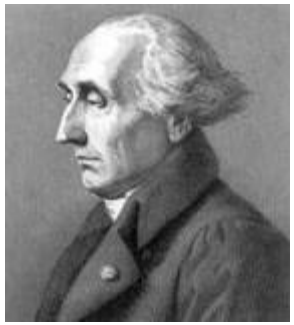
Introduction to Lattices

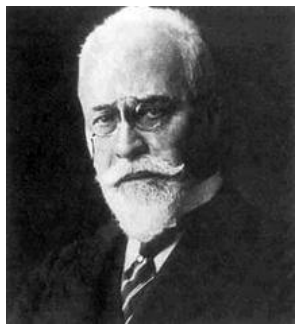
Daniele Micciancio

July 2024

(Point) Lattices

- Traditional area of mathematics
  - Bridge between number theory and geometry
  - Studied by Lagrange, Gauss, ..., Minkowski, ...





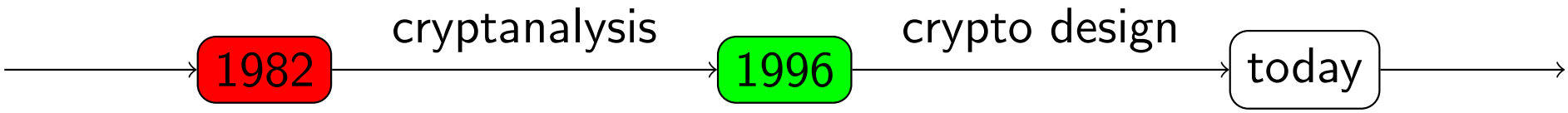
- Many applications in computer science and cryptography
  - Cryptanalysis: breaking low-exponent RSA
  - Coding Theory: error correcting codes for wireless communication
  - Optimization: Integer Programming
  - Cryptography: post-quantum cryptography and much more

Outline

(Post-quantum) Lattice-based Cryptography

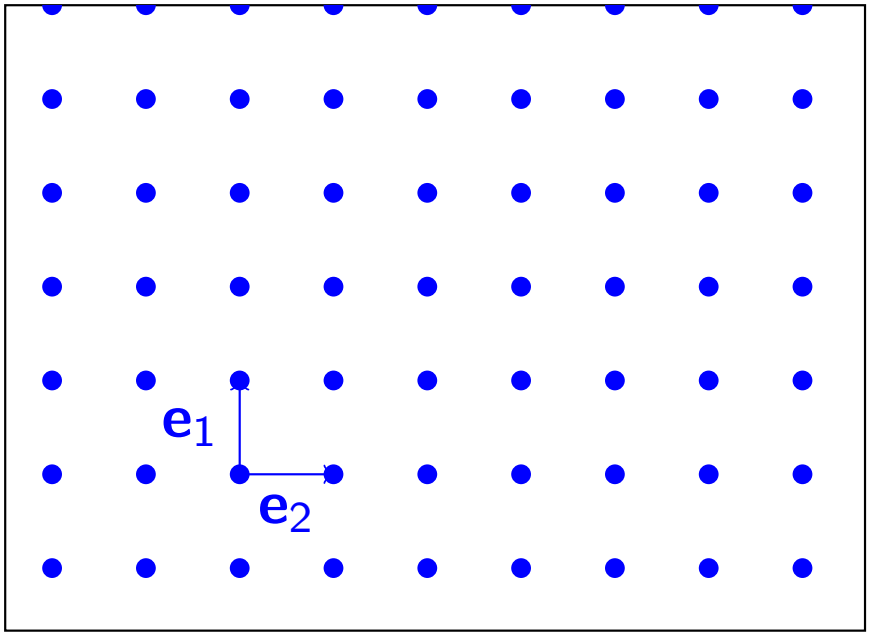
- Post-quantum cryptography
  - Can be used on conventional (non-quantum) computers
  - Remain secure in the face of quantum attacks
- Lattice problems believed to be hard even for quantum computers
- NIST Post-Quantum Cryptography standardization
  - Process started in 2016
  - A good half of the ~ 80 submissions based on lattices
  - 2023: lattice-based encryption and signatures picked as PQC standard
- Many other desirable properties of lattice-based cryptography
  - Fast and Parallelizable
  - Versatile: many advanced applications
  - Only known solution for some: Fully Homomorphic Encryption

Lattice Cryptography: a Timeline



- Lenstra, Lenstra, Lovasz (1982) : The "LLL" algorithm
- Ajtai (1996) : Hardness of "Short Integer Solution" (SIS) problem
  - Ajtai, Dwork (1997): Pubic Key Encryption
  - Hoffstein, Pipher, Silverman (1998): NTRU cryptosystem
- M. (2002) : "Generalized compact knapsacks" (RingSIS)
  - Efficient version of Ajtai's construction, based on structured lattices
- Regev (2005) : Hardness of "Learning with Errors" (LWE)
  - Injective variant of Ajtai's SIS, with wider range of applications
- ... RingLWE ... Fully Homomorphic Encryption ... Lattice Trapdoors
- 2016-2023: NIST Post-Quantum Cryptography Standardization

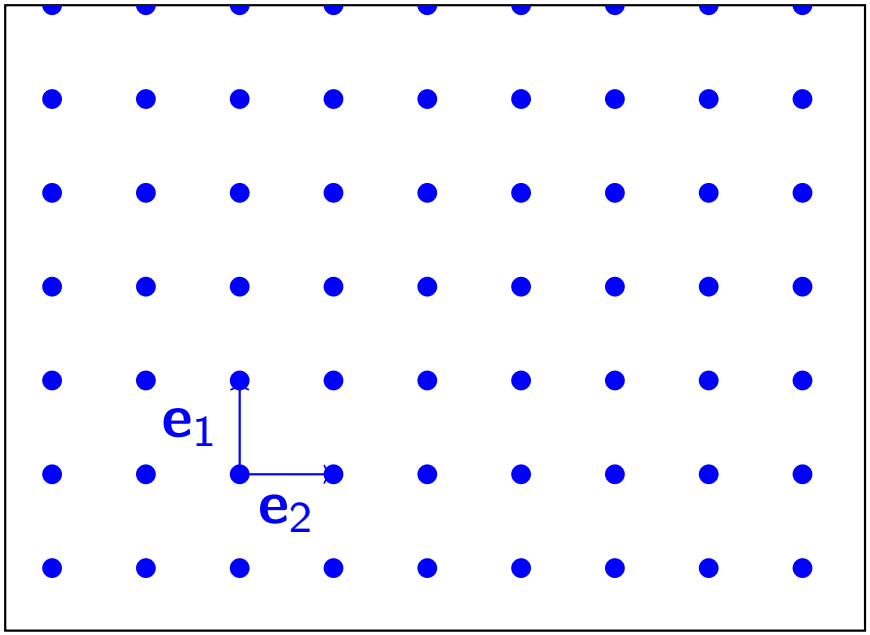
Lattices: Definition



The simplest lattice in n-dimensional space is the integer lattice

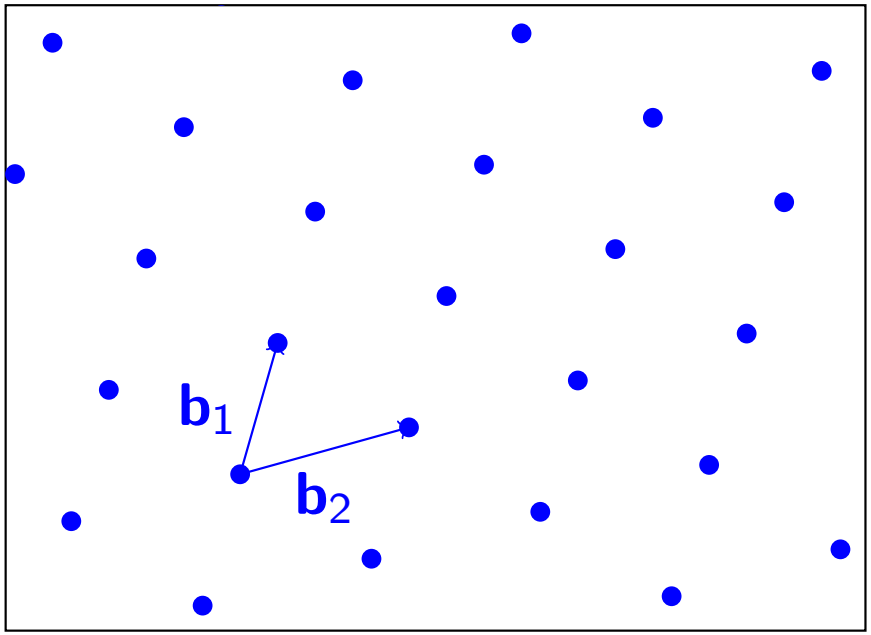
$$\Lambda = \mathbb{Z}^n$$

Lattices: Definition



The simplest lattice in n-dimensional space is the integer lattice

$$\Lambda = \mathbb{Z}^n$$



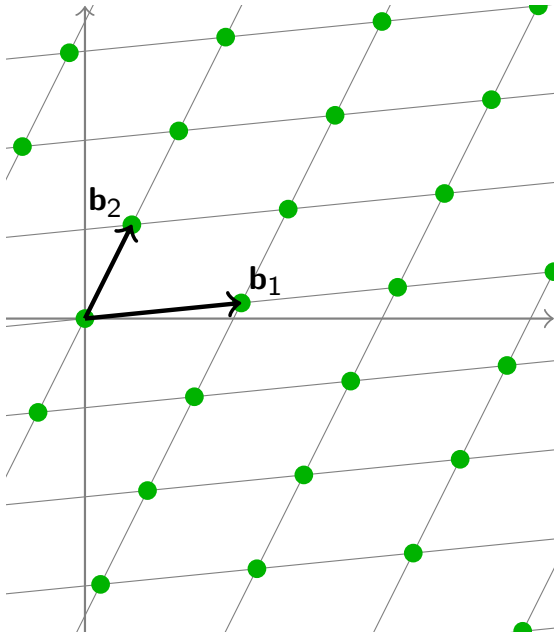
Other lattices are obtained by applying a linear transformation

$$\Lambda = \mathbf{B}\mathbb{Z}^n \qquad (\mathbf{B} \in \mathbb{R}^{d \times n})$$

Lattices and Bases

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$:

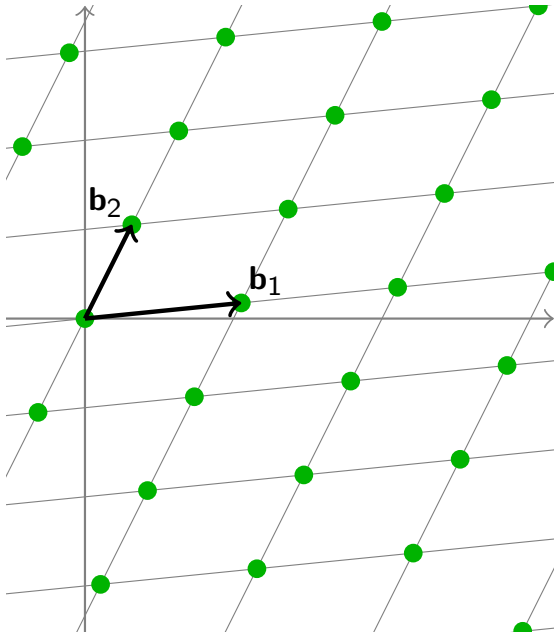
$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z}$$



Lattices and Bases

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} \colon \mathbf{x} \in \mathbb{Z}^n\}$$



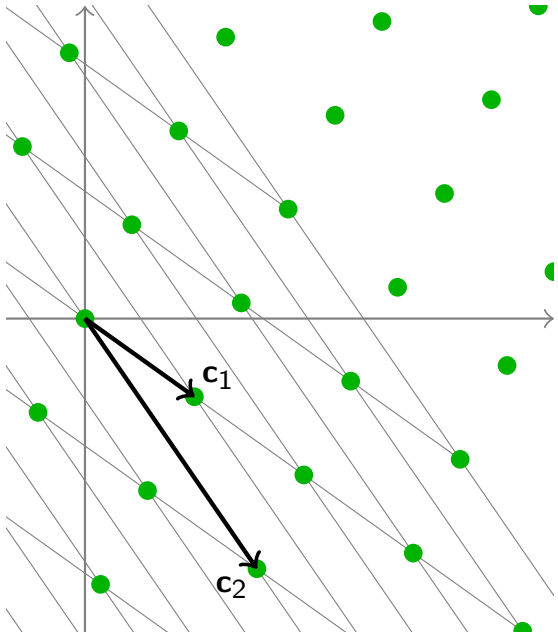
Lattices and Bases

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} \colon \mathbf{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}$$



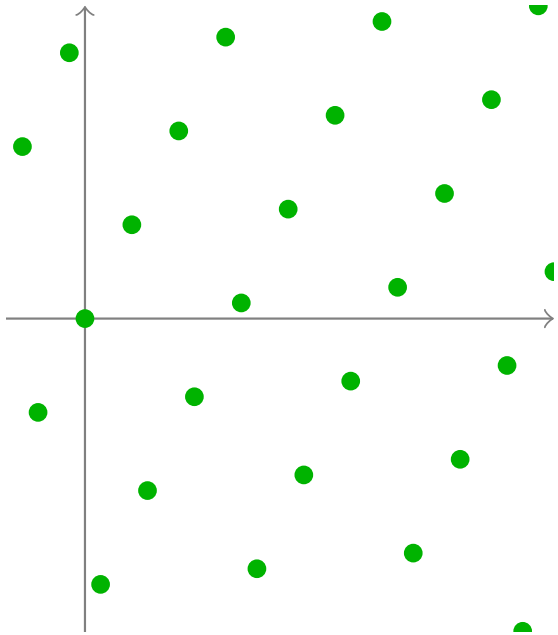
Lattices and Bases

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^n$:

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

The same lattice has many bases

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}$$



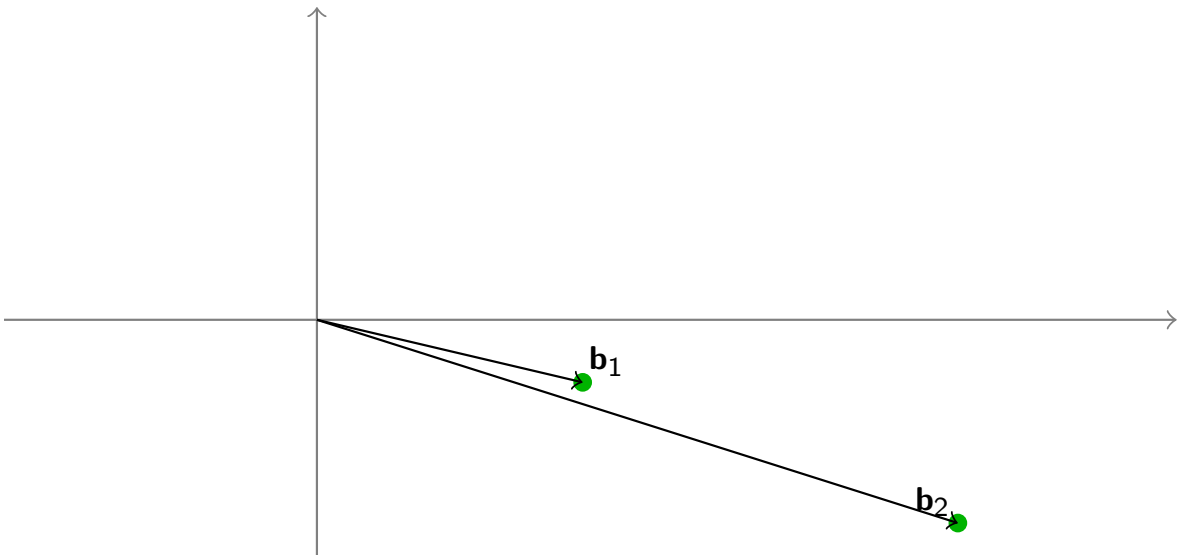
Definition (Lattice)

A discrete additive subgroup of $\mathbb{R}^n$

Shortest Vector Problem

Definition (Shortest Vector Problem, SVP)

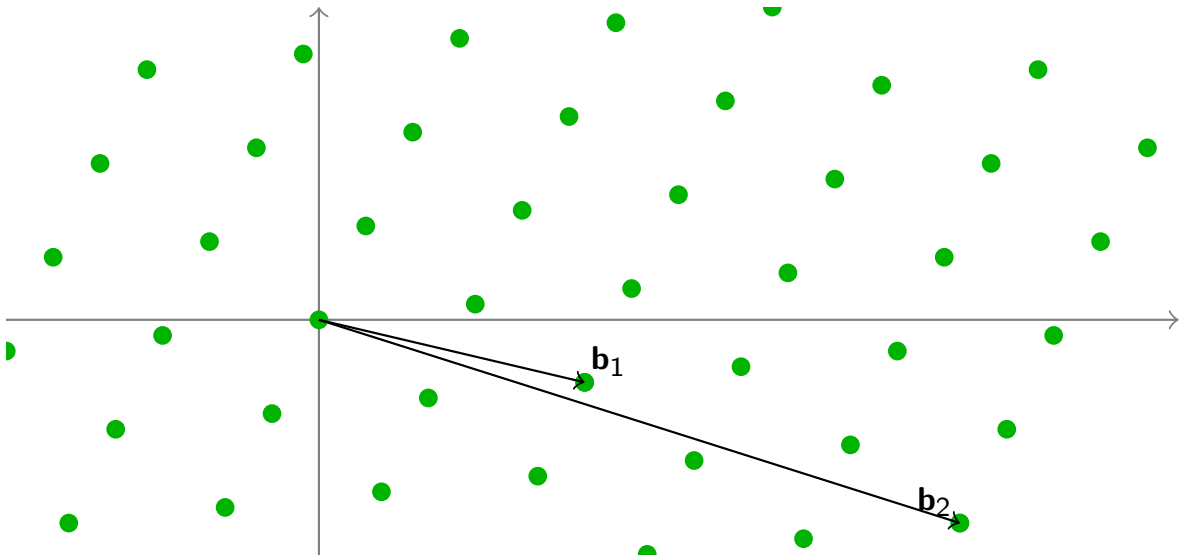
Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \lambda_1$



