Lattice Cryptography: Introduction and Open Problems

Daniele Micciancio

Department of Computer Science and Engineering
University of California, San Diego

August 2015
The simplest example of lattice is $\mathbb{Z}^n = \{(x_1, \ldots, x_n): x_i \in \mathbb{Z}\}$.
The simplest example of lattice is \( \mathbb{Z}^n = \{(x_1, \ldots, x_n) : x_i \in \mathbb{Z}\} \).

Other lattices are obtained by applying a linear transformation

\[
B : x = (x_1, \ldots, x_n) \mapsto Bx = x_1 \cdot b_1 + \cdots + x_n \cdot b_n
\]
Lattice Cryptography

  - Algorithmic breakthrough
  - Efficient approximate solution of lattice problems
  - Exponential approximation factor, but very good in practice
  - Killer App: Cryptanalysis
Lattice Cryptography

- **Lenstra, Lenstra, Lovasz (1982)**: The “LLL” paper “Factoring Polynomials with Rational Coefficients”
  - Algorithmic breakthrough
  - Efficient approximate solution of lattice problems
  - Exponential approximation factor, but very good in practice
  - Killer App: Cryptanalysis

- **Ajtai (1996)**: “Generating Hard Instances of Lattice Problems”
  - Marks the beginning of the modern use of lattices in the design of cryptographic functions
“cryptography . . . generation of a specific instance of a problem in NP which is thought to be difficult”.

“NP-hard problems”
“very famous question (e.g., prime factorization).”

“Unfortunately ‘difficult to solve’ means . . . in the worst case”

“no guidance about how to create [a hard instance]”

“possible solution”

1. “find a set of randomly generated problems”, and
2. “show that if there is an algorithm which [works] with a positive probability, then there is also an algorithm which solves the famous problem in the worst case.”

“In this paper we give such a class of random problems.”
Example: Discrete Logarithm (DLOG)

- $p$: a prime
- $\mathbb{Z}_p^*$: multiplicative group
- $g \in \mathbb{Z}_p^*$: generator of (prime order sub-)group $G = \{g^i : i \in \mathbb{Z}\} \subseteq \mathbb{Z}_p^*$
- Input: $h = g^i \mod p$

DLOG Problem

Given $p, g, h$, recover $i$ (modulo $q = o(g)$)
Example: Discrete Logarithm (DLOG)

- \( p \): a prime
- \( \mathbb{Z}_p^* \): multiplicative group
- \( g \in \mathbb{Z}_p^* \): generator of (prime order sub-)group \( G = \{ g^i : i \in \mathbb{Z} \} \subseteq \mathbb{Z}_p^* \)
- Input: \( h = g^i \mod p \)

**DLOG Problem**

Given \( p, g, h \), recover \( i \) (modulo \( q = o(g) \))

**Random Self Reducibility**

If you can solve DLOG for random \( g \) and \( h \) (with some probability), then you can solve it for any \( g, h \) in the worst-case.
Given arbitrary $g, h$

Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$. Notice: $g', h' \in G$ are (almost) uniformly random.

Find $j = \text{DLOG}(g', h') = ib$. Output $j/b \pmod{q}$.

Conclusion: We know how to choose $g, h, G$. But, how do we choose $G$?
1. Given arbitrary $g, h$

2. Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$. 
1. Given arbitrary $g, h$

2. Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$.

3. Notice:
   - $g', h' \in G$ are (almost) uniformly random
   - $h' = h^{ab} = g^{iab} = (g')^{ib}$

Conclusion
We know how to choose $g, h \in G$.
But, how do we choose $G$?
Given arbitrary $g, h$

1. Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$.

2. Notice:
   - $g', h' \in G$ are (almost) uniformly random
   - $h' = h^{ab} = g^{iab} = (g')^i$

3. Find $j = DLOG(g', h') = ib$
DLOG: Random Self Reducibility (RSR)

1. Given arbitrary $g, h$
2. Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$.
3. Notice:
   - $g', h' \in G$ are (almost) uniformly random
   - $h' = h^{ab} = g^{iab} = (g')^{ib}$
4. Find $j = DLOG(g', h') = ib$
5. Output $j/b \pmod{q}$. 

Conclusion: We know how to choose $g, h \in G$.

But, how do we choose $G$?
DLOG: Random Self Reducibility (RSR)

1. Given arbitrary $g, h$
2. Compute $g' = g^a$ and $h' = h^{ab}$ for random $a, b \in \mathbb{Z}_q^*$.
3. Notice:
   - $g', h' \in G$ are (almost) uniformly random
   - $h' = h^{ab} = g^{iab} = (g')^b$
4. Find $j = DLOG(g', h') = ib$
5. Output $j/b \pmod q$.

