligible probability, where $\gamma = \beta \cdot \text{poly}(n)$.

We can use SIS to directly obtain a <u>collision-resistant hash function</u> (CRHF).

<u>Definition</u>. A keyed hash family $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is <u>collision-resistant</u> if the following properties hold:
- <u>Compressing</u>: $|\mathcal{Y}| < |\mathcal{X}|$
- <u>Collision-Resistant</u>: For all efficient adversaries $\mathcal{A}$:

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

We can directly appeal to SIS to obtain a CRHF: $H: \mathbb{Z}_q^{n\times m} \times \{0,1\}^m \longrightarrow \mathbb{Z}_q^n$ 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 $SIS_{n,m,q,\beta}$ 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

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

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

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

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

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 hash function is universal – this will be a very useful property (in conjunction with the leftover hash lemma)

Definition. Let $H: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed hash function. We say H is universal if for all $x_0, x_1 \in \mathcal{X}$ where $x_0 \neq x_1$, $\Pr[k \xleftarrow{R} \mathcal{K} : H(k,x_0) = H(k,x_1)] \leq 1/|\mathcal{Y}|$.

Lemma. The SIS hash function $H: \mathbb{Z}_q^{n\times m} \times \{0,1\}^m \longrightarrow \mathbb{Z}_q^n$ is universal.

Proof. Take any $x_0, x_1 \in \{0,1\}^m$ with $x_0 \neq x_1$. If $H(A,x_0) = H(A,x_1)$, then $A(x_0 - x_1) = 0$. Let $a_1, \ldots, a_m \in \mathbb{Z}_q^n$ be columns of A. Then,

$$A(x_0 - x_1) = \sum_{i\in[m]} a_i (x_{0,i} - x_{1,i})$$

Since there exists some $j \in [m]$ where $x_{0,j} \neq x_{1,j}$, the above relation holds only if

$$a_j = \underbrace{\overset{\in\{-1,1\}}{\boxed{(x_{1,j} - x_{0,j})}} \sum_{i\neq j} a_i (x_{0,i} - x_{1,i})}_{\text{independent of } a_j}$$

Thus,

$$\Pr[A \xleftarrow{R} \mathbb{Z}_q^{n\times m} : A(x_0 - x_1) = 0]$$

$$= \Pr[a_1, \ldots, a_m \xleftarrow{R} \mathbb{Z}_q^n : a_j = (x_{1,j} - x_{0,j}) \sum_{i\neq j} a_i (x_{0,i} - x_{1,i})]$$

$$= \frac{1}{q^n}$$

Note: When q is prime, this argument also extends to any domain that is subset of $\mathbb{Z}_q^n$. Namely

$$H: \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m \longrightarrow \mathbb{Z}_q^n$$

is universal.

Definition. Let X be a random variable taking on values in a finite set S. We define the guessing probability of X to be

$$\max_{s \in S} \Pr[X = s]$$

We define the min-entropy of X to be

$$H_\infty(X) = -\log \max_{s \in S} \Pr[X = s]$$

Intuitively: if a random variable has k bits of min-entropy, then its most likely outcome occurs with probability at most 2^{-k} (i.e., there exists at least 2^k possible values for X)

Definition. Let D_0, D_1 be distributions with a common support S. Then, the statistical distance between D_1 and D_2 is defined to be

$$\Delta(D_0, D_1) = \frac{1}{2} \sum_{s \in S} \left| \Pr[t \leftarrow D_0 : t = s] - \Pr[t \leftarrow D_1 : t = s] \right|$$

If D_0 and D_1 are ε-close, then no adversary can distinguish with advantage better than ε

↳ When ε is negligible, we say the two distributions are statistically indistinguishable — denoted $D_0 \overset{s}{\approx} D_1$

↳ Contrast with computational indistinguishability which says no efficient adversary can distinguish — denoted $D_0 \overset{c}{\approx} D_1$

Theorem (Leftover Hash Lemma). Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be an universal hash function. Suppose $x \in \mathcal{X}$ is a random variable with t bits of min-entropy. Then, define the following two distributions:

$$D_0 : k \xleftarrow{R} \mathcal{K},\ y \leftarrow H(k, x);\ \text{output } (k, y)$$

$$D_1 : k \xleftarrow{R} \mathcal{K},\ y \xleftarrow{R} \mathcal{Y};\ \text{output } (k, y)$$

The statistical distance between D_0 and D_1 is at most

$$\Delta(D_0, D_1) \leq \frac{1}{2}\sqrt{|\mathcal{Y}|/2^t}$$

Typical setting: H is universal and $|\mathcal{Y}| = 2^{t - 2\lambda}$. By LHL, $(k, H(k, x)) \overset{s}{\approx} (k, y)$ where $y \xleftarrow{R} \mathcal{Y}$.

This is an example of a "randomness extractor."

We have a source (x) with min-entropy, but not necessarily uniform.