Shortest Vector Problem

Definition (Shortest Vector Problem, SVP)

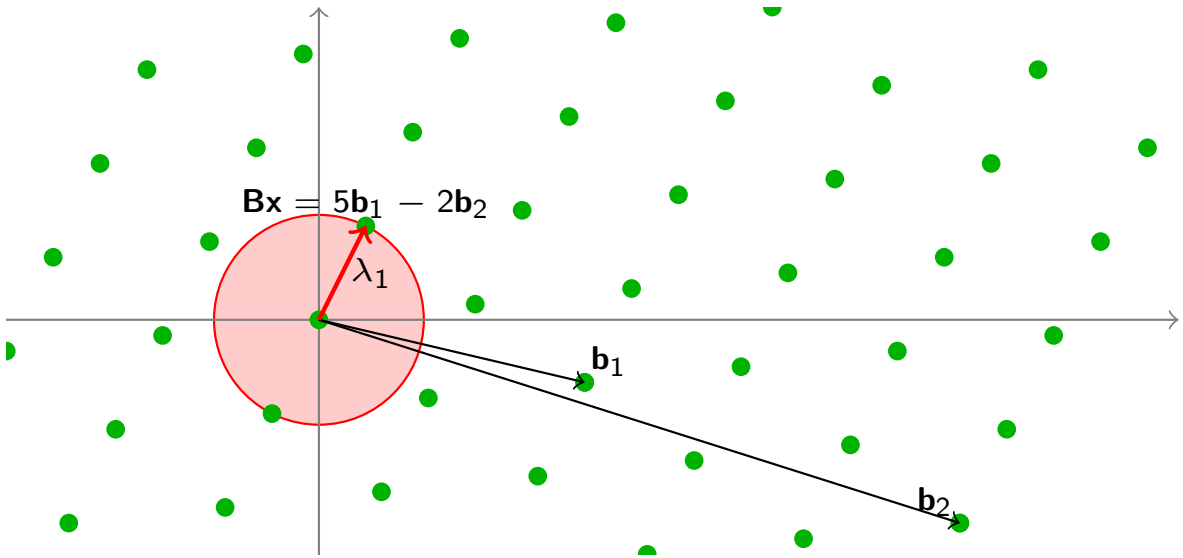
Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \lambda_1$



Shortest Vector Problem

Definition (Shortest Vector Problem, SVP)

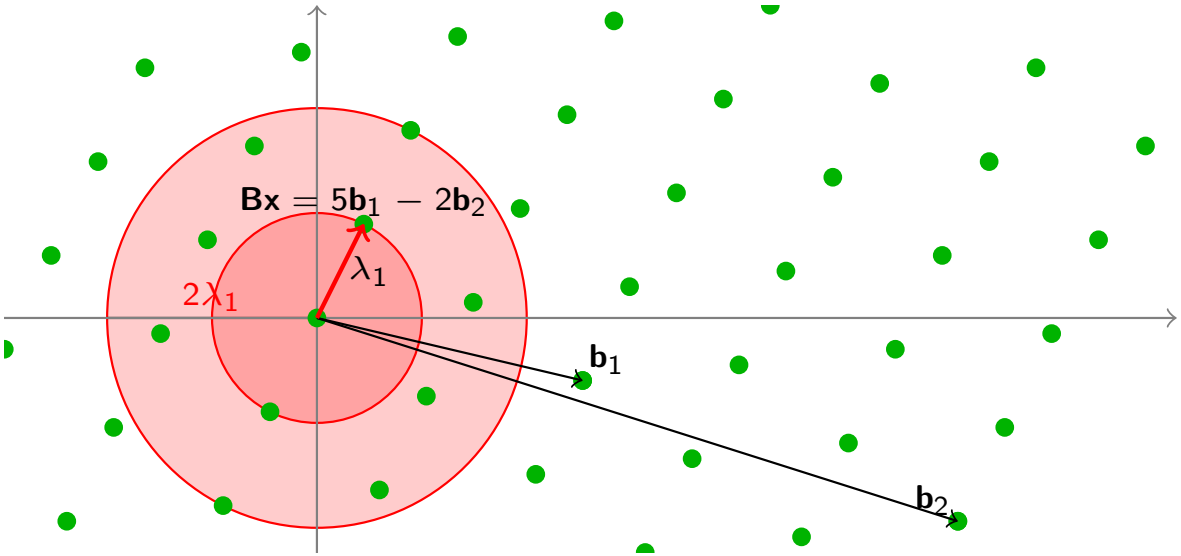
Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \lambda_1$



Shortest Vector Problem

Definition (Shortest Vector Problem, SVP_γ)

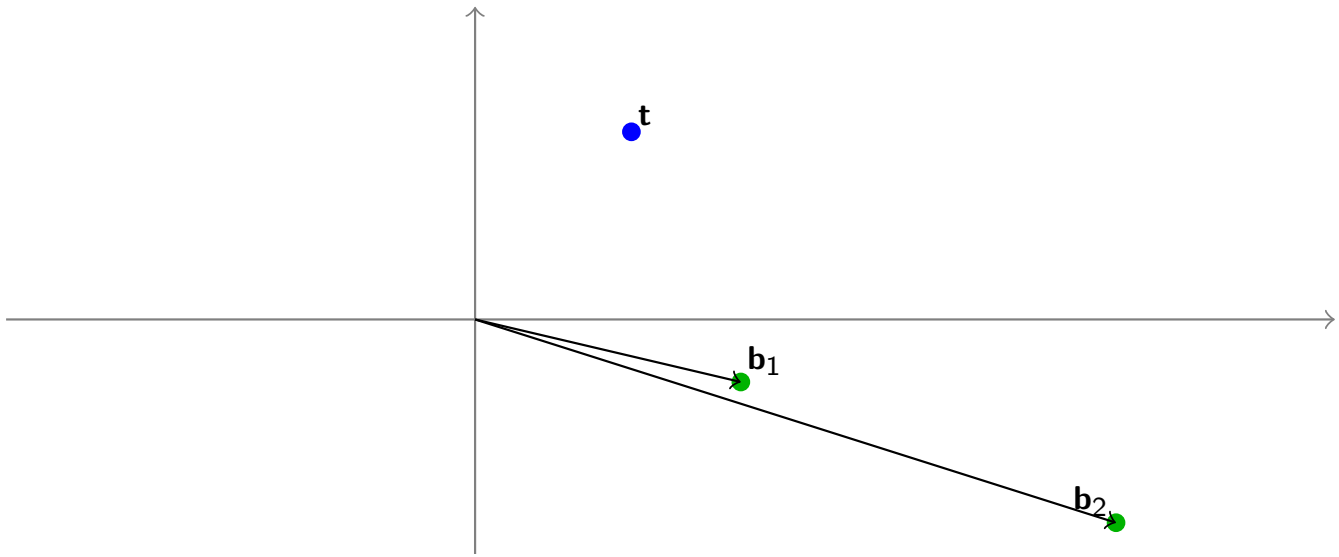
Given a lattice $\mathcal{L}(\mathbf{B})$, find a (nonzero) lattice vector $\mathbf{Bx}$ (with $\mathbf{x} \in \mathbb{Z}^k$) of length (at most) $\|\mathbf{Bx}\| \leq \gamma\lambda_1$



Closest Vector Problem

Definition (Closest Vector Problem, CVP)

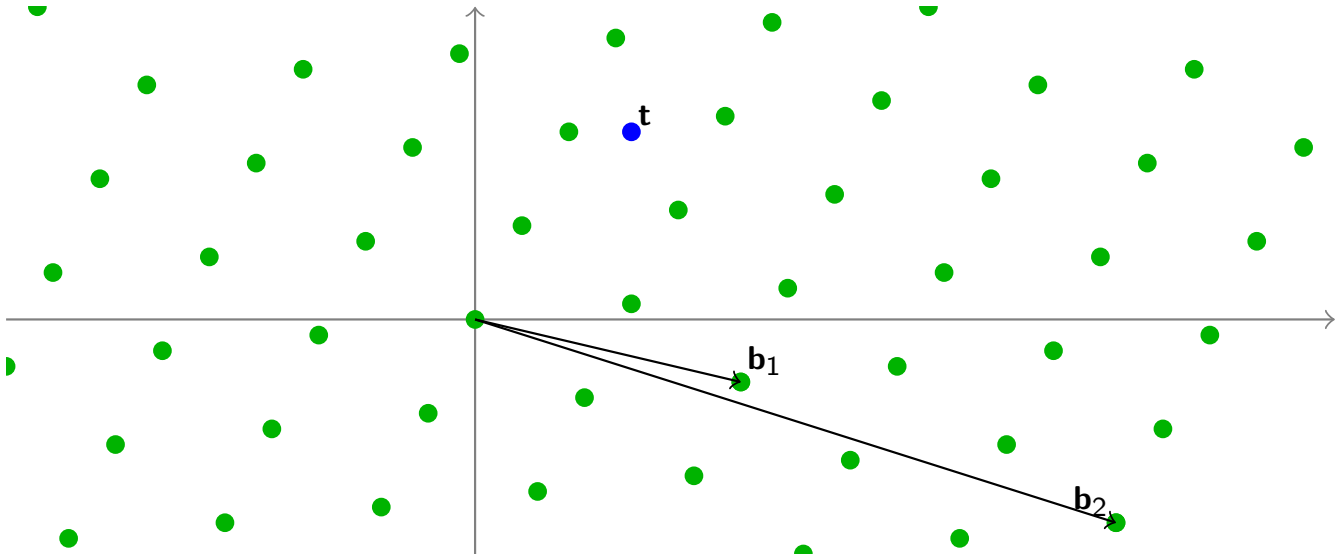
Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \mu$ from the target



Closest Vector Problem

Definition (Closest Vector Problem, CVP)

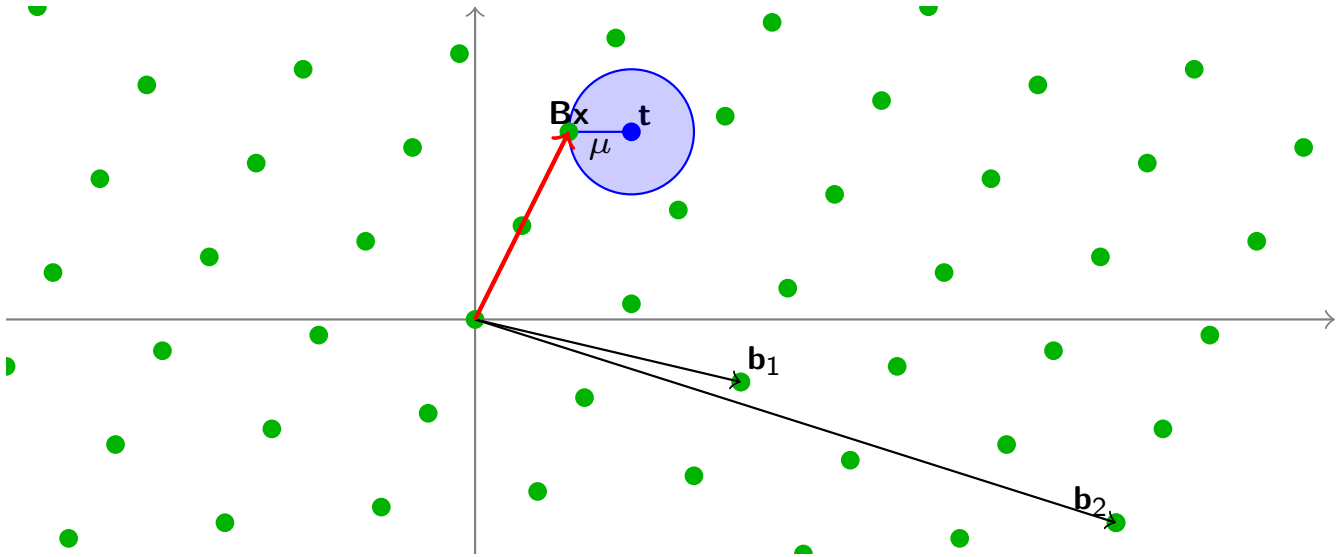
Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \mu$ from the target



Closest Vector Problem

Definition (Closest Vector Problem, CVP)

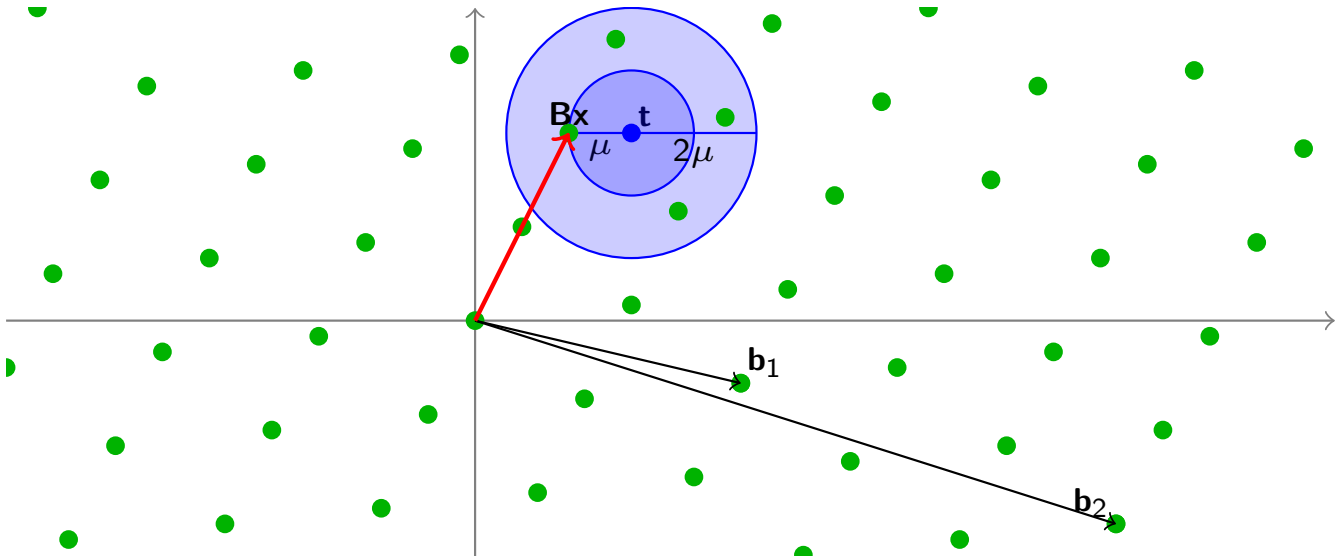
Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \mu$ from the target



Closest Vector Problem

Definition (Closest Vector Problem, CVP_γ)

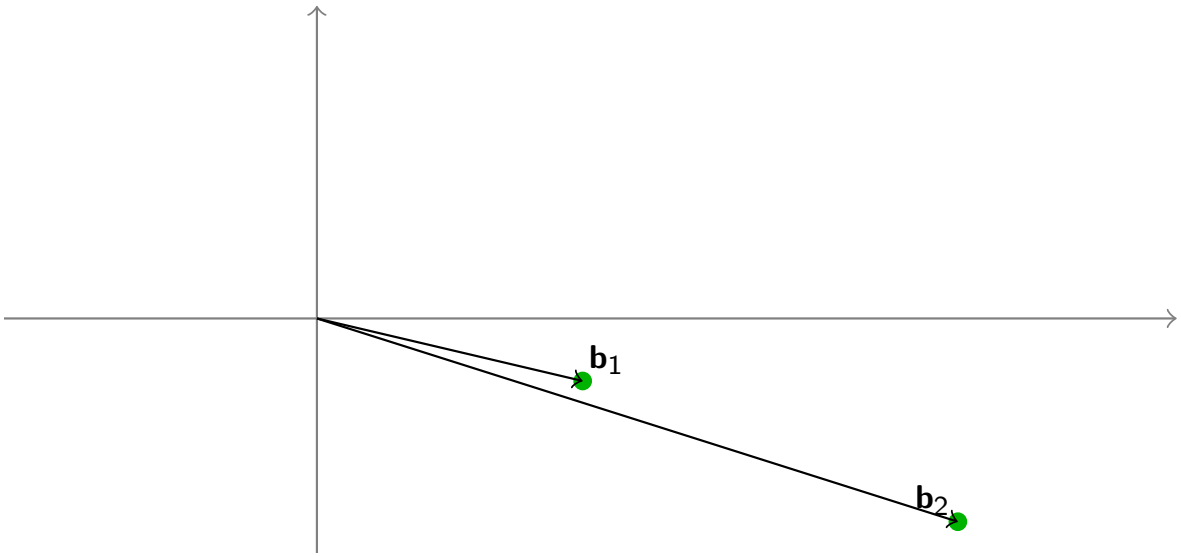
Given a lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t}$, find a lattice vector $\mathbf{Bx}$ within distance $\|\mathbf{Bx} - \mathbf{t}\| \leq \gamma\mu$ from the target