Conclusion

We know how to choose $g, h \in G$.
But, how do we choose $G$?
Lattice Assumption

The complexity of solving lattice problems in $n$-dimensional lattices grows superpolynomially (or exponentially) in $n$. Similarly, one may conjecture that the complexity of DLOG grows superpolynomially in $n = \log p$ or $n = \log |G|$. This is not the same: For any $n$, there are (exponentially) many primes $p$. Typically, $p$ is chosen at random among all $n$-bit primes. Assumption is still average-case: DLOG is hard for random $p$. We do not know how to reduce DLOG($\mathbb{Z}^*_p$) to DLOG($\mathbb{Z}^*_q$). RSR provides no guidance on how to choose $p$. 

Daniele Micciancio (UCSD) Lattice Cryptography: Introduction and Open Problems August 2015 7 / 32
Lattice Assumption

The complexity of solving lattice problems in $n$-dimensional lattices grows superpolynomially (or exponentially) in $n$.

- Similarly, one may conjecture that the complexity of DLOG grows superpolynomially in $n = \log p$ or $n = \log |G|$.
Lattice Assumption

The complexity of solving lattice problems in $n$-dimensional lattices grows superpolynomially (or exponentially) in $n$.

- Similarly, one may conjecture that the complexity of DLOG grows superpolynomially in $n = \log p$ or $n = \log |G|$.
- This is not the same:
  - For any $n$, there are (exponentially) many primes $p$.
  - Typically, $p$ is chosen at random among all $n$-bit primes
  - Assumption is still average-case: DLOG is hard for random $p$. 
Lattice Assumption

The complexity of solving lattice problems in $n$-dimensional lattices grows superpolynomially (or exponentially) in $n$.

- Similarly, one may conjecture that the complexity of DLOG grows superpolynomially in $n = \log p$ or $n = \log |G|$.
- This is not the same:
  - For any $n$, there are (exponentially) many primes $p$.
  - Typically, $p$ is chosen at random among all $n$-bit primes
  - Assumption is still average-case: DLOG is hard for random $p$.
- We do not know how to reduce $DLOG(\mathbb{Z}_p^*)$ to $DLOG(\mathbb{Z}_q^*)$.
  RSR provides no guidance on how to choose $p$. 
Alternative assumption

DLOG($p_n$) is hard when $p_n$ is the smallest prime $> 2^n$.

- Equivalent to worst-case family of problems (indexed by $n$)
- Ad-hoc: problem definition seems rather arbitrary
Alternative assumption

DLOG($p_n$) is hard when $p_n$ is the smallest prime $> 2^n$.

- Equivalent to worst-case family of problems (indexed by $n$)
- Ad-hoc: problem definition seems rather arbitrary

There is more:

- Lattice problems in dimension $n$ reduce to lattice problems in dimension $m > n$:

  \[
  \begin{array}{c|c|c}
  B & O & \infty \\
  \hline
  O & \infty & \infty
  \end{array}
  \]

- No such reduction for DLOG:

  \[
  DLOG(p_n) \not\rightarrow DLOG(p_{n+1})
  \]
Other (natural) representations:

\[ G = (\mathbb{Z}_p^*, \cdot) \equiv (\mathbb{Z}_{p-1}, +) \]

but “DLOG” in \((\mathbb{Z}_{p-1}, +)\) is easy.

Other (still natural) groups:

\[ G = \mathbb{Z}^*_p \]
Other (natural) representations:

\[ G = (\mathbb{Z}_p^*, \cdot) \equiv (\mathbb{Z}_{p-1}, +) \]

but “DLOG” in \((\mathbb{Z}_{p-1}, +)\) is easy.

Other (still natural) groups:

\[ G = \mathbb{Z}_{pq}^* \]

**Question**

Assume one of \(DLOG(\mathbb{Z}_p)\) and \(DLOG(\mathbb{Z}_{p,q})\) is polynomial time solvable, and one is not. Which group family would you choose?
Other (natural) representations:

\[ G = (\mathbb{Z}_p^*, \cdot) \equiv (\mathbb{Z}_{p-1}, +) \]

but “DLOG” in \((\mathbb{Z}_{p-1}, +)\) is easy.

Other (still natural) groups:

\[ G = \mathbb{Z}_{pq} \]

**Question**

Assume one of \(DLOG(\mathbb{Z}_p)\) and \(DLOG(\mathbb{Z}_{p\cdot q})\) is polynomial time solvable, and one is not. Which group family would you choose?

Chinese Reminder Theorem (CRT): \(\mathbb{Z}_{pq} \approx \mathbb{Z}_p \times \mathbb{Z}_q\)

\[ DLOG(\mathbb{Z}_p^*) \implies DLOG(\mathbb{Z}_{pq}^*). \]

Reduction in the other direction requires factoring.
Ajtai’s one-way function (SIS)

- Parameters:  $m, n, q \in \mathbb{Z}$
- Key:  $A \in \mathbb{Z}_q^{n \times m}$
- Input:  $x \in \{0, 1\}^m$

**Theorem (A'96)**

For $m > n \log q$, if lattice problems (SIVP) are hard to approximate in the worst-case, then $f_A(x) = Ax \mod q$ is a one-way function.

