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 distribution of hard instances)
- Lattice-based cryptography can be based on worst-case hardness (there does not exist an algorithm that solves all instances)
- Enables advanced cryptographic capabilities

Definition: An n-dimensional lattice \( \mathbb{L} \subseteq \mathbb{R}^n \) is a discrete additive subspace of \( \mathbb{R}^n \)
- Discrete: For every \( x \in \mathbb{L} \), there exists a neighborhood around \( x \) that only contains \( x \):
  \[
  B_\varepsilon(x) = \{ y \in \mathbb{R}^n : ||x - y|| \leq \varepsilon \}
  \]
  discrete means \( B_\varepsilon(x) \cap \mathbb{L} = \{ x \} \)
- Additive subspace: For all \( x, y \in \mathbb{L} \): \( x + y \in \mathbb{L} \)
  \(-x \in \mathbb{L} \)

Examples:
- \( \mathbb{Z}^n \) (n-dimensional integer-valued vectors)
- \( \mathbb{Q} \cdot \mathbb{Z}^n \) (n-dimensional integer-valued vectors where each coordinate is multiple of \( q \)) “\( q \)-ary” lattice

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

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

\[
\mathbb{L}(B) = \{ \sum_{i=1}^{k} \alpha_i b_i \mid \alpha_i \in \mathbb{Z} \} \quad \text{(full-rank: } k = n)
\]

A lattice can have many basis:

<table>
<thead>
<tr>
<th>Standard basis for ( \mathbb{Z}^2 )</th>
<th>Alternative basis for ( \mathbb{Z}^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice of basis makes a big difference in hardness of lattice problems</td>
<td></td>
</tr>
<tr>
<td>Bad basis is public key</td>
<td></td>
</tr>
<tr>
<td>Good basis is trapdoor</td>
<td></td>
</tr>
</tbody>
</table>
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 successive minimum \( \lambda_i(\mathcal{L}) \) is the smallest \( r \in \mathbb{R} \) such that \( \mathcal{L} \) contains \( i \) linearly independent basis vectors of norm at most \( r \).

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\( _\gamma \)): Given a basis \( B \) of an \( n \)-dimensional lattice \( \mathcal{L} = \mathcal{L}(B) \), find \( v \in \mathcal{L} \) such that \( \|v\| \leq \gamma \cdot \lambda_1(\mathcal{L}) \)
- Decisional approximate SVP (GapSVP\( _\gamma \)): Given a basis \( B \) of an \( n \)-dimensional lattice \( \mathcal{L} = \mathcal{L}(B) \), decide if \( \lambda_1(\mathcal{L}) \leq 1 \) or if \( \lambda_1(\mathcal{L}) \geq \gamma \)

Complexity of GapSVP depends on approximation factor \( \gamma \):

<table>
<thead>
<tr>
<th>Approximation Factor</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma = 1 )</td>
<td>( \text{NP-hard}^* )</td>
</tr>
<tr>
<td>( \gamma \ll 1 )</td>
<td>( \text{quasi-NP-hard}^* )</td>
</tr>
<tr>
<td>( \gamma \gg 1 )</td>
<td>( \text{NP} \cap \text{coAM} )</td>
</tr>
<tr>
<td>( \gamma \ll \sqrt{n} )</td>
<td>( \text{NP} \cap \text{coNP} )</td>
</tr>
<tr>
<td>( \gamma \gg \sqrt{n} )</td>
<td>( \text{BPP} )</td>
</tr>
</tbody>
</table>

Example language in \( \text{NP} \cap \text{coAM} \): graph isomorphism (not known to be \( \text{NP} \)-hard)

Approximation factor \( \gamma \) 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\( _\gamma \) \( \in \text{NP} \cap \text{coNP} \)

Algorithms for SVP:
- Lenstra - Lenstra - Lovasz (LLL) algorithm (lattice reduction)
  - Polynomial time algorithm for \( \gamma = 2^{\log^* n} \) approximation
  - Known algorithms for poly(\( n \)) approx run in time \( 2^{O(n)} \) (many need similar space as well)
  - Can trade-off time for approximation factor: solve GapSVP\( _\gamma \) in time \( 2^{\Theta(n/\log \gamma)} \)
  - Some asymptotics with quantum algorithms

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

- These reduce to GapSVP and SVP\( _\gamma \)
- Currently open: basing crypto on search-SVP (SVP or SVP\( _\gamma \))
Short Integer Solutions (SIS): The SIS problem is defined with respect to lattice parameters \( m, n, q \) and a norm bound \( p \). The SIS\(_{m,n,q}\_p\) problem says that for \( A \in \mathbb{Z}_q^{n \times m} \), no efficient adversary can find a non-zero vector \( X \in \mathbb{Z}^m \) where \( Ax = 0 \in \mathbb{Z}_q^n \) and \( \|X\| < p \).