Shortest Independent Vectors Problem

Definition (Shortest Independent Vectors Problem, SIVP)

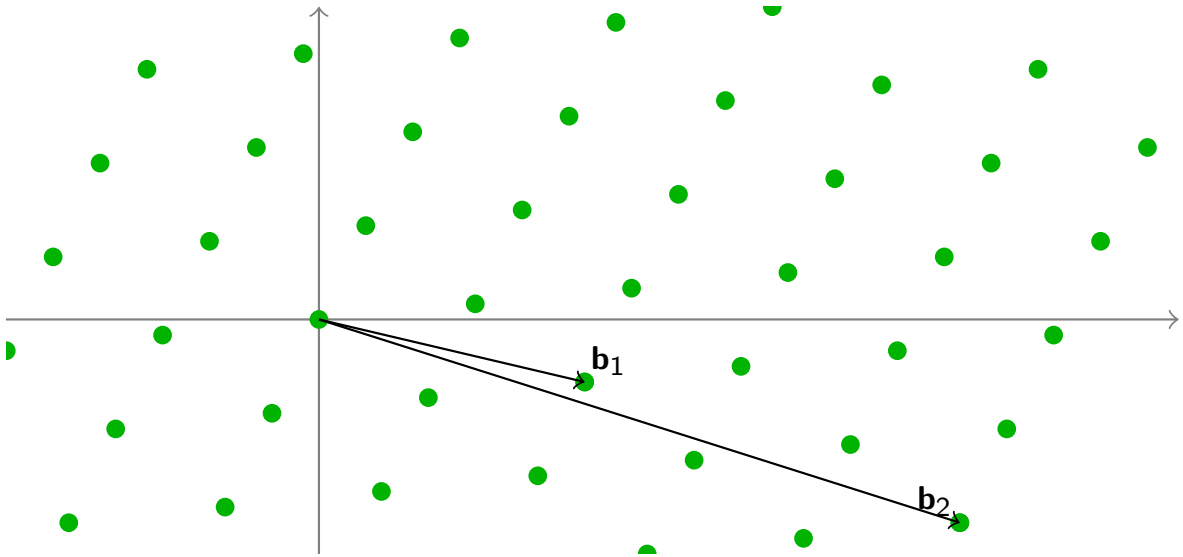
Given a lattice $\mathcal{L}(\mathbf{B})$, find n linearly independent lattice vectors $\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n$ of length (at most) $\max_i \|\mathbf{Bx}_i\| \leq \lambda_n$



Shortest Independent Vectors Problem

Definition (Shortest Independent Vectors Problem, SIVP)

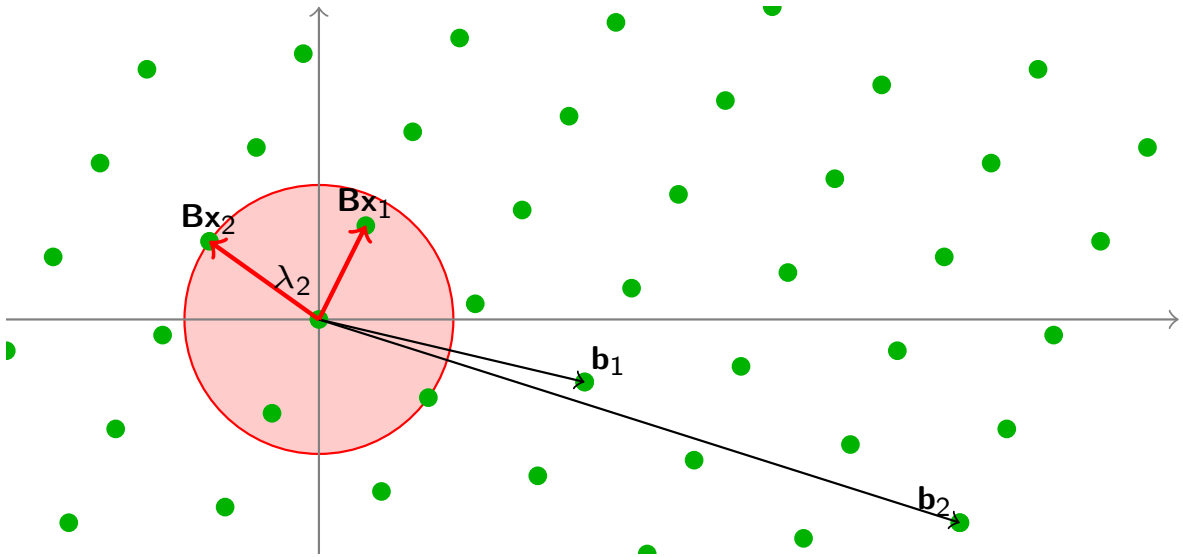
Given a lattice $\mathcal{L}(\mathbf{B})$, find n linearly independent lattice vectors $\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n$ of length (at most) $\max_i \|\mathbf{Bx}_i\| \leq \lambda_n$



Shortest Independent Vectors Problem

Definition (Shortest Independent Vectors Problem, SIVP)

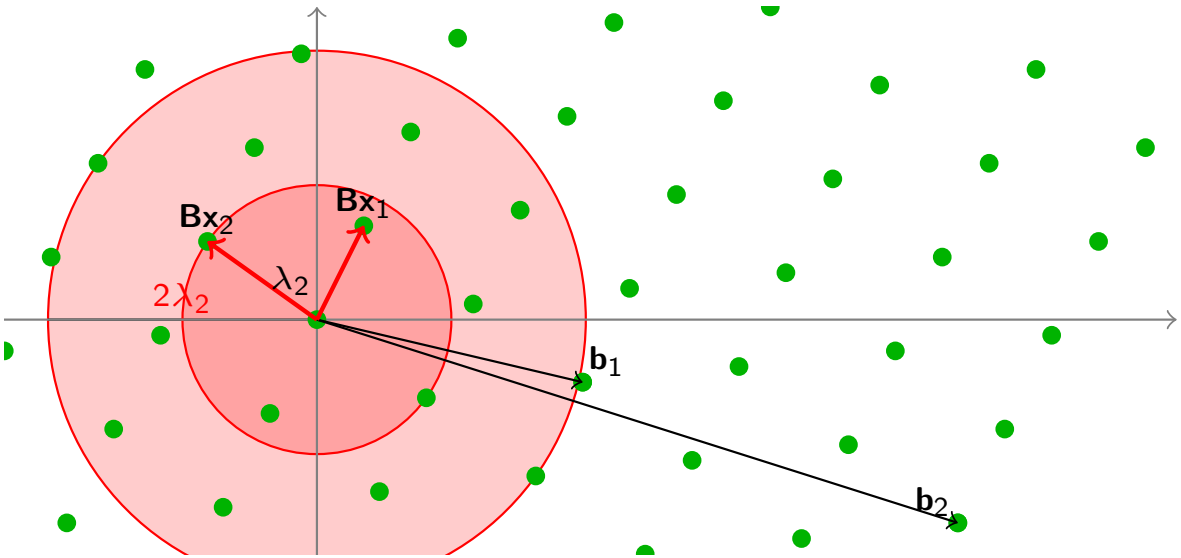
Given a lattice $\mathcal{L}(\mathbf{B})$, find n linearly independent lattice vectors $\mathbf{Bx}_1, \ldots, \mathbf{Bx}_n$ of length (at most) $\max_i \|\mathbf{Bx}_i\| \leq \lambda_n$



Shortest Independent Vectors Problem

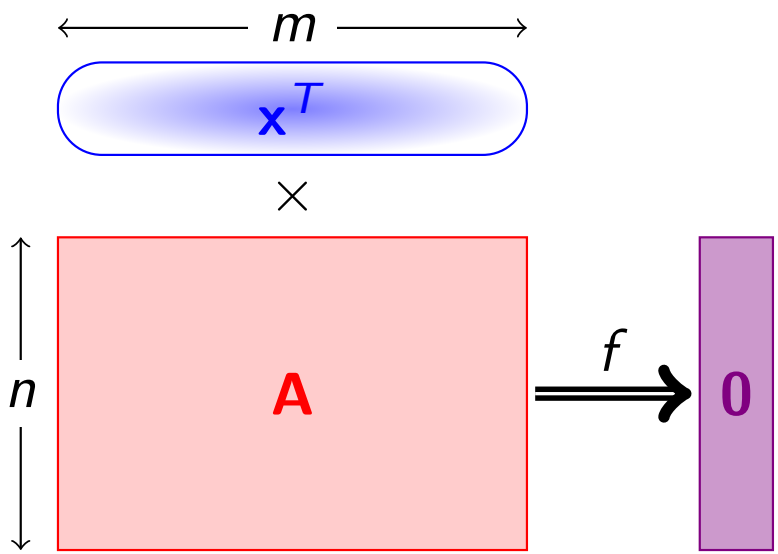
Definition (Shortest Independent Vectors Problem, SIVP_γ)

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



Short Integer Solution (SIS): random lattices

- Parameters: $m, n, q \in \mathbb{Z}$
- Key: $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
- Function: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$
- Choose $\mathbf{A}$ uniformly at random

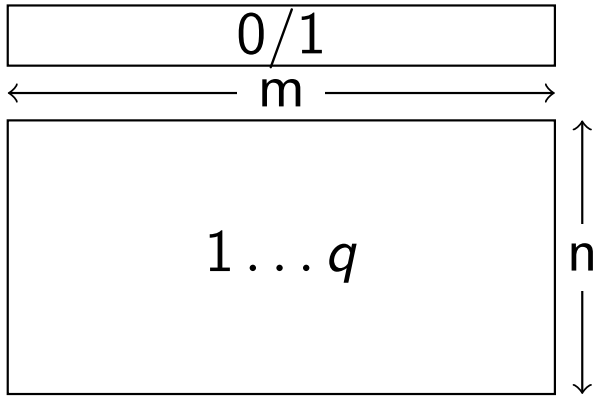


The SIS lattice: $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \pmod q\} \subseteq \mathbb{Z}^m$.

- Does $\Lambda_q^{\perp}(\mathbf{A})$ contain short nonzero vectors?
- Can you solve SVP in $\Lambda_q^{\perp}(\mathbf{A})$ when $\mathbf{A}$ is chosen at random?
  - If not, what makes it hard?
  - If not, how is it useful for cryptography?

SIS parameters and Collision Resistant Hashing

- $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
- Restrict $f_{\mathbf{A}}$ to $\mathbf{x} \in \{0,1\}^m$
- Parameters:
  - n: main security parameter
  - $q = n^2 = n^{O(1)}$ small modulus
  - $m = 2n \log_2 q = O(n \log n)$
  - e.g., $n = 256$, $q = 2^{16}$, $m = 8192$



- $f_{\mathbf{A}}$ is a compression function mapping $8192 \to 4096$ bits
  - There exist collisions $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{y} \pmod q$
  - Equivalently, $\mathbf{z} = \mathbf{x} - \mathbf{y} \in \{0, 1, -1\}^m$ satisfies $\mathbf{A}\mathbf{z} = \mathbf{0} \pmod q$
  - $\mathbf{z} \in \Lambda_q^{\perp}(\mathbf{A})$ is a short nonzero lattice vector of length $\|\mathbf{z}\|_\infty \leq 1$

Remark

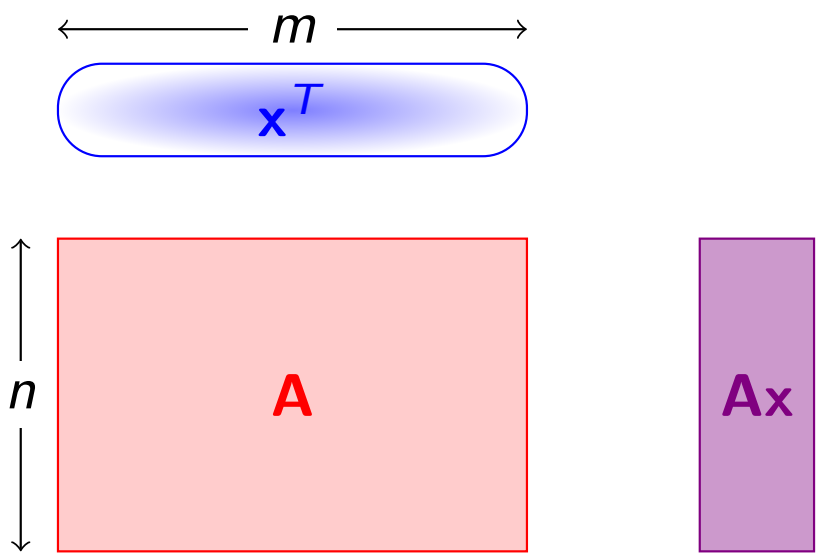
If SVP on $\Lambda_q^{\perp}(\mathbf{A})$ is hard, then $f_{\mathbf{A}}$ is a collision resistant hash function!

Example: Efficiency

- Function: $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
- Example parameters: $n = 2^8$, $q = 2^{16}$, $m = 2n \log_2 q = 2^{13}$
- $f_{\mathbf{A}}$ maps $\mathbf{x} \in \{0,1\}^m$ (8192 bits) to $\mathbb{Z}_q^n$ (4096 bits)
- Computing $f_{\mathbf{A}}$ takes $n \cdot m = 2^{21}$ additions and multiplications on small (2-byte) numbers
  - Easy to parallelize: using AVX512 SIMD instructions at 4 GHz, this is just $32\mu s$.
- Key Storage: $\mathbf{A}$ takes $n \cdot m = 2^{22}$ bytes, or $4MB$
  - Storage is not very good
  - More complex cryptographic applications may use higher $q = 2^{64}$, and $n = 2^{12} = 4096$
  - In the second part, we will see how to improve efficiency using more complex mathematics

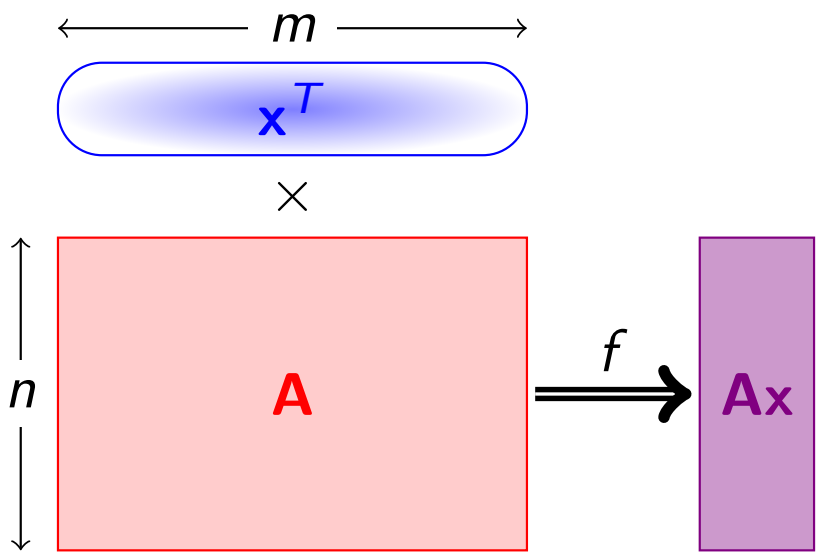
Ajtai's one-way function (SIS)

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



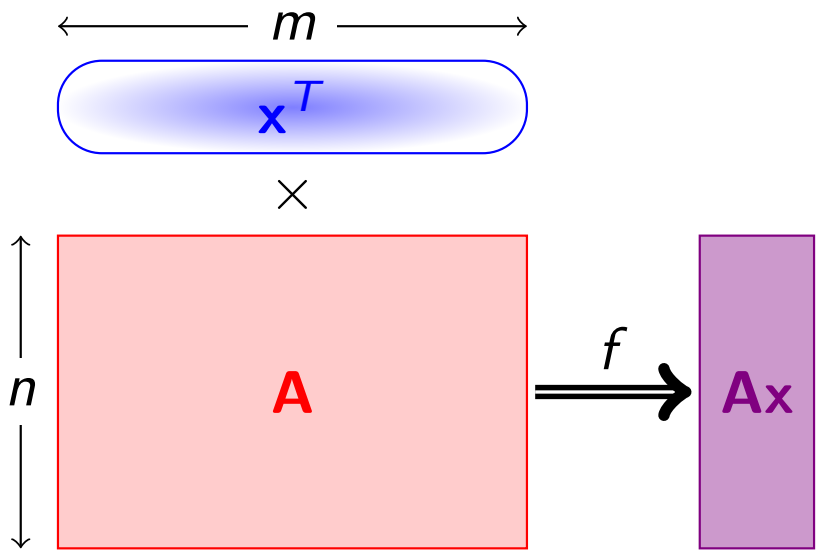
Ajtai's one-way function (SIS)

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



Ajtai's one-way function (SIS)

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



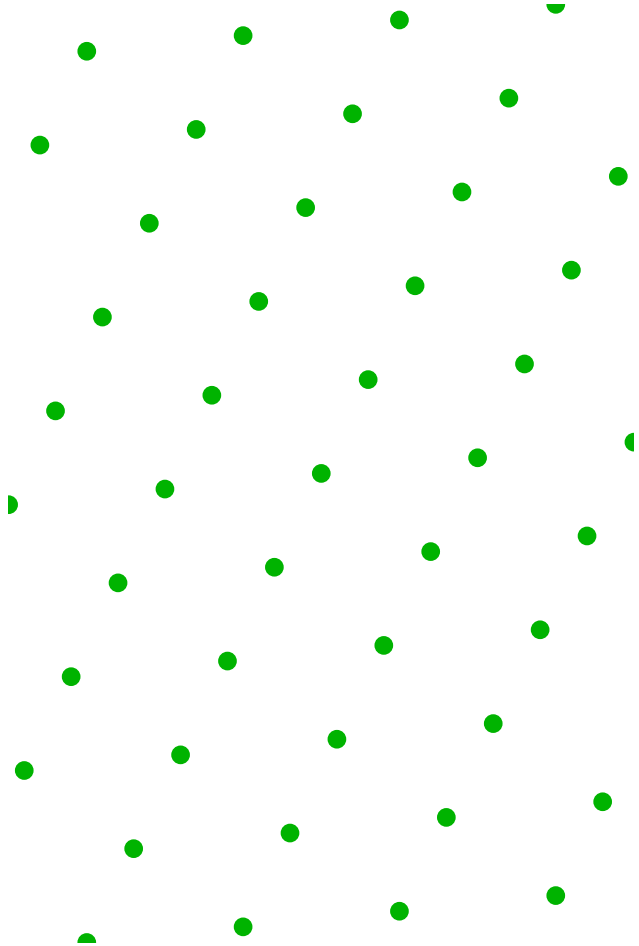
Theorem (A'96)

For $m > n \lg q$, if lattice problems (SIVP) are hard to approximate in the worst-case, then $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$ 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] ...