Applications: OWF [A'96], Hashing [GGH'97], Commit [KTX'08], IDs schemes [L'08], Signatures [LM'08,GPV'08, . . . ,DDLL'13] . . .
Ajtai’s one-way function (SIS)

- Parameters: $m, n, q \in \mathbb{Z}$
- Key: $A \in \mathbb{Z}_q^{n \times m}$
- Input: $x \in \{0, 1\}^m$
- Output: $f_A(x) = Ax \mod q$

Theorem (A'96)
For $m > n \lg q$, if lattice problems (SIVP) are hard to approximate in the worst-case, then $f_A(x) = Ax \mod q$ is a one-way function.

Applications: OWF [A'96], Hashing [GGH'97], Commit [KTX'08], IDs schemes [L'08], Signatures [LM'08, GPV'08, ... , DDLL'13] ...
Ajtai’s one-way function (SIS)

- Parameters: \( m, n, q \in \mathbb{Z} \)
- Key: \( A \in \mathbb{Z}_q^{n \times m} \)
- Input: \( x \in \{0, 1\}^m \)
- Output: \( f_A(x) = Ax \mod q \)

Theorem (A’96)

For \( m > n \log_2 q \), if lattice problems (SIVP) are hard to approximate in the worst-case, then \( f_A(x) = Ax \mod q \) is a one-way function.

Applications: OWF [A’96], Hashing [GGH’97], Commit [KTX’08], ID schemes [L’08], Signatures [LM’08, GPV’08, …, DDLL’13] …
Relation to lattices

- The kernel set $\Lambda^\perp(A)$ is a lattice
  \[ \Lambda^\perp(A) = \{ z \in \mathbb{Z}^m : Az = 0 \pmod{q} \} \]

- Collisions $Ax = Ay \pmod{q}$ can be represented by a single vector $z = x - y \in \{-1, 0, 1\}$ such that
  \[ z = x - y \]
Relation to lattices

- The kernel set $\Lambda^\perp(A)$ is a lattice

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

- Collisions $Ax = Ay \pmod{q}$ can be represented by a single vector $z = x - y \in \{-1, 0, 1\}$ such that

$$Az = Ax - Ay = 0 \pmod{q}$$
Relation to lattices

- The kernel set $\Lambda^\perp(A)$ is a lattice

\[ \Lambda^\perp(A) = \{ z \in \mathbb{Z}^m : Az = 0 \pmod{q} \} \]

- Collisions $Ax = Ay \pmod{q}$ can be represented by a single vector $z = x - y \in \{-1, 0, 1\}$ such that

\[ Az = Ax - Ay = 0 \pmod{q} \]

- Collisions are lattice vectors $z \in \Lambda^\perp(A)$ with small norm

\[ \|z\|_\infty = \max_i |z_i| = 1. \]
The kernel set $\Lambda_{\perp}(A)$ is a lattice

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

Collisions $Ax = Ay \pmod{q}$ can be represented by a single vector $z = x - y \in \{-1, 0, 1\}$ such that

$$Az = Ax - Ay = 0 \pmod{q}$$

Collisions are lattice vectors $z \in \Lambda_{\perp}(A)$ with small norm

$$\|z\|_{\infty} = \max_i |z_i| = 1.$$
**Definition (Shortest Vector Problem, SVP)**

Given a lattice $\mathcal{L}(B)$, find a (nonzero) lattice vector $Bx$ (with $x \in \mathbb{Z}^k$) of length (at most) $\|Bx\| \leq \lambda_1$
Definition (Shortest Vector Problem, SVP)

Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \lambda_1$
Definition (Shortest Vector Problem, SVP)

Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \lambda_1$
Definition (Shortest Vector Problem, $\text{SVP}_\gamma$)

Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \gamma \lambda_1$
Definition (Closest Vector Problem, CVP)

Given a lattice $\mathcal{L}(B)$ and a target point $t$, find a lattice vector $Bx$ within distance $\|Bx - t\| \leq \mu$ from the target.
Definition (Closest Vector Problem, CVP)

Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \mu$ from the target.
Definition (Closest Vector Problem, CVP)

Given a lattice $\mathcal{L}(B)$ and a target point $t$, find a lattice vector $Bx$ within distance $\|Bx - t\| \leq \mu$ from the target.
Definition (Closest Vector Problem, CVP$_\gamma$)

Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \gamma \mu$ from the target.
Definition (Shortest Independent Vectors Problem, SIVP)

Given a lattice $\mathcal{L}(\mathbf{B})$, find $n$ linearly independent lattice vectors $\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n$ of length (at most) $\max_i \|\mathbf{Bx}_i\| \leq \lambda_n$.
Definition (Shortest Independent Vectors Problem, SIVP)

Given a lattice $\mathcal{L}(B)$, find $n$ linearly independent lattice vectors $Bx_1, \ldots, Bx_n$ of length (at most) $\max_i \|Bx_i\| \leq \lambda_n$.
Definition (Shortest Independent Vectors Problem, SIVP)