We want to extract from it a uniform random value

LHL shows that universal hash functions can "smooth" out a non-uniform distribution

↓

Incurs loss of 2λ bits of entropy

Common application: extracting uniformly random cryptographic keys from non-uniform source

↳ consider $H : \mathbb{Z}_2^{n \times m} \times \{0,1\}^m \to \mathbb{Z}_2^n$

$$H(A, x) := Ax$$

could be binary representation of a group element

suitable for use as a symmetric key

Not typically used in practice because we need distribution with at least $n + 2\lambda$ bits of min-entropy (≥ 384 bits if $n = \lambda = 128$)

Practical heuristic: use random oracle

does not have to be uniform — just needs min-entropy

typically, we just take n to be the security parameter and set $m = \Theta(n \log q)$

In lattices: If $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $v \xleftarrow{R} \{0,1\}^m$, then $Av \in \mathbb{Z}_q^n$ is uniform when $m > n \log q + 2\lambda$

By a hybrid argument, if we sample $R \xleftarrow{R} \{0,1\}^{m \times m}$, then AR is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$

We will see this used in many constructions

Commitments from SIS (recall: commitment is a "sealed envelope")

- Setup $(1^\lambda) \to$ crs : Samples a common reference string
- Commit $(\text{crs}, \mu; r) \to \sigma$: Commits to a message μ with randomness r

} Here, opening can simply be the pair (m, r). Verifier checks that $\sigma = \text{Commit}(\text{crs}, m; r)$

Useful building block for zero-knowledge proofs and other cryptographic protocols

- Setup (1^λ) : Let n, q be lattice parameters, and $m = \Theta(n \log q)$

  Sample $A_1, A_2 \xleftarrow{R} \mathbb{Z}_q^{n \times m}$. Output crs $= (A_1, A_2)$
- Commit $(\text{crs}, \mu; r)$: Output $\sigma = A_1 m + A_2 r$ where crs $= (A_1, A_2)$

  $\mu, r \in \{0,1\}^m$

  $$= [A_1 \mid A_2] \begin{bmatrix} m \\ r \end{bmatrix} \in \mathbb{Z}_q^n$$

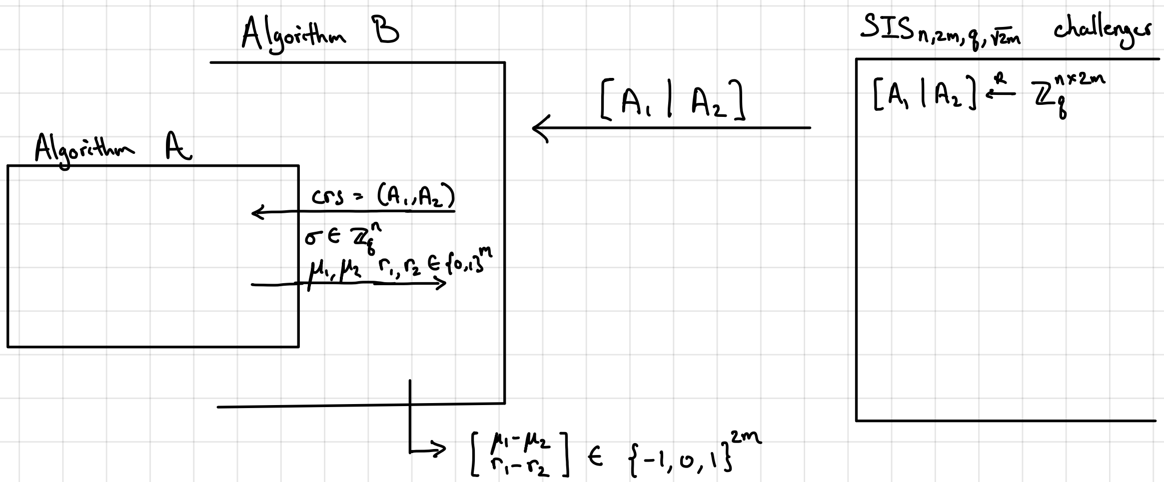
  compressing when $m > n \log q$

Theorem (Statistically Hiding). If $m > 3n \log q$, then scheme is statistically hiding.

Proof. By the LHL, for $r \xleftarrow{R} \{0,1\}^m$, $A_2 r \overset{s}{\approx} \text{Uniform}(\mathbb{Z}_q^n)$. Thus, $A_2 r$ acts as a one-time pad for $A_1 m$.

Theorem (Computational Hiding). Under $\text{SIS}_{n, 2m, q, \sqrt{2m}}$, the commitment scheme is computationally binding.

Proof. Suppose $\mathcal{A}$ can break the binding property. We use $\mathcal{A}$ to construct SIS adversary $\mathcal{B}$:



If $\mathcal{A}$ is successful, then $\mu_1 \neq \mu_2$ and $[A_1 \mid A_2] \begin{bmatrix} \mu_1 \\ r_1 \end{bmatrix} = \sigma = [A_1 \mid A_2] \begin{bmatrix} \mu_2 \\ r_2 \end{bmatrix}$, which means $[A_1 \mid A_2] \begin{bmatrix} \mu_1 - \mu_2 \\ r_1 - r_2 \end{bmatrix} = 0$.

Since $\mu_1 \neq \mu_2$ this is a non-zero SIS solution with norm at most $\sqrt{2m}$.

Compare this with Pedersen commitments from discrete log:

Setup (1^λ) : Take a prime-order group $\mathbb{G} \leftarrow \text{GroupGen}(1^\lambda)$. Let p be the order of $\mathbb{G}$.

Sample $g, h \xleftarrow{R} \mathbb{G}$. Output crs $= (g, h)$

Commit $(\text{crs}, \mu; r)$: Output $g^\mu h^r$.

$\mu, r \in \mathbb{Z}_p$

discrete log		SIS
$g, h \xleftarrow{R} \mathbb{G}$	$\longrightarrow$	$A_1, A_2 \xleftarrow{R} \mathbb{Z}_q^{n \times m}$
g^μ		$A_1 \mu$
h^r		$A_2 r$

We will see many similar parallels between discrete log based systems and lattice-based systems