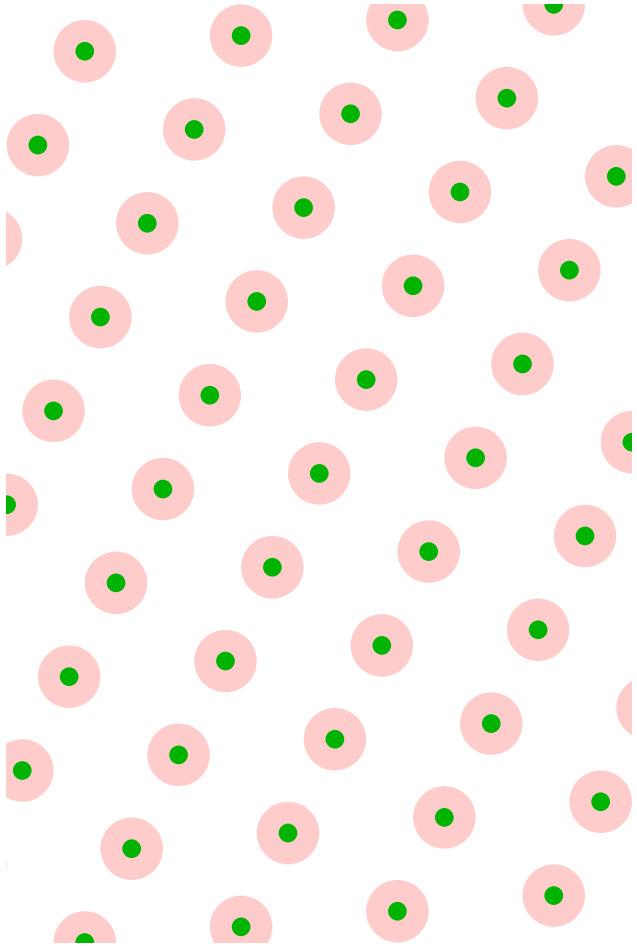
Blurring a lattice

Consider a lattice Λ, and



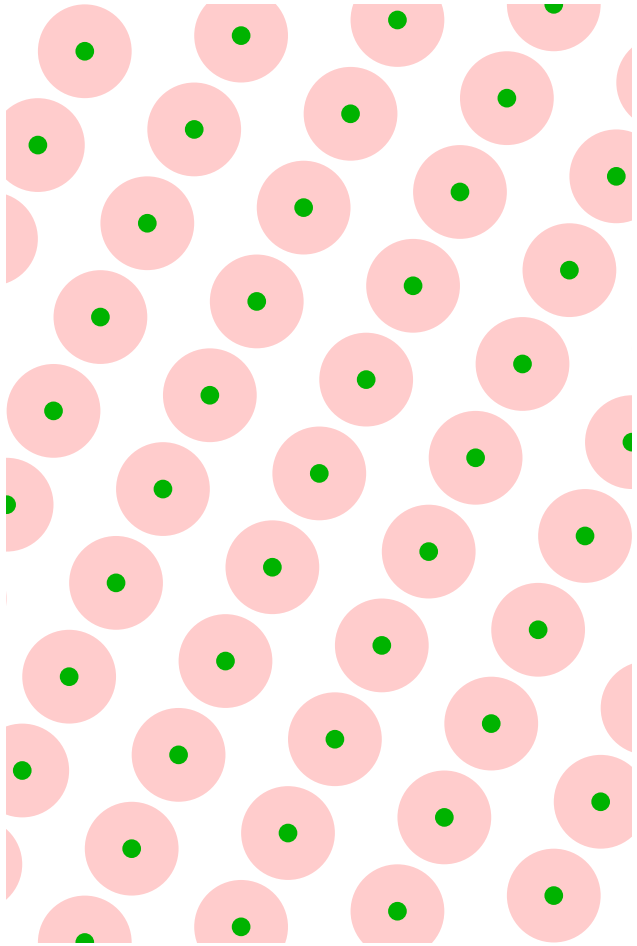
Blurring a lattice

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



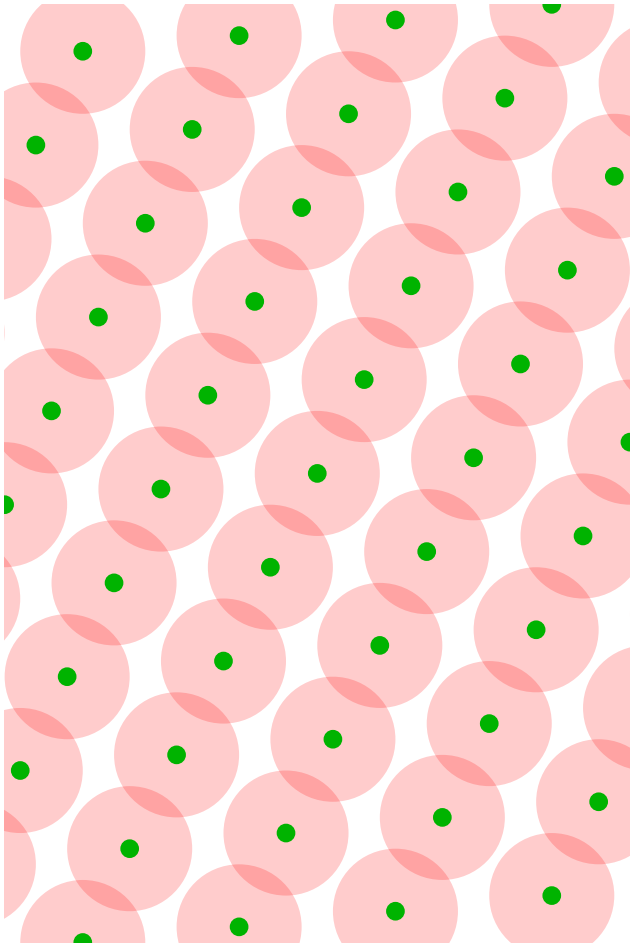
Blurring a lattice

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



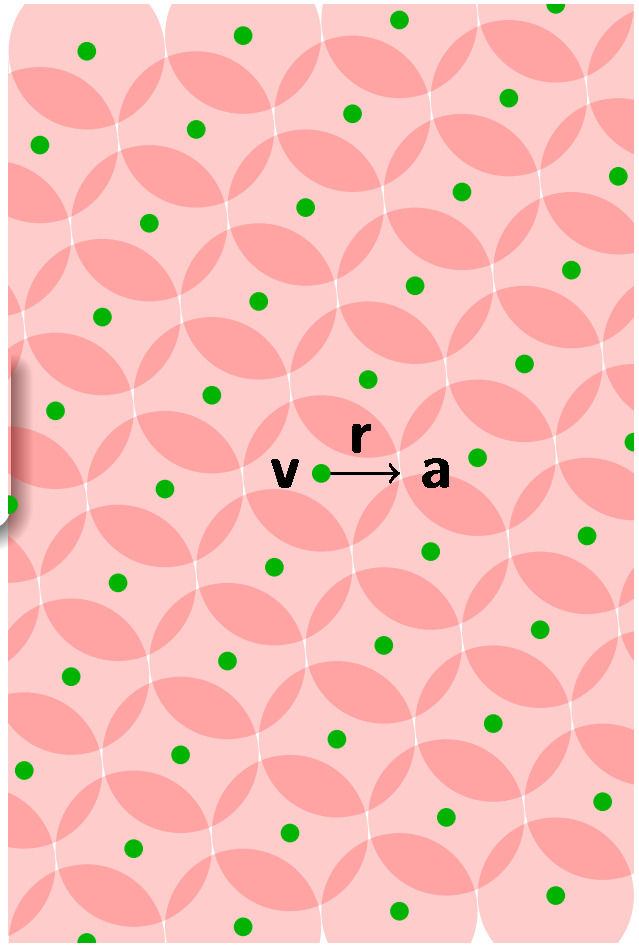
Blurring a lattice

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



Blurring a lattice

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



How much noise is needed?

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

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

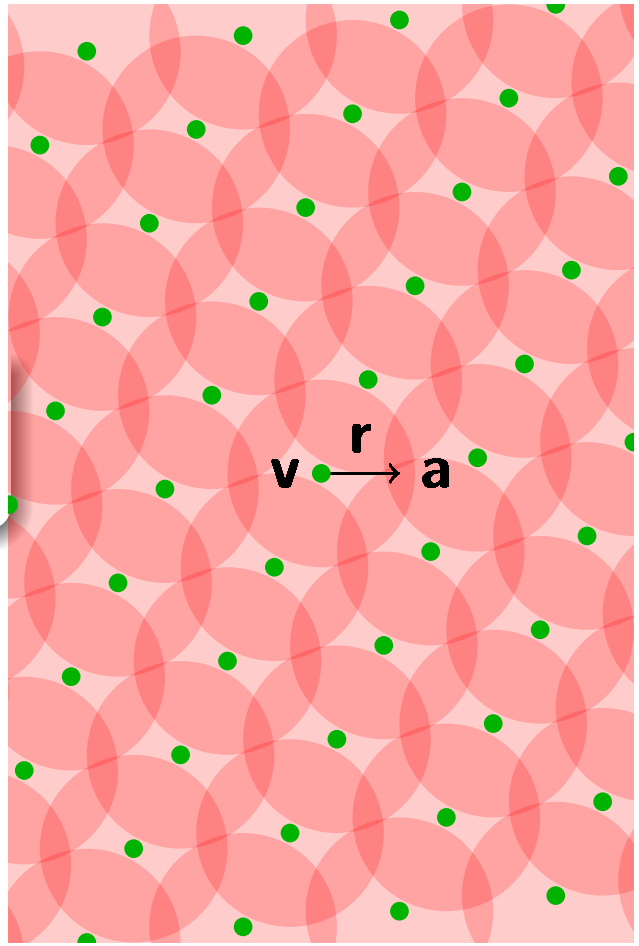
Blurring a lattice

Consider a lattice Λ, 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?

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

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



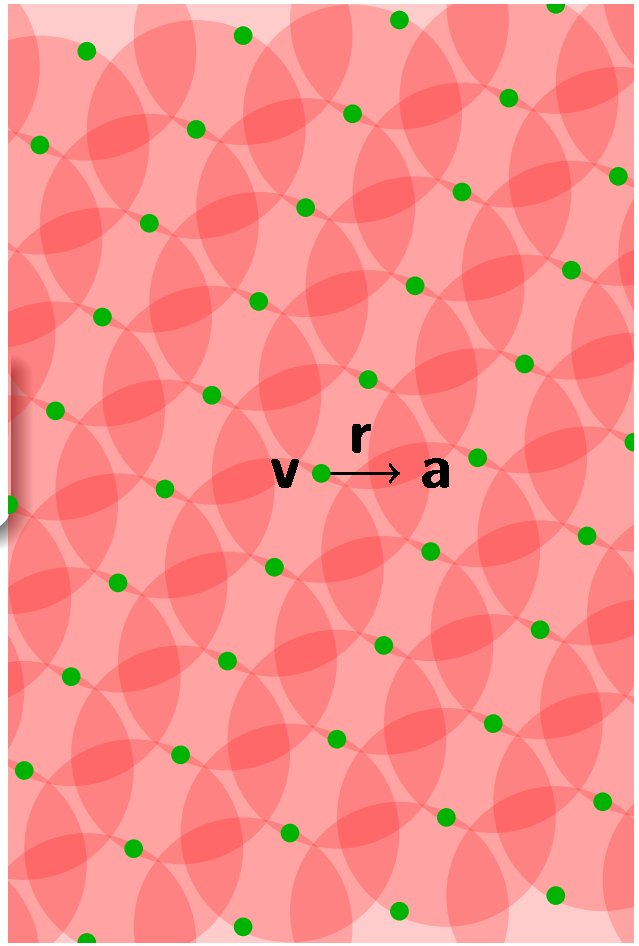
Blurring a lattice

Consider a lattice Λ, 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?

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

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



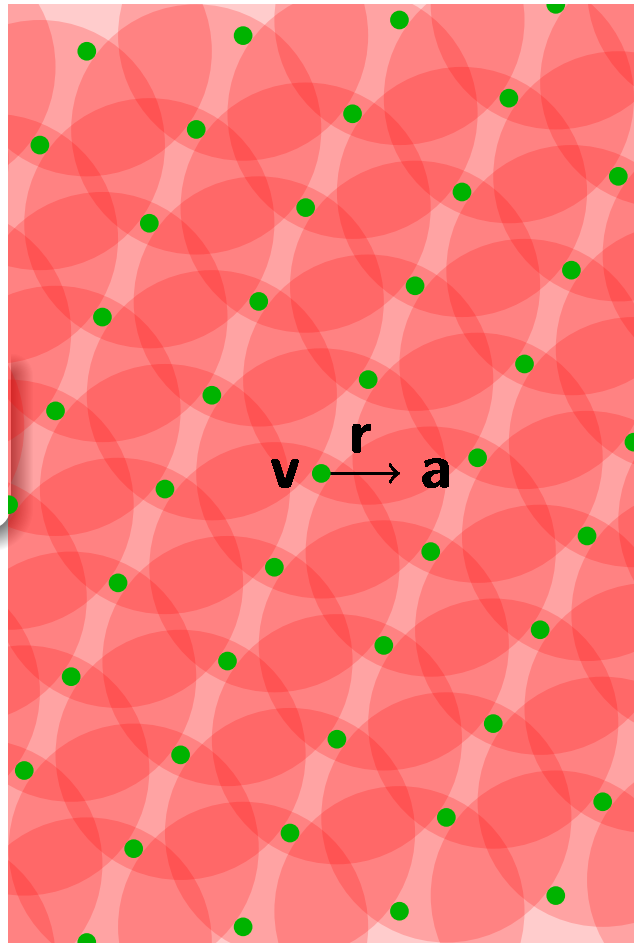
Blurring a lattice

Consider a lattice Λ, 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?

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

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



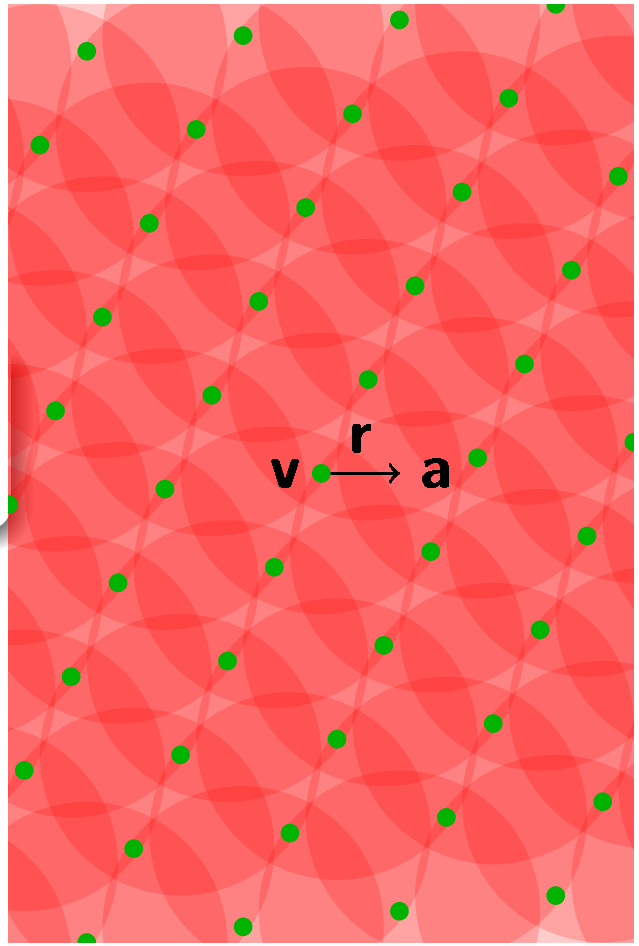
Blurring a lattice

Consider a lattice Λ, 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?