Given a lattice $\mathcal{L}(B)$, find $n$ linearly independent lattice vectors $Bx_1, \ldots, Bx_n$ of length (at most) $\max_i \|Bx_i\| \leq \lambda_n$
Definition (Shortest Independent Vectors Problem, SIVP\(\gamma\))

Given a lattice \(\mathcal{L}(\mathbf{B})\), find \(n\) linearly independent lattice vectors \(\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n\) of length (at most) \(\max_i \|\mathbf{Bx}_i\| \leq \gamma \lambda_n\)
Minimum Distance and Successive Minima

- Minimum distance

\[ \lambda_1 = \min_{x, y \in \mathcal{L}, x \neq y} \| x - y \| \]
\[ = \min_{x \in \mathcal{L}, x \neq 0} \| x \| \]

Examples:
- \( \mathbb{Z}_n \): \( \lambda_1 = \lambda_2 = \ldots = \lambda_n = 1 \)
- Always: \( \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \)
Minimum Distance and Successive Minima

- Minimum distance

\[ \lambda_1 = \min_{x, y \in \mathcal{L}, x \neq y} \|x - y\| \]

\[ = \min_{x \in \mathcal{L}, x \neq 0} \|x\| \]
Minimum Distance and Successive Minima

- **Minimum distance**

\[
\lambda_1 = \min_{x, y \in \mathcal{L}, x \neq y} \|x - y\|
\]

\[
= \min_{x \in \mathcal{L}, x \neq 0} \|x\|
\]

- **Successive minima** \((i = 1, \ldots, n)\)

\[
\lambda_i = \min \{ r : \dim \text{span}(B(r) \cap \mathcal{L}) \geq i \}
\]
Minimum Distance and Successive Minima

- **Minimum distance**

  \[ \lambda_1 = \min_{x, y \in \mathcal{L}, x \neq y} \| x - y \| \]

  \[ = \min_{x \in \mathcal{L}, x \neq 0} \| x \| \]

- **Successive minima** \((i = 1, \ldots, n)\)

  \[ \lambda_i = \min \{ r : \dim \text{span}(B(r) \cap \mathcal{L}) \geq i \} \]
Minimum Distance and Successive Minima

- **Minimum distance**

  $$\lambda_1 = \min_{x, y \in \mathcal{L}, x \neq y} \|x - y\| = \min_{x \in \mathcal{L}, x \neq 0} \|x\|$$

- **Successive minima** ($i = 1, \ldots, n$)

  $$\lambda_i = \min \{ r : \dim \text{span}(B(r) \cap \mathcal{L}) \geq i \}$$

- **Examples**
  - $\mathbb{Z}^n$: $\lambda_1 = \lambda_2 = \ldots = \lambda_n = 1$
  - Always: $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered. How much noise is needed?

\[
\|r\| \leq (\log n) \cdot \sqrt{n} \cdot \lambda^{n/2}
\]

Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in \Lambda$ and $\|r\| \approx \sqrt{n} \lambda^{n/2}$. $a \in \mathbb{R}^n / \Lambda$ is uniformly distributed.
Blurring a lattice

Consider a lattice Λ, and add noise to each lattice point until the entire space is covered.

Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in \Lambda$ and $\|r\| \approx \sqrt{n \lambda n}$. $a \in \mathbb{R}^n / \Lambda$ is uniformly distributed.
Blurring a lattice

Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered.
Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered.

How much noise is needed?

$$\|r\| \leq (\log n) \cdot \sqrt{n \cdot \lambda_n / 2}$$

Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in \Lambda$ and $\|r\| \approx \sqrt{n \lambda_n}$.

$a \in \mathbb{R}^n / \Lambda$ is uniformly distributed.
Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered.

How much noise is needed?

$$\|r\| \leq \sqrt{n} \cdot \lambda_n / 2$$

- Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in L$ and $\|r\| \approx \sqrt{n} \lambda_n$. 
Blurring a lattice

Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

How much noise is needed?

$$\|r\| \leq \sqrt{n} \cdot \lambda_n / 2$$

- Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in L$ and $\|r\| \approx \sqrt{n} \lambda_n$. 
Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

How much noise is needed?

$$\|r\| \leq \sqrt{n} \cdot \lambda_n/2$$

- Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in \mathcal{L}$ and $\|r\| \approx \sqrt{n} \lambda_n$. 

Daniele Micciancio (UCSD)

Lattice Cryptography: Introduction and Open Problems

August 2015 16 / 32
Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

**How much noise is needed?**

$$\|r\| \leq \sqrt{n} \cdot \lambda_n / 2$$

- Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in \mathcal{L}$ and $\|r\| \approx \sqrt{n} \lambda_n$. 

Daniele Micciancio (UCSD) Lattice Cryptography: Introduction and Open Problems August 2015 16 / 32
Blurring a lattice

Consider a lattice $\Lambda$, and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

How much noise is needed?

$$\|r\| \leq \sqrt{n} \cdot \lambda_n/2$$

- Each point in $a \in \mathbb{R}^n$ can be written $a = v + r$ where $v \in L$ and $\|r\| \approx \sqrt{n} \lambda_n$. 