In lattice-based cryptography, the lattice dimension \( m \) will be the primary security parameter.

Notes:
- The norm bound \( p \) should satisfy \( p < g \). Otherwise, a trivial solution is to set \( X = (g, 0, 0, \ldots, 0) \).
- We need to choose \( m, p \) to be large enough so that a solution does exist.

\[ \implies \text{When } m = \Omega(n \log q) \text{ and } p > \sqrt{n}, \text{ a solution always exists.} \]

\[ \text{In particular, when } m \geq \Omega(n \log q), \text{ there always exists } x \in \{-1, 0, 1\}^m \text{ such that } Ax = 0: \]

- 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 most \( q^n \) possible outputs of \( Ay \).
- Thus, if we set \( x = y - y_0 \in \{-1, 0, 1\}^m \), then \( Ax = A(y - y_0) = A_y - A_y = 0 \in \mathbb{Z}_q^n \) and \( \|y - y_0\| < p \).

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

\[
\mathbb{L}^d(A) = \{ x \in \mathbb{Z}^m : Ax = 0 \pmod{q} \}.
\]

\[ \uparrow \text{ called a "g-ary" lattice} \]

\[ \text{since } q \mathbb{Z}^m \subseteq \mathbb{L}^d(A) \]

Theorem: For any \( m = \text{poly}(n) \), any \( \beta > 0 \), and sufficiently large \( g \geq \beta \cdot \text{poly}(n) \), there is a probabilistic polynomial time (PPT) reduction from solving \( \text{GapSVP}_g \) or \( \text{SIVP}_g \) in the worst case to solving \( \text{SIS}_{m,n,q,p} \) with non-negligible probability, where \( \gamma = \beta \cdot \text{poly}(n) \).

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

Definition: A keyed hash family \( H : K \times X \rightarrow Y \) is collision-resistant if the following properties hold:

- Compressing: \( |Y| < |X| \)
- Collision-Resistant: For all efficient adversaries \( A \):

\[ \Pr \left[ k \in K ; (z, x) \leftarrow A(\mathbb{Z}^n, p) : H(k, z) = H(k, x) \text{ and } z \neq x \right] = \text{negl}(\lambda) \]
We can directly appeal to SIS to obtain a CRHF: \( H : \mathbb{Z}_q^{m \times n} \times \{0,1\}^m \rightarrow \mathbb{Z}_q^n \) where we set \( m > \lceil \ln \log q \rceil \).

In this case, domain has size \( 2^n > 2^{\lceil \ln \log q \rceil} = q^n \), which is the size of the output space. Collision-resistance follows assuming SIS \( n, q, p \) for any \( p \geq \sqrt{\ln \log q} \).

The SIS hash function supports efficient local updates:

Suppose you have a public hash \( h = H(x) \) of a bit-string \( x \in \{0,1\}^n \). Later, you want to update \( x \rightarrow x' \) 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 \( (e.g., x = 0 \text{ and } x' = 1) \). Then observe the following:

\[
\begin{align*}
  h(x) &= \left( \begin{array}{c}
    a_1 \\
    a_2 \\
    \vdots \\
    a_m
  \end{array} \right) \\
  h'(x') &= \left( \begin{array}{c}
    a_1' \\
    a_2' \\
    \vdots \\
    a_m'
  \end{array} \right)
\end{align*}
\]

Then, we can easily update \( h \) to \( h' \) by just adding to it the first column of \( A \) without recomputing 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 \( H : K \times X \rightarrow Y \) be a keyed hash function. We say \( H \) is universal if for all \( X_0, X_1 \in X \) where \( X_0 \neq X_1 \), \( \Pr[k \in K : H(k, X_0) = H(k, X_1)] \leq \frac{1}{|Y|} \).

**Lemma.** The SIS hash function \( H : \mathbb{Z}_q^{m \times n} \times \{0,1\}^m \rightarrow \mathbb{Z}_q^n \) is universal.