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

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



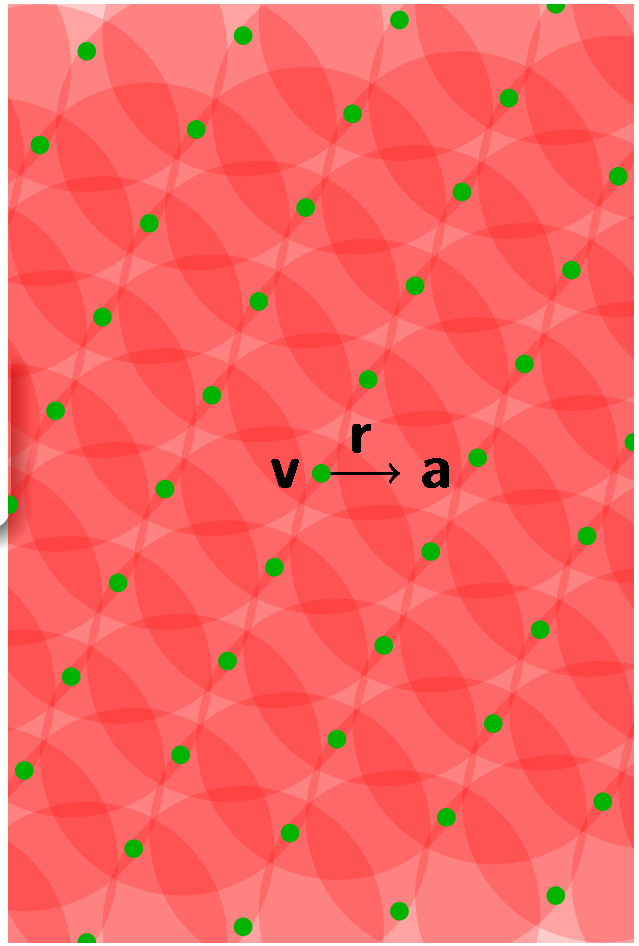
Blurring a lattice

Consider a lattice Λ, 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]

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

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



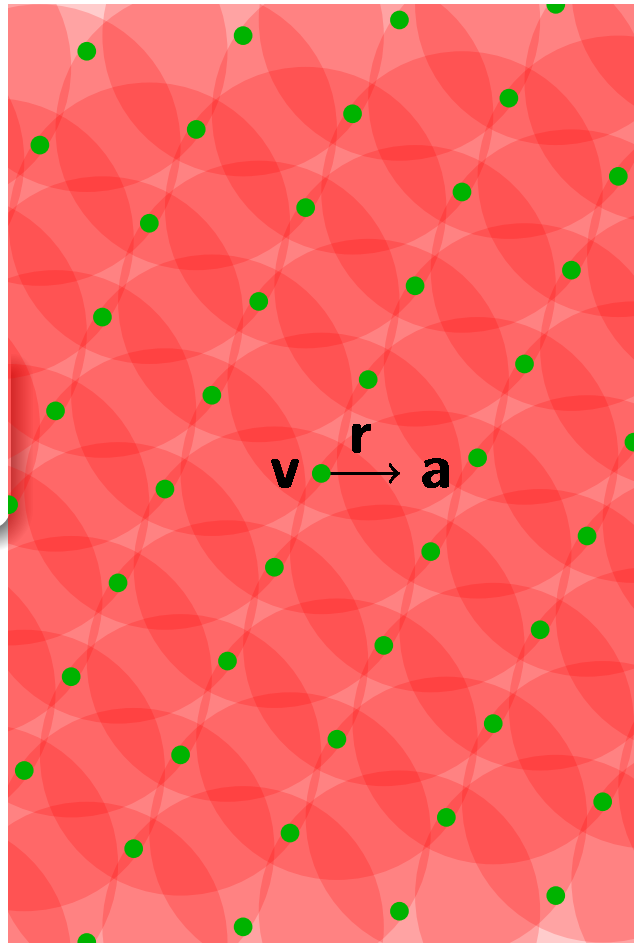
Blurring a lattice

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

How much noise is needed? [MR]

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

- Each point in $\mathbf{a} \in \mathbb{R}^n$ can be written $\mathbf{a} = \mathbf{v} + \mathbf{r}$ where $\mathbf{v} \in \mathcal{L}$ and $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.
- $\mathbf{a} \in \mathbb{R}^n/\Lambda$ is uniformly distributed.
- Think of $\mathbb{R}^n \approx \frac{1}{q}\Lambda$ [GPV'07]



Security of Ajtai's function (sketch)

- Generate random points $\mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i$, where
  - $\mathbf{v}_i$ is a random lattice point
  - $\mathbf{r}_i$ is a random error vector of length $\|\mathbf{r}_i\| \approx \sqrt{n}\lambda_n$

Security of Ajtai's function (sketch)

- Generate random points $\mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i$, where
  - $\mathbf{v}_i$ is a random lattice point
  - $\mathbf{r}_i$ is a random error vector of length $\|\mathbf{r}_i\| \approx \sqrt{n}\lambda_n$
- $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ is distributed almost uniformly at random in $\mathbb{R}^{n \times m}$, $q = n^{O(1)}$, $m = O(n \log q) = O(n \log n)$, so

Security of Ajtai's function (sketch)

- Generate random points $\mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i$, where
  - $\mathbf{v}_i$ is a random lattice point
  - $\mathbf{r}_i$ is a random error vector of length $\|\mathbf{r}_i\| \approx \sqrt{n}\lambda_n$
- $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ is distributed almost uniformly at random in $\mathbb{R}^{n \times m}$, $q = n^{O(1)}$, $m = O(n \log q) = O(n \log n)$, so
  - if we can break Ajtai's function $f_{\mathbf{A}}$, then
  - we can find a vector $\mathbf{z} \in \{-1, 0, 1\}^m$ such that

$$\sum \mathbf{a}_i z_i = \mathbf{0}$$

Security of Ajtai's function (sketch)

- Generate random points $\mathbf{a}_i = \mathbf{v}_i + \mathbf{r}_i$, where
  - $\mathbf{v}_i$ is a random lattice point
  - $\mathbf{r}_i$ is a random error vector of length $\|\mathbf{r}_i\| \approx \sqrt{n}\lambda_n$
- $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ is distributed almost uniformly at random in $\mathbb{R}^{n \times m}$, $q = n^{O(1)}$, $m = O(n \log q) = O(n \log n)$, so
  - if we can break Ajtai's function $f_{\mathbf{A}}$, then
  - we can find a vector $\mathbf{z} \in \{-1, 0, 1\}^m$ such that

$$\sum (\mathbf{v}_i + \mathbf{r}_i) z_i = \sum \mathbf{a}_i z_i = \mathbf{0}$$

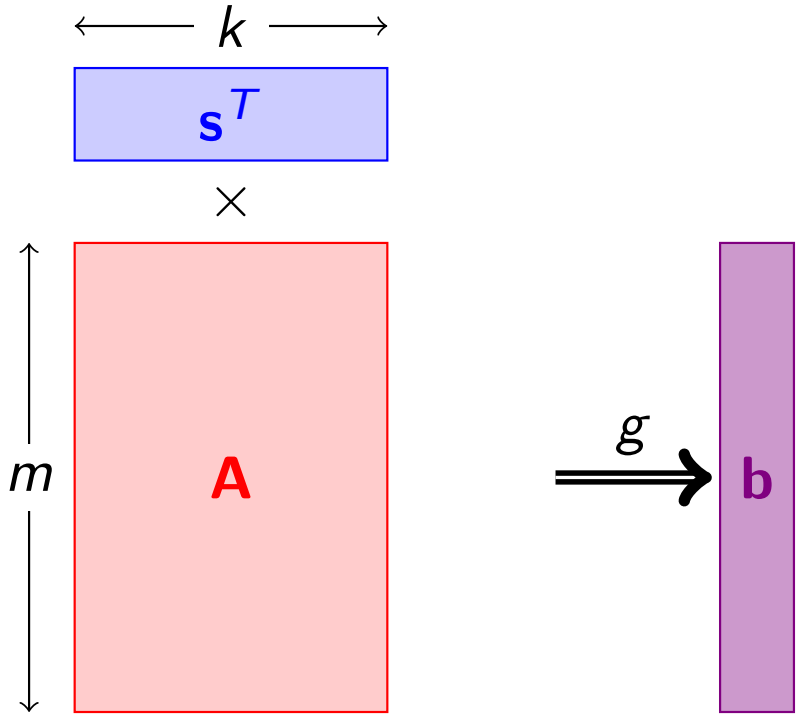
- Rearranging the terms yields a lattice vector

$$\sum \mathbf{v}_i z_i = -\sum \mathbf{r}_i z_i$$

of length at most $\| \sum \mathbf{r}_i z_i \| \approx \sqrt{m} \cdot \max \|\mathbf{r}_i\| \approx n \cdot \lambda_n$

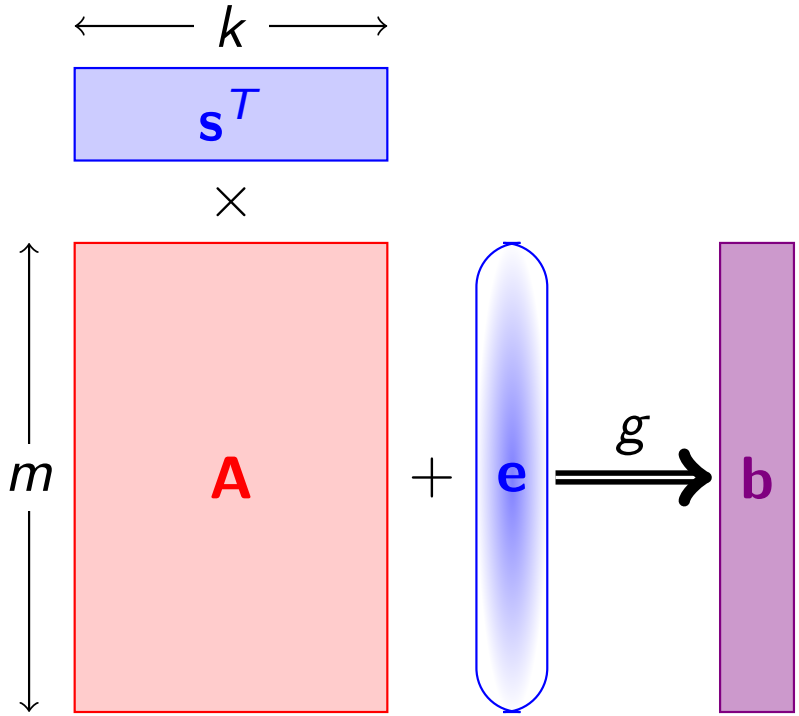
Regev's Learning With Errors (LWE)

- $\mathbf{A} \in \mathbb{Z}_q^{m \times k}$, $\mathbf{s} \in \mathbb{Z}_q^k$, $\mathbf{e} \in \mathcal{E}^m$.
- $g_{\mathbf{A}}(\mathbf{s} \quad) = \mathbf{As} \qquad \bmod q$



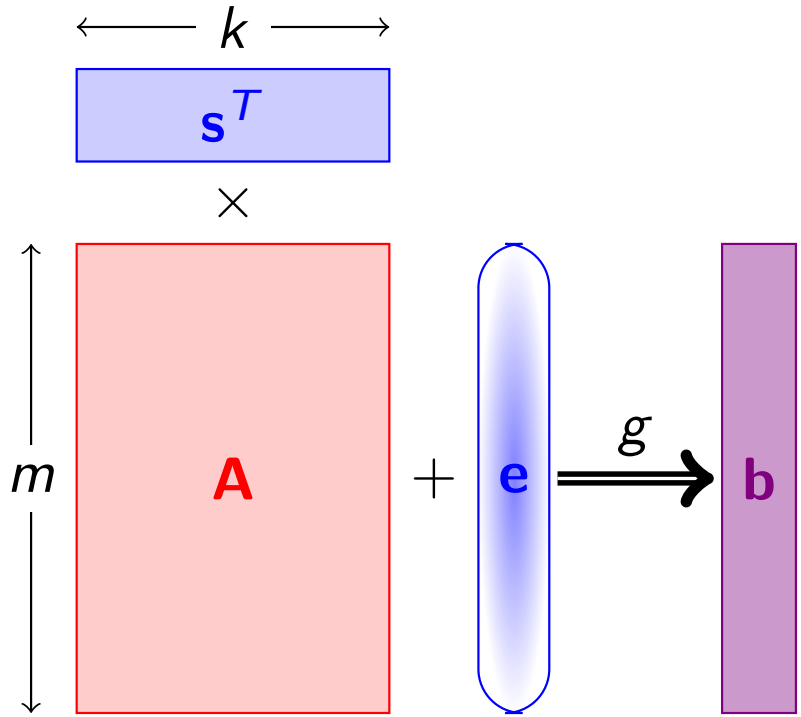
Regev's Learning With Errors (LWE)

- $\mathbf{A} \in \mathbb{Z}_q^{m \times k}$, $\mathbf{s} \in \mathbb{Z}_q^k$, $\mathbf{e} \in \mathcal{E}^m$.
- $g_{\mathbf{A}}(\mathbf{s}; \mathbf{e}) = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$
- Learning with Errors: Given $\mathbf{A}$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$, recover $\mathbf{s}$.



Regev's Learning With Errors (LWE)

- $\mathbf{A} \in \mathbb{Z}_q^{m \times k}$, $\mathbf{s} \in \mathbb{Z}_q^k$, $\mathbf{e} \in \mathcal{E}^m$.
- $g_{\mathbf{A}}(\mathbf{s}; \mathbf{e}) = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$
- Learning with Errors: Given $\mathbf{A}$ and $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$, recover $\mathbf{s}$.



Theorem (R'05)

The function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ is hard to invert on the average, assuming SIVP is hard to approximate in the worst-case.

Applications: CPA PKE [R'05], CCA PKE [PW'08], (H)IBE [GPV'08,CHKP'10,ABB'10], FHE [...,B'12,AP'13,GSW'13], ...

Hermite Normal Form (HNF)

- Assuming $\mathbf{A}$ is nondegenerate (i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^n$), one can find nonsingular $\mathbf{U} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{UA} = [\mathbf{I} \mid \mathbf{H}]$

Hermite Normal Form (HNF)

- Assuming $\mathbf{A}$ is nondegenerate (i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^n$), one can find nonsingular $\mathbf{U} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{UA} = [\mathbf{I} \mid \mathbf{H}]$
- $[\mathbf{I} \mid \mathbf{H}]$ is called the Hermite Normal Form (HNF) of $\mathbf{A}$, and can be efficiently computed from $\mathbf{A}$

Hermite Normal Form (HNF)

- Assuming $\mathbf{A}$ is nondegenerate (i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^n$), one can find nonsingular $\mathbf{U} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{UA} = [\mathbf{I} \mid \mathbf{H}]$
- $[\mathbf{I} \mid \mathbf{H}]$ is called the Hermite Normal Form (HNF) of $\mathbf{A}$, and can be efficiently computed from $\mathbf{A}$
- Putting $\mathbf{A}$ in HNF does not decrease the security of cryptographic functions [M'01]:
  - Breaking $f_{\mathbf{A}}$ is equivalent to breaking $f_{\mathbf{UA}}(\mathbf{x}) = \mathbf{U} \cdot f_{\mathbf{A}}(\mathbf{x})$ because $\mathbf{y} \mapsto \mathbf{Uy}$ is injective.