Daniele Micciancio (UCSD) Lattice Cryptography: Introduction and Open Problems August 2015 16 / 32
Consider a lattice \( \Lambda \), and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

**How much noise is needed? [MR]**

\[ \|r\| \leq (\log n) \cdot \sqrt{n} \cdot \lambda_n/2 \]

- Each point in \( a \in \mathbb{R}^n \) can be written \( a = v + r \) where \( v \in \mathcal{L} \) and \( \|r\| \approx \sqrt{n} \lambda_n \).
- \( a \in \mathbb{R}^n/\Lambda \) is uniformly distributed.
Blurring a lattice

Consider a lattice \( \Lambda \), and add noise to each lattice point until the entire space is covered. Increase the noise until the space is uniformly covered.

**How much noise is needed? [MR]**

\[
\|r\| \leq (\log n) \cdot \sqrt{n} \cdot \lambda_n / 2
\]

- Each point in \( a \in \mathbb{R}^n \) can be written \( a = v + r \) where \( v \in L \) and \( \|r\| \approx \sqrt{n} \lambda_n \).
- \( a \in \mathbb{R}^n / \Lambda \) is uniformly distributed.
- Think of \( \mathbb{R}^n \approx \frac{1}{q} \Lambda \) [GPV’07]
Average-case hardness (sketch)

- Generate random points $\mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i \in \frac{1}{q}\Lambda$, where
  - $\mathbf{v}_i \in \Lambda$ is a random lattice point
  - $\mathbf{r}_i$ is a random error vector of length $\|\mathbf{r}_i\| \approx \sqrt{n}\lambda_n$
- $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \approx \frac{1}{q}\Lambda^m \equiv \mathbb{Z}_q^{n \times m}$
- Assume we can find a short lattice vector $\mathbf{z} \in \mathbb{Z}^m$

$$\mathbf{A}\mathbf{z} = \mathbf{0}$$
Average-case hardness (sketch)

- Generate random points \( a_i = v_i + r_i \in \frac{1}{q} \Lambda \), where
  - \( v_i \in \Lambda \) is a random lattice point
  - \( r_i \) is a random error vector of length \( \|r_i\| \approx \sqrt{n} \lambda_n \)
- \( A = [a_1, \ldots, a_m] \approx \frac{1}{q} \Lambda^m \equiv \mathbb{Z}_q^{n \times m} \)
- Assume we can find a short lattice vector \( z \in \mathbb{Z}^m \)
  \[
  \sum (v_i + r_i)z_i = \sum a_i z_i = Az = 0
  \]
Average-case hardness (sketch)

- Generate random points \( \mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i \in \frac{1}{q} \Lambda \), where
  - \( \mathbf{v}_i \in \Lambda \) is a random lattice point
  - \( \mathbf{r}_i \) is a random error vector of length \( \|\mathbf{r}_i\| \approx \sqrt{n \lambda_n} \)
- \( \mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \approx \frac{1}{q} \Lambda^m \equiv \mathbb{Z}^{n \times m}_q \)
- Assume we can find a short lattice vector \( \mathbf{z} \in \mathbb{Z}^m \)

\[
\sum (\mathbf{v}_i + \mathbf{r}_i)z_i = \sum \mathbf{a}_i z_i = \mathbf{A}\mathbf{z} = \mathbf{0}
\]

- Rearranging the terms yields a lattice vector

\[
\sum \mathbf{v}_i z_i = -\sum \mathbf{r}_i z_i
\]

of length at most \( \|\sum \mathbf{r}_i z_i\| \approx \sqrt{m} \cdot \max \|\mathbf{r}_i\| \approx n \cdot \lambda_n \)
Shortcomings of Ajtai’s function

Expressivity:

- Ajtai’s proof requires $m > n \log q$
- The function $f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n$ is not injective
- Enough for one-way functions, collision resistant hashing, some digital signatures, commitments, identification, etc.
- ... but (public key) encryption seem to require stronger assumptions.
- **1996**: Ajtai-Dwork cryptosystem, based on the “unique” Shortest Vector Problem.

Efficiency:

- The matrix/key $A \in \mathbb{Z}_q^{n \times m}$ requires $\Omega(n^2)$ storage (and computation)
- **1996**: NTRU Cryptosystem, efficient, but not supported by security proof from worst-case lattice problems.
Learning with errors (LWE)

- \( A \in \mathbb{Z}_q^{m \times n}, \ s \in \mathbb{Z}_q^n, \ e \in \mathcal{E}^m. \)
- \( g_A(s, e) = As \mod q \)
Learning with errors (LWE)

- $A \in \mathbb{Z}_q^{m \times n}$, $s \in \mathbb{Z}_q^n$, $e \in \mathcal{E}^m$.
- $g_A(s; e) = As + e \mod q$
- Learning with Errors: Given $A$ and $g_A(s, e)$, recover $s$.
Learning with errors (LWE)

- \( A \in \mathbb{Z}_q^{m \times n}, \ s \in \mathbb{Z}_q^n, \ e \in \mathcal{E}^m \).
- \( g_A(s; e) = As + e \mod q \)
- Learning with Errors: Given \( A \) and \( g_A(s, e) \), recover \( s \).