**Proof.** Take any \( x_0, x_1 \in \{0,1\}^n \) 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,

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

Since there exists some \( j \in [n] \) where \( x_{0j} \neq x_{1j} \), the above relation holds only if \( \sum_{i \in [n]} a_i = 0 \).

\[
a_j = \frac{(x_{0j} - x_{1j}) \sum_{i \in [n]} a_i \cdot (x_{0i} - x_{1i})}{x_{1j} - x_{0j}}
\]

is independent of \( a_j \).

Thus, \( \Pr[A \in \mathbb{Z}_q^{m \times n} : A(x_0 - x_1) = 0] \)

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

\[
= \frac{1}{q^n}
\]

Note: When \( q \) is prime, this argument also extends to any domain that is subset of \( \mathbb{Z}_q^n \). Namely, \( H : \mathbb{Z}_q^{m \times n} \rightarrow \mathbb{Z}_q^n \) 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

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

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 $D_0$ and $D_1$ is defined to be

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

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

$\implies$ When $\epsilon$ is negligible, we say the two distributions are statistically indistinguishable.

$\implies$ Contrast with computational indistinguishability which says no efficient adversary can distinguish.

denoted $D_0 \approx D_1$

denoted $D_0 \triangleq D_1$

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

$D_0: k \sim K, y \sim H(k, x); \text{ output } (k, y)$

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

The statistical distance between $D_0$ and $D_1$ is at most

$$\Delta(D_0, D_1) \leq \frac{1}{2} \sqrt{181/2^t}$$

Typical setting: $H$ is universal and $|Y| = 2^t - 2^n$. By LHL, $\Pr[(k, H(k, x)) \neq (k, y)]$ where $y \sim Y$.

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.

Incorporates loss of $2N$ bits of entropy.

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

$\implies$ Consider $H: Z_2^{n+m} \rightarrow \{0,1\}^n$:

$H(A, x) = A \oplus x$

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 $n + 2N$ bits of min-entropy ($\geq 3N$ bits if $n = \log q = 2N$)

Practical heuristic: use random oracle.

In lattices: If $A \in Z_q^{n+m}$ and $v \in \{0,1\}^m$, then $AV \in Z_q^n$ is uniform when $m > n \log q + 2N$ the security parameter and $q = \Theta(n \log q)$

By a hybrid argument, if we sample $R \in \{0,1\}^{n+m}$, then $AR$ is statistically close to uniform over $Z_q^n$.

We will see this used in many constructions.
Commitments from SIS (recall: commitment is a "sealed envelope")

- Setup (14): $\text{crs} = (A, A_x)$: Samples a common reference string $\text{crs}$: $\text{Samples m}_r$ with randomness $r$

  - $\text{Commit (crs, } m; r) \rightarrow o$: $\text{Commits to a message } m$ with randomness $r$

  - $\text{Setup (1): Let } n, q \text{ be lattice parameters, and } m = \Theta(n \log q)$
    
    Sample $A, A_x \in \mathbb{Z}_q^m$. Output $\text{crs} = (A, A_x)$

  - $\text{Commit (crs, } m; r) \rightarrow o$: $\text{Output } o = A, m + A_x r$ where $\text{crs} = (A, A_x)$

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

- Setup (2): Let $n, q$ be lattice parameters, and $m = \Theta(n \log q)$

  Sample $A, A_x \in \mathbb{Z}_q^m$. Output $\text{crs} = (A, A_x)$

  - $\text{Commit (crs, } m; r) \rightarrow o$: $\text{Output } o = A, m + A_x r$ where $\text{crs} = (A, A_x)$

Here, opening can simply be the pair $(m, r)$.

Verifier checks that $o = \text{Commit (crs, } m; r)$

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

Proof. By the LHL, for $r \in \{0,1\}^n$, $A_x r \approx \text{Uniform } (\mathbb{Z}_q^n)$. Thus, $A_x r$ acts as a one-time pad for $A, m$.

Theorem (Computational Hiding). Under SIS$_n, \sigma, g, \mu_n$, the commitment scheme is computationally binding.

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

Algorithm $B$

Algorithm $A$

If $A$ is successful, then $\mu, \mu_x$ and $[A_1 | A_x][\mu + r]$ = $\sigma = [A_1 | A_x][\mu_x + r]$ which means $\sigma = [A_1 | A_x][\mu + r] = 0$.

Since $\mu, \mu_x$ this is a non-zero SIS solution with norm at most $\frac{1}{2} m$.

Compare this with Pedersen commitments from discrete log:

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

Sample $g, h \leftarrow G$. Output $\text{crs} = (g, h)$

Commit (crs, $m; r$): Output $g^{m + h r}$.

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