Hermite Normal Form (HNF)

- Assuming $\mathbf{A}$ is nondegenerate (i.e., $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^n$), one can find nonsingular $\mathbf{U} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{UA} = [\mathbf{I} \mid \mathbf{H}]$
- $[\mathbf{I} \mid \mathbf{H}]$ is called the Hermite Normal Form (HNF) of $\mathbf{A}$, and can be efficiently computed from $\mathbf{A}$
- Putting $\mathbf{A}$ in HNF does not decrease the security of cryptographic functions [M'01]:
  - Breaking $f_{\mathbf{A}}$ is equivalent to breaking $f_{\mathbf{UA}}(\mathbf{x}) = \mathbf{U} \cdot f_{\mathbf{A}}(\mathbf{x})$ because $\mathbf{y} \mapsto \mathbf{Uy}$ is injective.
  - Similarly, inverting $g_{\mathbf{A}}$ is equivalent to inverting $g_{\mathbf{AU}}$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$
- Parameters $k + n = m$: $\quad n = m - k, \quad m - n = k$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$
- Parameters $k + n = m$: $n = m - k$, $m - n = k$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$
- Parameters $k + n = m$: $n = m - k$, $m - n = k$
- Input: set $\mathbf{e} = \mathbf{x}$ and pick random $\mathbf{s} \in \mathbb{Z}_q^k$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$
- Parameters $k + n = m$: $\quad n = m - k, \quad m - n = k$
- Input: set $\mathbf{e} = \mathbf{x}$ and pick random $\mathbf{s} \in \mathbb{Z}_q^k$
- Can compute $g'_{\mathbf{H}}(\mathbb{Z}_q, \mathbf{x}) \mapsto f'_{\mathbf{H}}(\mathbf{x})$

$$\begin{aligned} f'_{\mathbf{H}}(g'_{\mathbf{H}}(\mathbf{s}, \mathbf{x})) &= [\mathbf{I} \mid -\mathbf{H}] \left(\begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix} \mathbf{s} + \mathbf{x} \right) \\ &= (\mathbf{I}\mathbf{H} - \mathbf{H}\mathbf{I})\mathbf{s} + [\mathbf{I} \mid -\mathbf{H}]\mathbf{x} = f'_{\mathbf{H}}(\mathbf{x}) \end{aligned}$$

Equivalence of SIS and LWE

- Equivalent HNF variant of SIS/LWE [M'01]:
  - $f'_{\mathbf{H}}(\mathbf{x}) = f_{\mathbf{A}}(\mathbf{x})$ for $\mathbf{A} = [\mathbf{I} \mid -\mathbf{H}]$, where $\mathbf{H} \in \mathbb{Z}_q^{n \times (m-n)}$
  - $g'_{\mathbf{H}}(\mathbf{s}, \mathbf{e}) = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$ for $\mathbf{A} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix}$, where $\mathbf{H} \in \mathbb{Z}_q^{(m-k) \times k}$
- Parameters $k + n = m$: $\quad n = m - k, \quad m - n = k$
- Input: set $\mathbf{e} = \mathbf{x}$ and pick random $\mathbf{s} \in \mathbb{Z}_q^k$
- Can compute $g'_{\mathbf{H}}(\mathbb{Z}_q, \mathbf{x}) \mapsto f'_{\mathbf{H}}(\mathbf{x})$

$$\begin{aligned} f'_{\mathbf{H}}(g'_{\mathbf{H}}(\mathbf{s}, \mathbf{x})) &= [\mathbf{I} \mid -\mathbf{H}] \left(\begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix} \mathbf{s} + \mathbf{x} \right) \\ &= (\mathbf{I}\mathbf{H} - \mathbf{H}\mathbf{I})\mathbf{s} + [\mathbf{I} \mid -\mathbf{H}]\mathbf{x} = f'_{\mathbf{H}}(\mathbf{x}) \end{aligned}$$

- Similarly, $f'_{\mathbf{H}}(\mathbf{x}) \mapsto g'_{\mathbf{H}}(\mathbb{Z}_q^k, \mathbf{x}) = g'_{\mathbf{H}}(\mathbf{s}, \mathbf{x}) + g'_{\mathbf{H}}(\mathbb{Z}_q^k, \mathbf{0})$, for $\mathbf{s} = -\mathbf{x}_2$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \qquad \mathbf{s} = -\mathbf{x}_2 \qquad g'_{\mathbf{H}}(\mathbf{s}, \mathbf{x}) = \begin{bmatrix} -\mathbf{H}\mathbf{x}_2 + \mathbf{x}_1 \\ -\mathbf{x}_2 + \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} f'_{\mathbf{H}}(\mathbf{x}) \\ \mathbf{0} \end{bmatrix}$$

Hardness of SIS/LWE

- $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially the same function, parametrized by dimensions $k + n = m$, modulus q and bound $\|\mathbf{x}\| \leq \beta$

Hardness of SIS/LWE

- $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially the same function, parametrized by dimensions $k + n = m$, modulus q and bound $\|\mathbf{x}\| \leq \beta$
- SIS [A'96] usually formulated using $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$
  - Larger β, resulting in a compression (hash) function
  - Collision resistant based on hardness of n-dim. lattices

Hardness of SIS/LWE

- $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially the same function, parametrized by dimensions $k + n = m$, modulus q and bound $\|\mathbf{x}\| \leq \beta$
- SIS [A'96] usually formulated using $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$
  - Larger β, resulting in a compression (hash) function
  - Collision resistant based on hardness of n-dim. lattices
- LWE [R'05] usually formulated using $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x}$
  - Smaller β, resulting in an injective function. (No collisions!)
  - Hard to invert based on hardness of k-dim. lattices.

Hardness of SIS/LWE

- $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially the same function, parametrized by dimensions $k + n = m$, modulus q and bound $\|\mathbf{x}\| \leq \beta$
- SIS [A'96] usually formulated using $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$
  - Larger β, resulting in a compression (hash) function
  - Collision resistant based on hardness of n-dim. lattices
- LWE [R'05] usually formulated using $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x}$
  - Smaller β, resulting in an injective function. (No collisions!)
  - Hard to invert based on hardness of k-dim. lattices.
- Proof techniques and applications are quite different depending on whether β is small or large.

Hardness of SIS/LWE

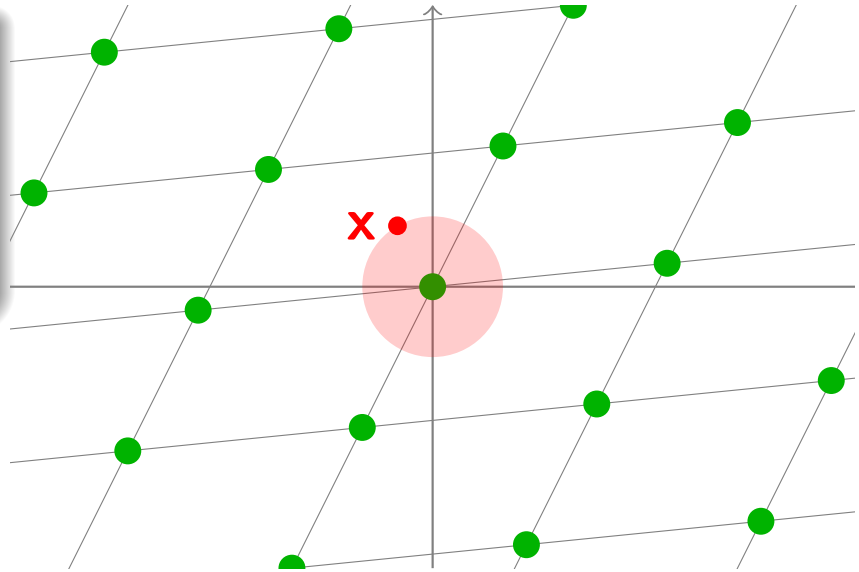
- $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are essentially the same function, parametrized by dimensions $k + n = m$, modulus q and bound $\|\mathbf{x}\| \leq \beta$
- SIS [A'96] usually formulated using $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$
  - Larger β, resulting in a compression (hash) function
  - Collision resistant based on hardness of n-dim. lattices
- LWE [R'05] usually formulated using $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x}$
  - Smaller β, resulting in an injective function. (No collisions!)
  - Hard to invert based on hardness of k-dim. lattices.
- Proof techniques and applications are quite different depending on whether β is small or large.
- One can use $f_{\mathbf{A}}$ both for SIS and LWE