**Theorem (Regev’05)**

The function \( g_A(s, e) \) is hard to invert on the average, assuming SIVP is hard to approximate in the worst-case even for quantum computers.
Candidate OWF

Key: a hard lattice $\mathcal{L}$

Input: $x$, $\|x\| \leq \beta$

Output: $f_{\mathcal{L}}(x) = x \mod \beta$

$\beta < \lambda^{1/2}$: $f_{\mathcal{L}}$ is injective

$\beta > \lambda^{1/2}$: $f_{\mathcal{L}}$ is not injective

$\beta \geq \mu$: $f_{\mathcal{L}}$ is surjective

$\beta \gg \mu$: $f_{\mathcal{L}}(x)$ is almost uniform

Question: Are these functions cryptographically hard to invert?
Candidate OWF

Key: a hard lattice $\mathcal{L}$
Input: $x$, $\|x\| \leq \beta$
Output: $f_{\mathcal{L}}(x) = x \mod \mathcal{L}$

Question: Are these functions cryptographically hard to invert?
**Candidate OWF**

Key: a hard lattice $\mathcal{L}$

Input: $x$, $\|x\| \leq \beta$

Output: $f_\mathcal{L}(x) = x \mod \mathcal{L}$

- $\beta < \lambda_1/2$: $f_\mathcal{L}$ is injective

- $\beta > \lambda_1/2$: $f_\mathcal{L}$ is not injective

- $\beta \geq \mu$: $f_\mathcal{L}$ is surjective

- $\beta \gg \mu$: $f_\mathcal{L}(x)$ is almost uniform
Candidate OWF

Key: a hard lattice $\mathcal{L}$

Input: $x$, $\|x\| \leq \beta$

Output: $f_{\mathcal{L}}(x) = x \mod \mathcal{L}$

- $\beta < \lambda_1/2$: $f_{\mathcal{L}}$ is injective
- $\beta > \lambda_1/2$: $f_{\mathcal{L}}$ is not injective
Candidate OWF

Key: a hard lattice $\mathcal{L}$

Input: $x, \|x\| \leq \beta$

Output: $f_\mathcal{L}(x) = x \text{ mod } \mathcal{L}$

- $\beta < \lambda_1 / 2$: $f_\mathcal{L}$ is injective
- $\beta > \lambda_1 / 2$: $f_\mathcal{L}$ is not injective
- $\beta \geq \mu$: $f_\mathcal{L}$ is surjective

$\beta \gg \mu$: $f_\mathcal{L}(x)$ is almost uniform
Candidate OWF

Key: a hard lattice $\mathcal{L}$

Input: $x, \|x\| \leq \beta$

Output: $f_\mathcal{L}(x) = x \mod \mathcal{L}$

- $\beta < \lambda_1/2$: $f_\mathcal{L}$ is injective
- $\beta > \lambda_1/2$: $f_\mathcal{L}$ is not injective
- $\beta \geq \mu$: $f_\mathcal{L}$ is surjective
- $\beta \gg \mu$: $f_\mathcal{L}(x)$ is almost uniform
Candidate OWF

Key: a hard lattice $\mathcal{L}$
Input: $x$, $\|x\| \leq \beta$
Output: $f_\mathcal{L}(x) = x \mod \mathcal{L}$

- $\beta < \lambda_1/2$: $f_\mathcal{L}$ is injective
- $\beta > \lambda_1/2$: $f_\mathcal{L}$ is not injective
- $\beta \geq \mu$: $f_\mathcal{L}$ is surjective
- $\beta \gg \mu$: $f_\mathcal{L}(x)$ is almost uniform

Question

Are these functions cryptographically hard to invert?
Special Versions of CVP

Definition (Closest Vector Problem (CVP))

Given $(\mathcal{L}, t, d)$, with $\mu(t, \mathcal{L}) \leq d$, find a lattice point within distance $d$ from $t$.