SIS/LWE as CVP

Candidate OWF

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

Input: $\mathbf{x}$, $\|\mathbf{x}\| \leq \beta$



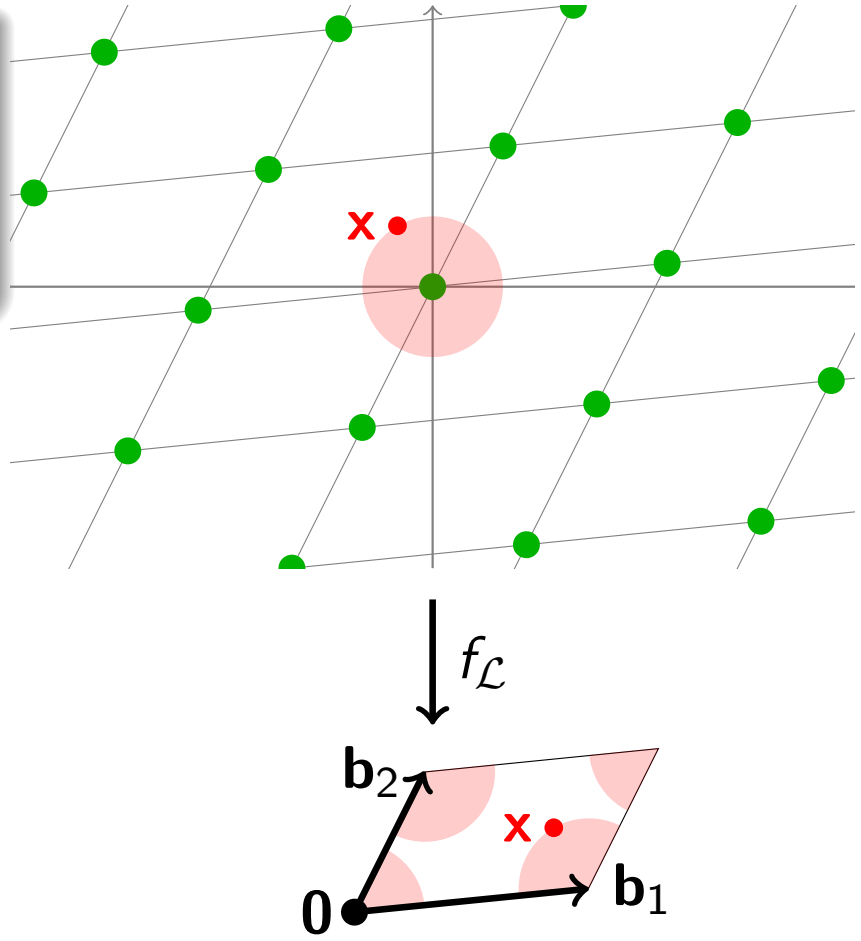
SIS/LWE as CVP

Candidate OWF

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

Input: $\mathbf{x}$, $\|\mathbf{x}\| \leq \beta$

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

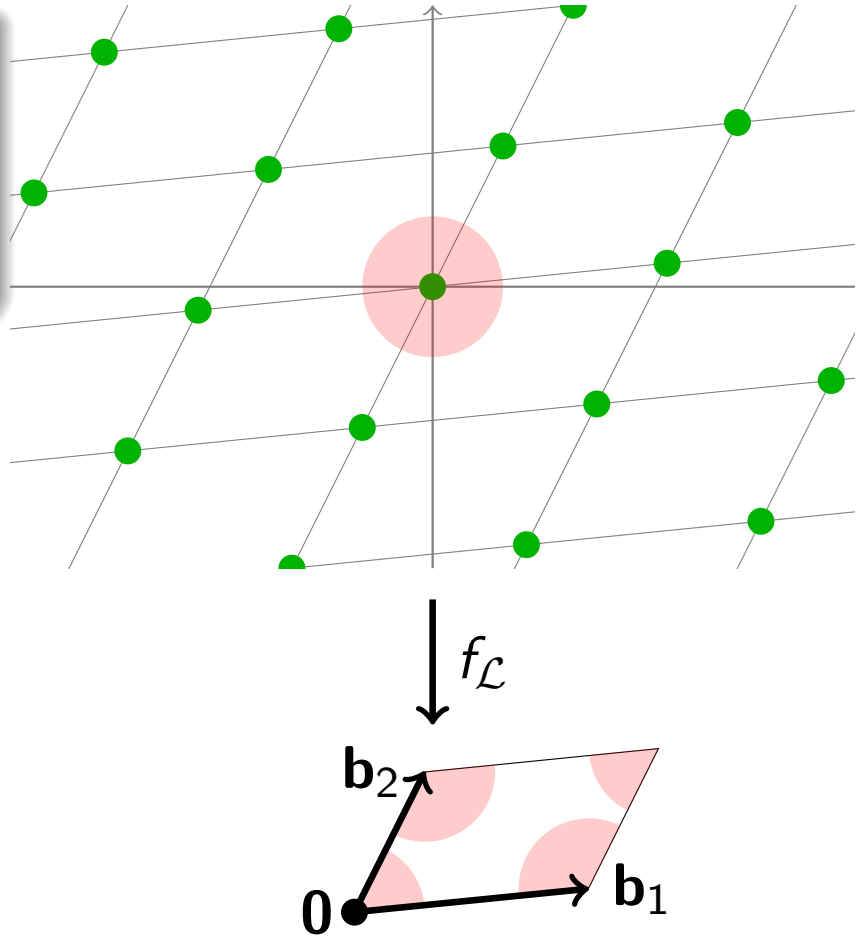


SIS/LWE as CVP

Candidate OWF

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

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

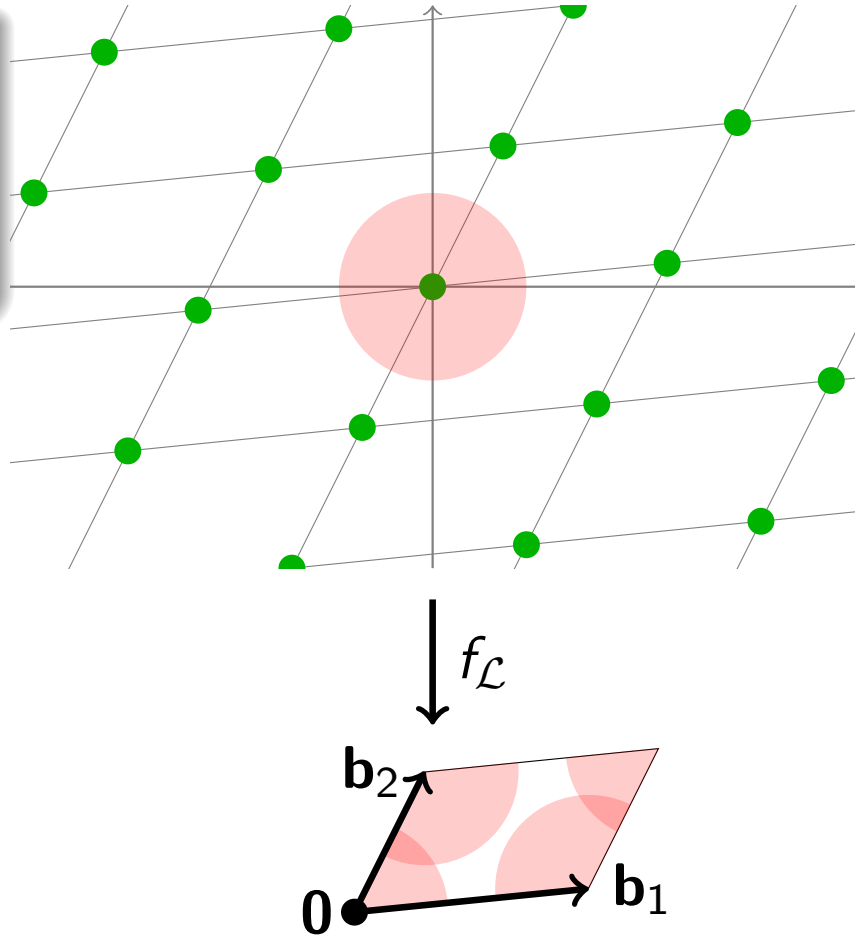


SIS/LWE as CVP

Candidate OWF

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

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

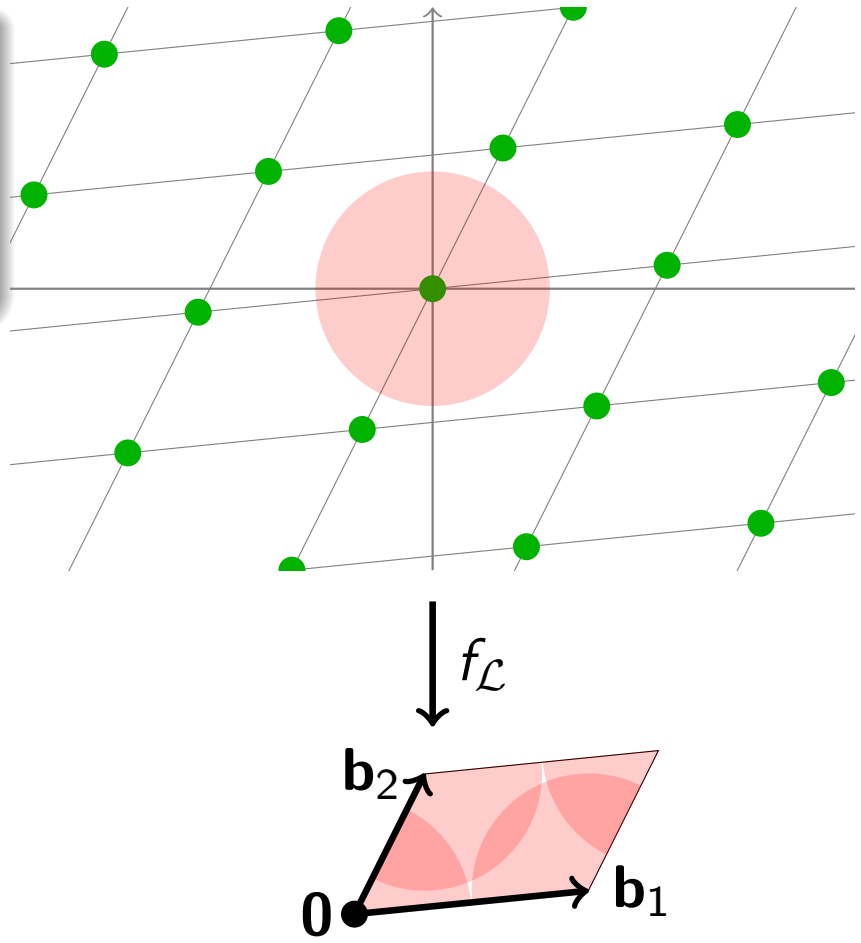


SIS/LWE as CVP

Candidate OWF

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

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

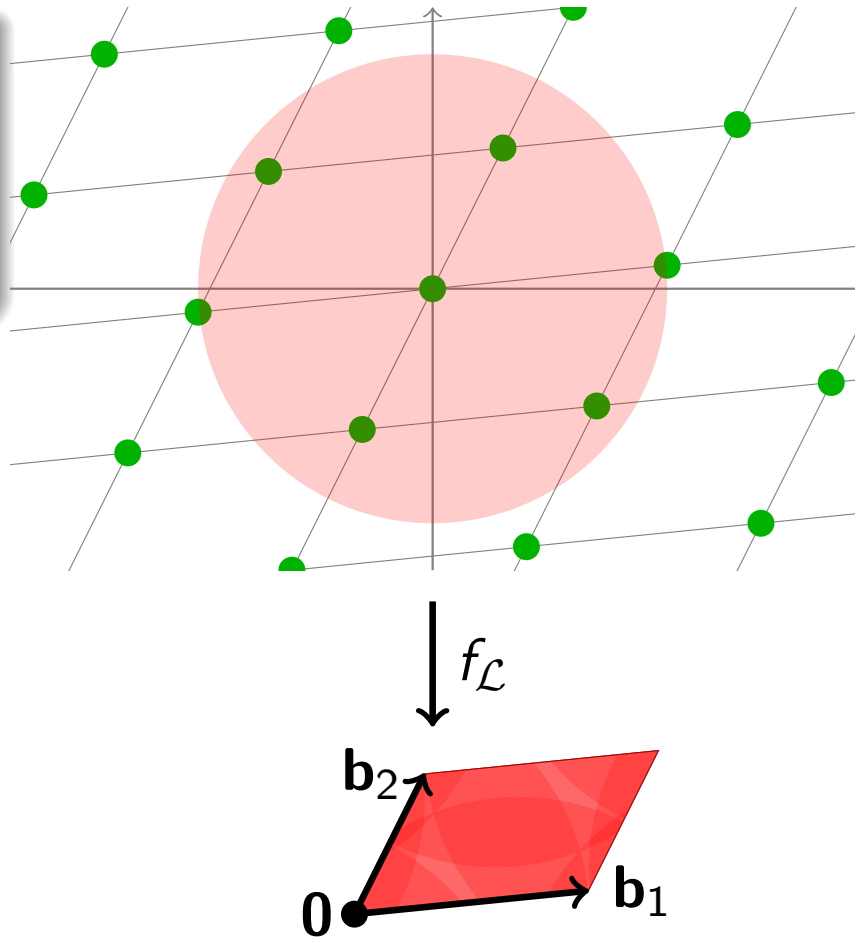


SIS/LWE as CVP

Candidate OWF

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

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



SIS/LWE as CVP

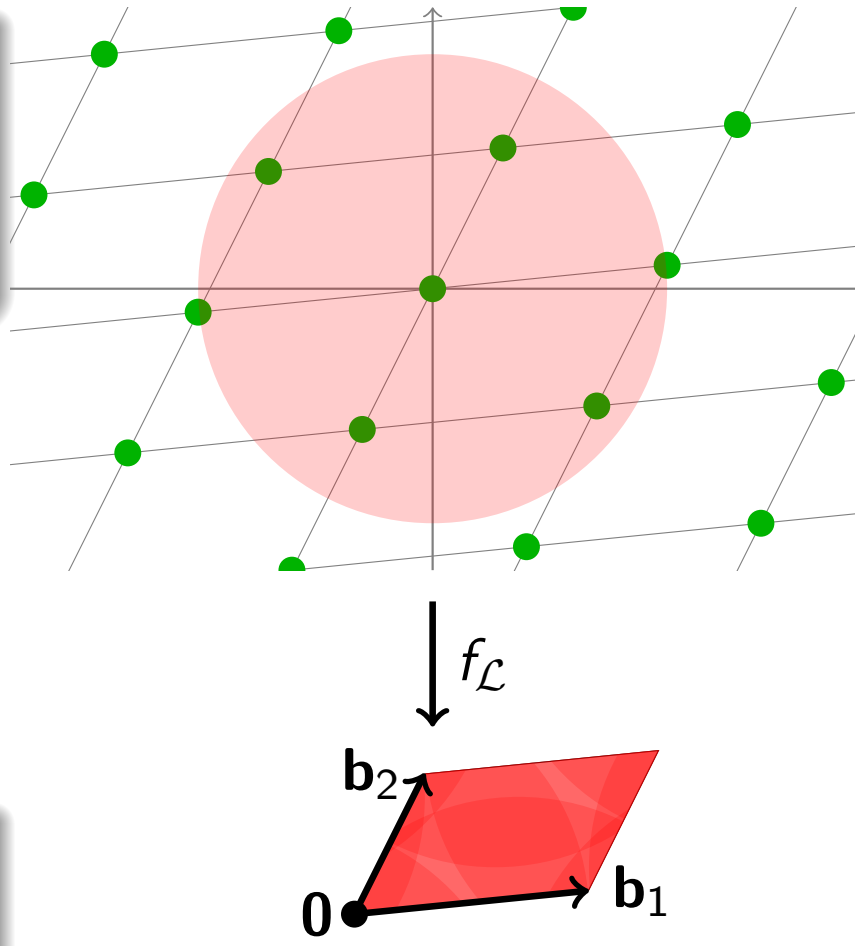
Candidate OWF

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

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

Question

Are these functions cryptographically hard to invert?



SIS Property: Regularity

$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Property: Regularity

$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

SIS Property: Regularity

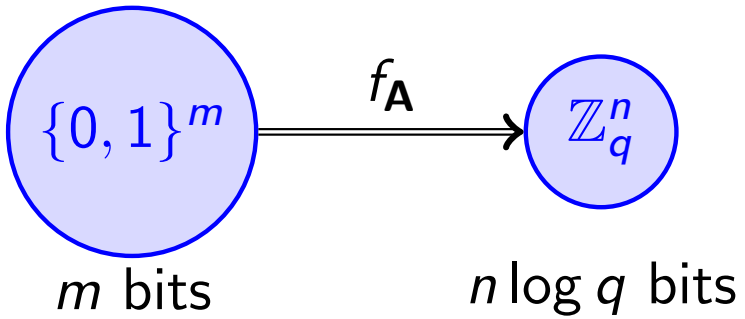
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

Pairwise independence:

- Fix $\mathbf{x_1} \neq \mathbf{x_2} \in \{0,1\}^m$,
- Random $\mathbf{A}$
- $f_{\mathbf{A}}(\mathbf{x}_1)$ and $f_{\mathbf{A}}(\mathbf{x}_2)$ are independent.



SIS Property: Regularity

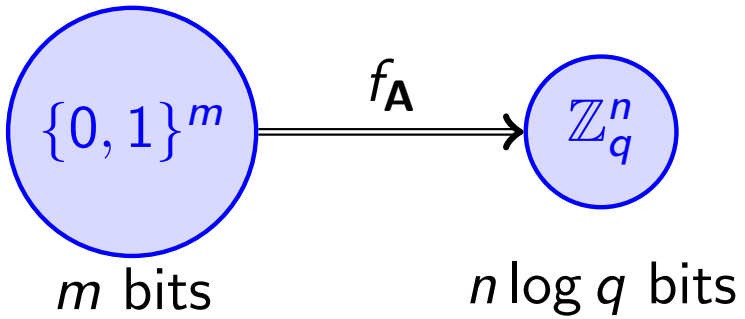
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

Pairwise independence:

- Fix $\mathbf{x_1} \neq \mathbf{x_2} \in \{0,1\}^m$,
- Random $\mathbf{A}$
- $f_{\mathbf{A}}(\mathbf{x}_1)$ and $f_{\mathbf{A}}(\mathbf{x}_2)$ are independent.



Lemma (Leftover Hash Lemma)

Pairwise Indepencence + Compression $\Longrightarrow$ Regular

SIS Property: Regularity

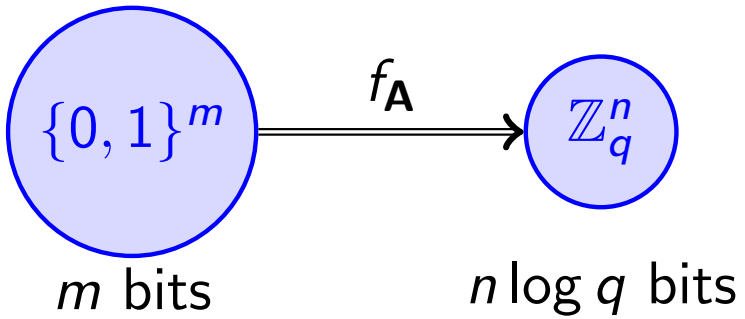
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

Pairwise independence:

- Fix $\mathbf{x_1} \neq \mathbf{x_2} \in \{0,1\}^m$,
- Random $\mathbf{A}$
- $f_{\mathbf{A}}(\mathbf{x}_1)$ and $f_{\mathbf{A}}(\mathbf{x}_2)$ are independent.



Lemma (Leftover Hash Lemma)

Pairwise Indepencence + Compression $\Longrightarrow$ Regular

$f_{\mathbf{A}} : (U(\{0,1\}^n)) \approx U(\mathbb{Z}_q^n)$ maps uniform to uniform.

SIS Application: Commitment

- Choose $\mathbf{A_1}, \mathbf{A_2}$ at random

SIS Application: Commitment

- Choose $\mathbf{A_1}, \mathbf{A_2}$ at random
- message $\mathbf{m} \in \{0,1\}^m$ and randomness $\mathbf{r} \in \{0,1\}^m$

SIS Application: Commitment

- Choose $\mathbf{A_1}, \mathbf{A_2}$ at random
- message $\mathbf{m} \in \{0,1\}^m$ and randomness $\mathbf{r} \in \{0,1\}^m$
- Commitment: $C(\mathbf{m}, \mathbf{r}) = f_{[\mathbf{A_1}, \mathbf{A_2}]}(\mathbf{m}, \mathbf{r}) = \mathbf{A_1 m} + \mathbf{A_2 r}$.

SIS Application: Commitment

- Choose $\mathbf{A_1}, \mathbf{A_2}$ at random
- message $\mathbf{m} \in \{0,1\}^m$ and randomness $\mathbf{r} \in \{0,1\}^m$
- Commitment: $C(\mathbf{m}, \mathbf{r}) = f_{[\mathbf{A_1}, \mathbf{A_2}]}(\mathbf{m}, \mathbf{r}) = \mathbf{A_1 m} + \mathbf{A_2 r}$.
- Hiding Property: $C(\mathbf{m})$ hides the message because $\mathbf{A_2 r} = f_{\mathbf{A}_2}(\mathbf{r}) \approx U(\mathbb{Z}_q^n)$

SIS Application: Commitment

- Choose $\mathbf{A_1}, \mathbf{A_2}$ at random
- message $\mathbf{m} \in \{0,1\}^m$ and randomness $\mathbf{r} \in \{0,1\}^m$
- Commitment: $C(\mathbf{m}, \mathbf{r}) = f_{[\mathbf{A_1}, \mathbf{A_2}]}(\mathbf{m}, \mathbf{r}) = \mathbf{A_1 m} + \mathbf{A_2 r}$.
- Hiding Property: $C(\mathbf{m})$ hides the message because $\mathbf{A_2 r} = f_{\mathbf{A}_2}(\mathbf{r}) \approx U(\mathbb{Z}_q^n)$
- Binding Property: Finding $(m, r) \neq (m', r')$ such that $C(\mathbf{m}, \mathbf{r}) = C(\mathbf{m}', \mathbf{r}')$ breaks the collision resistance of $f_{[\mathbf{A_1}, \mathbf{A_2}]}$

SIS Property: (Approximate) Linear Homomorphism

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

- The SIS function is linearly homomorphic:

$$f_{\mathbf{A}}(\mathbf{x}_1) + f_{\mathbf{A}}(\mathbf{x}_2) = f_{\mathbf{A}}(\mathbf{x}_1 + \mathbf{x}_2)$$

SIS Property: (Approximate) Linear Homomorphism

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

- The SIS function is linearly homomorphic:

$$f_{\mathbf{A}}(\mathbf{x}_1) + f_{\mathbf{A}}(\mathbf{x}_2) = f_{\mathbf{A}}(\mathbf{x}_1 + \mathbf{x}_2)$$

- Homomorphism is only approximate:
  - If $\mathbf{x}_1, \mathbf{x}_2$ are small, then also $\mathbf{x}_1 + \mathbf{x}_2$ is small
  - However, $\mathbf{x}_1 + \mathbf{x}_2$ can be slightly larger than $\mathbf{x}_1, \mathbf{x}_2$
  - Domain of $f_{\mathbf{A}}$ is not closed under $+$

SIS Property: (Approximate) Linear Homomorphism

SIS Function

$$\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \quad \mathbf{x} \in \{0,1\}^m, \quad f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q \in \mathbb{Z}_q^n$$

- The SIS function is linearly homomorphic:

$$f_{\mathbf{A}}(\mathbf{x}_1) + f_{\mathbf{A}}(\mathbf{x}_2) = f_{\mathbf{A}}(\mathbf{x}_1 + \mathbf{x}_2)$$

- Homomorphism is only approximate:
  - If $\mathbf{x}_1, \mathbf{x}_2$ are small, then also $\mathbf{x}_1 + \mathbf{x}_2$ is small
  - However, $\mathbf{x}_1 + \mathbf{x}_2$ can be slightly larger than $\mathbf{x}_1, \mathbf{x}_2$
  - Domain of $f_{\mathbf{A}}$ is not closed under $+$
- $f_{\mathbf{A}}$ is also key-homomorphic:

$$f_{\mathbf{A}_1}(\mathbf{x}) + f_{\mathbf{A}_2}(\mathbf{x}) = f_{\mathbf{A}_1 + \mathbf{A}_2}(\mathbf{x})$$

SIS Application: One-Time Signatures

- Extend $f_{\mathbf{A}}$ to matrices $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$:

$$f_{\mathbf{A}}(\mathbf{X}) = [f_{\mathbf{A}}(\mathbf{x}_1), \ldots, f_{\mathbf{A}}(\mathbf{x}_l)] = \mathbf{AX} \pmod{q}$$

SIS Application: One-Time Signatures

- Extend $f_{\mathbf{A}}$ to matrices $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$:

$$f_{\mathbf{A}}(\mathbf{X}) = [f_{\mathbf{A}}(\mathbf{x}_1), \ldots, f_{\mathbf{A}}(\mathbf{x}_l)] = \mathbf{AX} \pmod{q}$$

- Key Generation:
  - Public Parameter: SIS function key $\mathbf{A}$
  - Secret Key: $sk = (\mathbf{X}, \mathbf{x})$ two (small) inputs to $f_{\mathbf{A}}$
  - Public Key: $pk = (\mathbf{Y} = f_{\mathbf{A}}(\mathbf{X}), \mathbf{y} = f_{\mathbf{A}}(\mathbf{x}))$ image of sk under $f_{\mathbf{A}}$

SIS Application: One-Time Signatures

- Extend $f_{\mathbf{A}}$ to matrices $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$:

$$f_{\mathbf{A}}(\mathbf{X}) = [f_{\mathbf{A}}(\mathbf{x}_1), \ldots, f_{\mathbf{A}}(\mathbf{x}_l)] = \mathbf{AX} \pmod{q}$$

- Key Generation:
  - Public Parameter: SIS function key $\mathbf{A}$
  - Secret Key: $sk = (\mathbf{X}, \mathbf{x})$ two (small) inputs to $f_{\mathbf{A}}$
  - Public Key: $pk = (\mathbf{Y} = f_{\mathbf{A}}(\mathbf{X}), \mathbf{y} = f_{\mathbf{A}}(\mathbf{x}))$ image of sk under $f_{\mathbf{A}}$
- Message: short vector $\mathbf{m} \in \{0,1\}^l$
- $Sign(sk, \mathbf{m}) = \mathbf{Xm} + \mathbf{x}$, linear combination of secret key

SIS Application: One-Time Signatures

- Extend $f_{\mathbf{A}}$ to matrices $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$:

$$f_{\mathbf{A}}(\mathbf{X}) = [f_{\mathbf{A}}(\mathbf{x}_1), \ldots, f_{\mathbf{A}}(\mathbf{x}_l)] = \mathbf{AX} \pmod{q}$$

- Key Generation:
  - Public Parameter: SIS function key $\mathbf{A}$
  - Secret Key: $sk = (\mathbf{X}, \mathbf{x})$ two (small) inputs to $f_{\mathbf{A}}$
  - Public Key: $pk = (\mathbf{Y} = f_{\mathbf{A}}(\mathbf{X}), \mathbf{y} = f_{\mathbf{A}}(\mathbf{x}))$ image of sk under $f_{\mathbf{A}}$
- Message: short vector $\mathbf{m} \in \{0,1\}^l$
- $Sign(sk, \mathbf{m}) = \mathbf{Xm} + \mathbf{x}$, linear combination of secret key
- $Verify(pk, \mathbf{m}, \sigma)$ uses homomorphic properties to check that

$$f_{\mathbf{A}}(\sigma) = f_{\mathbf{A}}(\mathbf{Xm} + \mathbf{x}) = f_{\mathbf{A}}(\mathbf{X})\mathbf{m} + f_{\mathbf{A}}(\mathbf{x}) = \mathbf{Ym} + \mathbf{y}$$

Pseudorandomness of LWE

LWE(k, q, m, χ) distribution: $[\mathbf{A}, \mathbf{b}] \in \mathbb{Z}_q^{m \times (k+1)}$ where

- $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times k}$
- $\mathbf{s} \leftarrow \mathbb{Z}_q^k$, $\mathbf{e} \leftarrow \chi^m$
- $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$

Definition (Search LWE)

Given $[\mathbf{A}, \mathbf{b}]$ recover $\mathbf{s}$ and $\mathbf{e} = \mathbf{b} - \mathbf{A}\mathbf{s}$

Definition (Decision LWE)

Distinguish $[\mathbf{A}, \mathbf{b}]$ from the uniform distribution over $\mathbb{Z}_q^{m \times (k+1)}$

Pseudorandomness of LWE

LWE(k, q, m, χ) distribution: $[\mathbf{A}, \mathbf{b}] \in \mathbb{Z}_q^{m \times (k+1)}$ where

- $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times k}$
- $\mathbf{s} \leftarrow \mathbb{Z}_q^k$, $\mathbf{e} \leftarrow \chi^m$
- $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$

Definition (Search LWE)

Given $[\mathbf{A}, \mathbf{b}]$ recover $\mathbf{s}$ and $\mathbf{e} = \mathbf{b} - \mathbf{A}\mathbf{s}$

Definition (Decision LWE)

Distinguish $[\mathbf{A}, \mathbf{b}]$ from the uniform distribution over $\mathbb{Z}_q^{m \times (k+1)}$

Theorem (Search to Decision Reduction (informal))

For a wide range of parameters, if Search LWE is hard, then Decision LWE is hard.