- If $d$ is arbitrary, then one can find the closest lattice vector by binary search on $d$.
- Bounded Distance Decoding (BDD): If $d < \lambda_1(\mathcal{L})/2$, then there is at most one solution. Solution is the closest lattice vector.
- Absolute Distance Decoding (ADD): If $d \geq \rho(\mathcal{L})$, then there is always at least one solution. Solution may not be closest lattice vector.
Computational problems on random lattices

Ajtai’s class of random lattices an their duals:

\[ \mathbf{A} \in \mathbb{Z}^{n \times m} \]

\[ \Lambda_q^\perp(\mathbf{A}) = \{ \mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \mod q \} \]

\[ \Lambda_q(\mathbf{A}) = \mathbf{A}^T\mathbb{Z}^n + q\mathbb{Z}^m \]

Inverting Ajtai’s function \( \mathbf{A}\mathbf{x} = \mathbf{b} \)

- Solution \( \mathbf{x} \) always exist, but it is hard to find
- Average case version of ADD on random \( \Lambda_q^\perp(\mathbf{A}) \)

Solving LWE \( \mathbf{sA} + \mathbf{x} = \mathbf{b} \)

- For small enough \( \mathbf{x} \), solution is unique
- Average case version of BDD on random dual lattice \( \Lambda_q(\mathbf{A}) \).
ADD reduces to SIVP

ADD input: \( \mathcal{L} \) and arbitrary \( t \)

- Compute short vectors \( V = \text{SIVP}(\mathcal{L}) \)
- Use \( V \) to find a lattice vector within distance
\[
\sum_i \frac{1}{2} \| v_i \| \leq (n/2)\lambda_n \leq n\rho \text{ from } t
\]
BDD reduces to SIVP

BDD input: \( t \) close to \( \mathcal{L} \)
BDD reduces to SIVP

BDD input: \( \mathbf{t} \) close to \( \mathcal{L} \)

- Compute \( \mathbf{V} = \text{SIVP}(\mathcal{L}^*) \)

For each \( v_i \in \mathcal{L}^* \), find the layer \( \mathcal{L}_i = \{ x | x \cdot v_i = c_i \} \) closest to \( \mathbf{t} \)

Output \( \mathcal{L}_1 \cap \mathcal{L}_2 \cap \cdots \cap \mathcal{L}_n \)

Output is correct as long as

\[
\mu(\mathbf{t}, \mathcal{L}) \leq \lambda_1^2 \leq \lambda_2 \leq \frac{1}{2} \| v_i \|_0
\]
BDD reduces to SIVP

BDD input: \( t \) close to \( \mathcal{L} \)

- Compute \( \mathbf{V} = \text{SIVP}(\mathcal{L}^*) \)
- For each \( \mathbf{v}_i \in \mathcal{L}^* \), find the layer
  \( L_i = \{ \mathbf{x} \mid \mathbf{x} \cdot \mathbf{v}_i = c_i \} \) closest to \( t \)

\[ \text{Output} = L_1 \cap L_2 \cap \ldots \cap L_n \]

Output is correct as long as
\[ \mu(t, \mathcal{L}) \leq \lambda_1^2 n \leq \frac{1}{2} \| \mathbf{v}_i \|_0 \]
BDD reduces to SIVP

BDD input: \( \mathbf{t} \) close to \( \mathcal{L} \)

- Compute \( \mathbf{V} = \text{SIVP}(\mathcal{L}^*) \)
- For each \( \mathbf{v}_i \in \mathcal{L}^* \), find the layer \( L_i = \{ \mathbf{x} | \mathbf{x} \cdot \mathbf{v}_i = c_i \} \) closest to \( \mathbf{t} \)
- Output \( L_1 \cap L_2 \cap \cdots \cap L_n \)
BDD reduces to SIVP

BDD input: \( t \) close to \( \mathcal{L} \)
- Compute \( \mathbf{V} = \text{SIVP}(\mathcal{L}^*) \)
- For each \( \mathbf{v}_i \in \mathcal{L}^* \), find the layer \( L_i = \{ \mathbf{x} \mid \mathbf{x} \cdot \mathbf{v}_i = c_i \} \) closest to \( t \)
- Output \( L_1 \cap L_2 \cap \cdots \cap L_n \)
- Output is correct as long as

\[
\mu(t, \mathcal{L}) \leq \frac{\lambda_1}{2n} \leq \frac{1}{2\lambda_n^*} \leq \frac{1}{2\|\mathbf{v}_i\|}
\]
Special Versions of SVP and SIVP

- **GapSVP**: compute (or approximate) the value $\lambda_1$ without necessarily finding a short vector
- **GapSIVP**: compute (or approximate) the value $\lambda_n$ without necessarily finding short linearly independent vectors
- Transference Theorem $\lambda_1 \approx 1/\lambda^*_n$: GapSVP can be (approximately) solved by solving GapSIVP in the dual lattice, and vice versa

Problems

- **Exercise**: Computing $\lambda_1$ (or $\lambda_n$) exactly is as hard as SVP (or SIVP)
- **Open Problem**: Reduce approximate SVP (or SIVP) to approximate GapSVP (or GapSIVP)
Relations among lattice problems

- $\text{SIVP} \approx \text{ADD}$ [MG’01]
- $\text{SVP} \leq \text{CVP}$ [GMSS’99]
- $\text{SIVP} \leq \text{CVP}$ [M’08]
- $\text{BDD} \preceq \text{SIVP}$
- $\text{CVP} \preceq \text{SVP}$ [L’87]
- $\text{GapSVP} \approx \text{GapSIVP}$ [LLS’91, B’93]
- $\text{GapSVP} \preceq \text{BDD}$ [LM’09]
Relations among lattice problems

- SIVP \approx ADD [MG’01]
- SVP \leq CVP [GMSS’99]
- SIVP \leq CVP [M’08]
- BDD \approx SIVP
- CVP \approx SVP [L’87]
- GapSVP \approx GapSIVP [LLS’91, B’93]
- GapSVP \leq BDD [LM’09]
Open Problems

- Does the ability to approximate $\lambda_1$ helps in solving SVP?
- Does the ability to approximate $\lambda_n$ helps in solving SIVP?
- Is there a reduction from CVP/SVP to SIVP?
  - Yes, for the exact version of the problems [M. 08]
  - Open for approximation version
- Is there a classical (nonquantum) reduction from SIVP/ADD to GapSVP/BDD?
Efficient Lattice Cryptography from Structured Lattices

Idea

Use structured matrix

\[ A = [A^{(1)} | \ldots | A^{(m/n)}] \]

where \( A^{(i)} \in \mathbb{Z}_q^{n \times n} \) is circulant

\[ A^{(i)} = \begin{bmatrix} a_1^{(i)} & a_n^{(i)} & \ldots & a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \ldots & a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ a_n^{(i)} & a_{n-1}^{(i)} & \ldots & a_1^{(i)} \end{bmatrix} \]

- “Generalized Compact Knapsacks and Efficient One-Way Functions” (Micciancio, FOCS 2002)
- Efficient version of Ajtai’s connection:
  - \( O(n \log n) \) space and time complexity
  - Provable security: guidance on how to choose random instances.

Theorem

“CyclicSIS” is hard to invert on average, assuming the worst-case hardness of lattice problems over “cyclic” lattices.
Ideal Lattices and Algebraic number theory

- Isomorphism: $\mathbb{A}^{\text{cyc}} \leftrightarrow \mathbb{Z}[X]/(X^n - 1)$
- Cyclic SIS:

$$f_{a_1, \ldots, a_k}(u_1, \ldots, u_k) = \sum_{i} a_i(X) \cdot u_i(X) \pmod{X^n - 1}$$

where $a_i, u_i \in R = \mathbb{Z}[X]/(X^n - 1)$.
- More generally, use $R = \mathbb{Z}[X]/p(X)$ for some monic polynomial $p(X) \in \mathbb{Z}[X]$
- If $p(X)$ is irreducible, then finding collisions to $f_a$ for random $a$ is as hard as solving lattice problems in the worst case in ideal lattices
- Can set $R$ to the ring of integers of $K = \mathbb{Q}[X]/p(X)$. 
How to choose $p(X)/R$?

RingSIS (Lyubashevsky, PhD Thesis, UCSD 2008)

- define $f_a(u) = \sum_i a_i(X) \cdot u_i(X)$
- Notice: no reduction modulo $p(X)$!
- If $f_a(u) = f_a(u')$ in $\mathbb{Z}[X]$, then $f_a(u) = f_a(u') \pmod{p(X)}$.
- Conclusion: breaking $f$ is at least as hard as solving lattices problems in ideal lattices for any $p(X)$.
How to choose \( p(X)/R \)?

RingSIS (Lyubashevsky, PhD Thesis, UCSD 2008)

- define \( f_a(u) = \sum_i a_i(X) \cdot u_i(X) \)
- Notice: no reduction modulo \( p(X) \)!
- If \( f_a(u) = f_a(u') \) in \( \mathbb{Z}[X] \), then \( f_a(u) = f_a(u') \pmod{p(X)} \).
- Conclusion: breaking \( f \) is at least as hard as solving lattices problems in ideal lattices for any \( p(X) \).

RingLWE:

- Most applications require not only hardness of inverting \( f_a \), but also pseudorandomness of output \( f_a(u) \)
- [Lyubashevsky,Peikert,Regev’10]: For cyclotomic \( p(X) \), hardness of inverting \( f_a \) implies pseudorandomness of \( f_a(u) \).
- [Lauter’15] constructs polynomial rings where inverting \( f_a \) is conceivably hard, but \( f_a(u) \) is easily distinguished from random.
[P’09, BLPRS’13] There is a classical reduction from GapSVP to LWE when \( q = 2^{O(n)} \), or LWE dimension \( d = O(n^2) \)

Open Problems

- Is there a more efficient reduction from GapSVP to LWE?
- Is there a classical reduction from SIVP to LWE?
- Is there a reduction from SVP/SIVP to LWE on ideal lattices?
More Open Problems – Tonight 7:30pm

- Bring your own open problems to share!
- Send email to daniele@cs.ucsd.edu with estimated time for scheduling.
- ...or, just talk to me over lunch or coffee break.
More Open Problems – Tonight 7:30pm

- Bring your own open problems to share!
- Send email to daniele@cs.ucsd.edu with estimated time for scheduling.
- ...or, just talk to me over lunch or coffee break.

Thank you!