Encrypting with LWE

Idea: if $[\mathbf{A}, \mathbf{b}]$ is pseudorandom, then we can use it to mask a message
Private Key Encryption

- $\mathtt{Gen}(k)$: output random $\mathbf{s} \leftarrow \mathbb{Z}_q^k$
- $\mathtt{Enc}_{\mathbf{s}}(\mathbf{m})$:
  - Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times k}$ and $\mathbf{e} \leftarrow \chi^m$ at random
  - Let $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \pmod{q}$
  - Output $[\mathbf{A}, \mathbf{b} + \mathbf{m}]$

Encrypting with LWE

Idea: if $[\mathbf{A}, \mathbf{b}]$ is pseudorandom, then we can use it to mask a message
Private Key Encryption

- $\mathtt{Gen}(k)$: output random $\mathbf{s} \leftarrow \mathbb{Z}_q^k$
- $\mathtt{Enc}_{\mathbf{s}}(\mathbf{m})$:
  - Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times k}$ and $\mathbf{e} \leftarrow \chi^m$ at random
  - Let $\mathbf{b} = \mathbf{As} + \mathbf{e} \pmod q$
  - Output $[\mathbf{A}, \mathbf{b} + \mathbf{m}]$
- $\mathtt{Dec}'_{\mathbf{s}}([\mathbf{A}, \mathbf{c}]) = \mathbf{c} - \mathbf{As} = \mathbf{m} + \mathbf{e}$
- Problems:
  - Decryption is "approximately" correct [CKKS 2016]
  - A passive adversary can completely break the scheme [Li,M. 2021]
- Solution: encode $\mathbf{m}$ with an error correcting code

Regev (private key) encryption

- Plaintext modulus: $p \ll q$
- Message: $\mathbf{m} \in \mathbb{Z}_p^m$
- Scaling factor: $\Delta = \left\lceil \frac{q}{p} \right\rfloor$
- $\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) = [\mathbf{A}, \mathbf{b} + \Delta\mathbf{m}]$ where $\mathbf{b} = \mathbf{As} + \mathbf{e} \pmod{q}$
- $\mathtt{Dec}_{\mathbf{s}}([\mathbf{A}, \mathbf{c}]) = \left\lceil \frac{\mathbf{c}-\mathbf{As}}{\Delta} \right\rfloor \pmod{p}$

Regev (private key) encryption

- Plaintext modulus: $p \ll q$
- Message: $\mathbf{m} \in \mathbb{Z}_p^m$
- Scaling factor: $\Delta = \left\lceil \frac{q}{p} \right\rfloor$
- $\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) = [\mathbf{A}, \mathbf{b} + \Delta\mathbf{m}]$ where $\mathbf{b} = \mathbf{As} + \mathbf{e} \pmod q$
- $\mathtt{Dec}_{\mathbf{s}}([\mathbf{A}, \mathbf{c}]) = \left\lceil \frac{\mathbf{c}-\mathbf{As}}{\Delta} \right\rfloor \pmod p$

Theorem (Correctness)

If $\|\chi\|_\infty < \Delta/2$ *then*

$$\mathtt{Dec}_{\mathbf{s}}(\mathtt{Enc}_{\mathbf{s}}(\mathbf{m})) = \left\lceil \frac{(\mathbf{As} + \mathbf{e} + \Delta m) - \mathbf{As}}{\Delta} \right\rfloor = \mathbf{m} + \left\lceil \frac{\mathbf{e}}{\Delta} \right\rfloor = \mathbf{m}.$$

Additive Homomorphism

The sum of two encryption

$$\begin{aligned} \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1) &= [\mathbf{A}_1, \mathbf{A}_1\mathbf{s} + \mathbf{e}_1 + \frac{q}{p}\mathbf{m}_1] \\ \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_2) &= [\mathbf{A}_2, \mathbf{A}_2\mathbf{s} + \mathbf{e}_2 + \frac{q}{p}\mathbf{m}_2] \end{aligned}$$

is an encryption of the sum

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1) + \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_2) = [\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} + \frac{q}{p}\mathbf{m}] = \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1 + \mathbf{m}_2)$$

where

- $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 \pmod q$
- $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2 \pmod q$
- $\mathbf{m} = \mathbf{m}_1 + \mathbf{m}_2 \pmod p$

Error growth

When adding two encryptions

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1; \mathbf{e}_1) + \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_2; \mathbf{e}_2) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1 + \mathbf{m}_2; \mathbf{e}_1 + \mathbf{e}_2)$$

with small errors $\|\mathbf{e}_1\|, \|\mathbf{e}_2\| \leq \beta$, the result is an encryption of $\mathbf{m}_1 + \mathbf{m}_2$ with slightly larger error

$$\|\mathbf{e}_1 + \mathbf{e}_2\| \leq \|\mathbf{e}_1\| + \|\mathbf{e}_2\| \leq 2\beta.$$

Error growth

When adding two encryptions

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1; \mathbf{e}_1) + \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_2; \mathbf{e}_2) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}_1 + \mathbf{m}_2; \mathbf{e}_1 + \mathbf{e}_2)$$

with small errors $\|\mathbf{e}_1\|, \|\mathbf{e}_2\| \leq \beta$, the result is an encryption of $\mathbf{m}_1 + \mathbf{m}_2$ with slightly larger error

$$\|\mathbf{e}_1 + \mathbf{e}_2\| \leq \|\mathbf{e}_1\| + \|\mathbf{e}_2\| \leq 2\beta.$$

Remarks:

- If $\mathbf{e}_1, \mathbf{e}_2$ are random and independent, their sum grows more like $\sqrt{2}\beta$
- If $|\chi| \ll (q/p)$, we can add several ciphertexts, and the results will still decrypt correctly
- If we keep adding ciphertexts, the error may become too big, and decryption will fail

Linear/Affine functions?

Start from a fresh ciphertext with small $\|\mathbf{e}\|_\infty \leq \beta \ll \Delta$

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) \quad = \quad [\mathbf{A}, \mathbf{As} + \mathbf{e} + \Delta\mathbf{m}]$$

What about multiplying ciphertexts by a constant or computing linear or affine function?

Linear/Affine functions?

Start from a fresh ciphertext with small $\|\mathbf{e}\|_\infty \leq \beta \ll \Delta$

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) \quad = \quad [\mathbf{A}, \mathbf{As} + \mathbf{e} + \Delta\mathbf{m}]$$

What about multiplying ciphertexts by a constant or computing linear or affine function?

$$\begin{aligned}
t \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) &= [(t\mathbf{A}), (t\mathbf{A})\mathbf{s} + t\mathbf{e} + \Delta(t\mathbf{m})] = \mathtt{Enc}_{\mathbf{s}}(t\mathbf{m}) \\
\mathbf{T} \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) &= [(\mathbf{TA}), (\mathbf{TA})\mathbf{s} + \mathbf{Te} + \Delta(\mathbf{Tm})] = \mathtt{Enc}_{\mathbf{s}}(\mathbf{Tm}) \\
\mathbf{T} \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) + [\mathbf{0}, \Delta\mathbf{c}] &= [(\mathbf{TA}), (\mathbf{TA})\mathbf{s} + \mathbf{Te} + \Delta(\mathbf{Tm} + \mathbf{c})] \\
&= \mathtt{Enc}_{\mathbf{s}}(\mathbf{Tm} + \mathbf{c})
\end{aligned}$$

Linear/Affine functions?

Start from a fresh ciphertext with small $\|\mathbf{e}\|_\infty \leq \beta \ll \Delta$

$$\mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) \quad = \quad [\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} + \Delta\mathbf{m}]$$

What about multiplying ciphertexts by a constant or computing linear or affine function?

$$\begin{aligned}
t \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) &= [(t\mathbf{A}), (t\mathbf{A})\mathbf{s} + t\mathbf{e} + \Delta(t\mathbf{m})] = \mathtt{Enc}_{\mathbf{s}}(t\mathbf{m}) \\
\mathbf{T} \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) &= [(\mathbf{T}\mathbf{A}), (\mathbf{T}\mathbf{A})\mathbf{s} + \mathbf{T}\mathbf{e} + \Delta(\mathbf{T}\mathbf{m})] = \mathtt{Enc}_{\mathbf{s}}(\mathbf{T}\mathbf{m}) \\
\mathbf{T} \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}) + [\mathbf{0}, \Delta\mathbf{c}] &= [(\mathbf{T}\mathbf{A}), (\mathbf{T}\mathbf{A})\mathbf{s} + \mathbf{T}\mathbf{e} + \Delta(\mathbf{T}\mathbf{m} + \mathbf{c})] \\
&= \mathtt{Enc}_{\mathbf{s}}(\mathbf{T}\mathbf{m} + \mathbf{c})
\end{aligned}$$

Remarks:

- It works, but error grows to $\|\mathbf{T}\mathbf{e}\|_\infty \leq m \cdot \|\mathbf{T}\|_\infty \cdot \beta$
- We need $\mathbf{T}$ to be small for final result to be correct

Public Key Encryption (Regev 2005)

We will use homomorphic properties to transform private key encryption to public key encryption

- Start from LWE private key encryption scheme $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$
- Set parameters n, k, q, χ so to support the addition of w ciphertexts

Public Key Encryption scheme $(\mathtt{Gen}', \mathtt{Enc}', \mathtt{Dec})$:

- Public key: $\mathbf{P} = \mathtt{Enc}_{\mathbf{s}}(\mathbf{0})$
- $\mathtt{Enc}'_{\mathbf{P}}(\mathbf{m}) = [\mathbf{O}, \Delta\mathbf{m}] + \mathbf{R} \cdot \mathbf{P}$ where $\mathbf{R} \leftarrow \{0,1\}^{w' \times w}$

Public Key Encryption (Regev 2005)

We will use homomorphic properties to transform private key encryption to public key encryption

- Start from LWE private key encryption scheme $(\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$
- Set parameters n, k, q, χ so to support the addition of w ciphertexts

Public Key Encryption scheme $(\mathtt{Gen}', \mathtt{Enc}', \mathtt{Dec})$:

- Public key: $\mathbf{P} = \mathtt{Enc}_{\mathbf{s}}(\mathbf{0})$
- $\mathtt{Enc}'_{\mathbf{P}}(\mathbf{m}) = [\mathbf{O}, \Delta\mathbf{m}] + \mathbf{R} \cdot \mathbf{P}$ where $\mathbf{R} \leftarrow \{0,1\}^{w' \times w}$

Theorem (Correctness)

$$Enc'_{\mathbf{P}}(\mathbf{m}) = [\mathbf{O}, \Delta\mathbf{m}] + \mathbf{R}\mathtt{Enc}_{\mathbf{s}}(\mathbf{0}) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{m} + \mathbf{R} \cdot \mathbf{0}) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{m})$$

Theorem (Security)

If w is large enough, $\mathtt{Enc}'_{\mathbf{P}}(\mathbf{m}) \approx \mathtt{Enc}_{\mathbf{s}}(\mathbf{m}; \chi')$.

Optimized Public Key Encryption (Lindner,Peikert 2011)

- LWE PKE = "Symmetric Encryption + Linearity"
- We can also describe it directly in terms of matrices:
  - Public Key: $\mathbf{P} = g_{\mathbf{A}}(\mathbf{S}, \mathbf{E}) = [\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}]$
  - Encryption (of 0): $\mathtt{Enc}_{\mathbf{P}}(0)^t = \mathbf{P}^t\mathbf{R} = f_{\mathbf{P}^t}(\mathbf{R})$

Optimized Public Key Encryption (Lindner,Peikert 2011)

- LWE PKE = "Symmetric Encryption + Linearity"
- We can also describe it directly in terms of matrices:
  - Public Key: $\mathbf{P} = g_{\mathbf{A}}(\mathbf{S}, \mathbf{E}) = [\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}]$
  - Encryption (of 0): $\mathtt{Enc}_{\mathbf{P}}(0)^t = \mathbf{P}^t\mathbf{R} = f_{\mathbf{P}^t}(\mathbf{R})$
- We can use a smaller dimensional $\mathbf{A}, \mathbf{P}$ (and improve efficiency) using the SIS/HNF variant of LWE:
  - $\mathbf{A}, \mathbf{S}, \mathbf{E}, \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 \in \mathbb{Z}_q^{k \times k}$
  - Public Key: $\mathbf{P} = [\mathbf{A}, f_{[\mathbf{A},\mathbf{I}]}(\mathbf{S}, \mathbf{E})] = [\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}] \in \mathbb{Z}_q^{k \times 2k}$
  - Expanded public key: $[\mathbf{P}^t, \mathbf{I}] \in \mathbb{Z}_q^{2k \times 3k}$

Optimized Public Key Encryption (Lindner,Peikert 2011)

- LWE PKE = "Symmetric Encryption + Linearity"
- We can also describe it directly in terms of matrices:
  - Public Key: $\mathbf{P} = g_{\mathbf{A}}(\mathbf{S}, \mathbf{E}) = [\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}]$
  - Encryption (of 0): $\mathtt{Enc}_{\mathbf{P}}(0)^t = \mathbf{P}^t\mathbf{R} = f_{\mathbf{P}^t}(\mathbf{R})$
- We can use a smaller dimensional $\mathbf{A}, \mathbf{P}$ (and improve efficiency) using the SIS/HNF variant of LWE:
  - $\mathbf{A}, \mathbf{S}, \mathbf{E}, \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3 \in \mathbb{Z}_q^{k \times k}$
  - Public Key: $\mathbf{P} = [\mathbf{A}, f_{[\mathbf{A},\mathbf{I}]}(\mathbf{S}, \mathbf{E})] = [\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}] \in \mathbb{Z}_q^{k \times 2k}$
  - Expanded public key: $[\mathbf{P}^t, \mathbf{I}] \in \mathbb{Z}_q^{2k \times 3k}$
  - Encryption (of 0):

$$\mathtt{Enc}_{\mathbf{P}}(0)^t = f_{[\mathbf{P}^t,\mathbf{I}]}(\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3) = \mathbf{P}^t\mathbf{R}_1 + \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{R}_3 \end{bmatrix} \in \mathbb{Z}_q^{2k \times k}$$

- This is the blueprint followed by the ML-KEM (Kyber) cryptosystem standardized by NIST, but with $\mathbb{Z}_q^{k \times k}$ replaced by more compact "algebraically structured" matrices

Gadgets

Questions:

- Can we encrypt messages $\mathbf{m} \in \mathbb{Z}_q$ with plaintext modulus $p = q$?
- Can we multiply ciphertext by any $t \in \mathbb{Z}_q$ with error growth $\ll q$?

Gadgets

Questions:

- Can we encrypt messages $\mathbf{m} \in \mathbb{Z}_q$ with plaintext modulus $p = q$?
- Can we multiply ciphertext by any $t \in \mathbb{Z}_q$ with error growth $\ll q$?

Assume for simplicity $q = 2^\ell$. Define the "gadget" vector

$$\mathbf{g}^t = [1, 2, 4, 8, ..., 2^i, ..., 2^{\ell-1}]$$

This vector has two fundamental properties:

Gadgets

Questions:

- Can we encrypt messages $\mathbf{m} \in \mathbb{Z}_q$ with plaintext modulus $p = q$?
- Can we multiply ciphertext by any $t \in \mathbb{Z}_q$ with error growth $\ll q$?

Assume for simplicity $q = 2^\ell$. Define the "gadget" vector

$$\mathbf{g}^t = [1, 2, 4, 8, ..., 2^i, ..., 2^{\ell-1}]$$

This vector has two fundamental properties:

1. Any $c \in \mathbb{Z}_q$ can be written as $\mathbf{x}^t \cdot \mathbf{g}$ for some small $\mathbf{x} \in \{0,1\}^\ell$ (written $\mathbf{x} = \mathbf{g}^{-1}(c)$).

Gadgets

Questions:

- Can we encrypt messages $\mathbf{m} \in \mathbb{Z}_q$ with plaintext modulus $p = q$?
- Can we multiply ciphertext by any $t \in \mathbb{Z}_q$ with error growth $\ll q$?

Assume for simplicity $q = 2^\ell$. Define the "gadget" vector

$$\mathbf{g}^t = [1, 2, 4, 8, ..., 2^i, ..., 2^{\ell-1}]$$

This vector has two fundamental properties:

1. Any $c \in \mathbb{Z}_q$ can be written as $\mathbf{x}^t \cdot \mathbf{g}$ for some small $\mathbf{x} \in \{0,1\}^\ell$ (written $\mathbf{x} = \mathbf{g}^{-1}(c)$).
2. Given $\mathbf{c} = \mathbf{g} \cdot m + \mathbf{e}$ for some $\|\mathbf{e}\|_\infty < q/4$, we can recover $m \in \mathbb{Z}_q$ (written $m = \lceil \mathbf{c} \rfloor_{\mathbf{g}}$)

The "Gadget-LWE" encryption scheme

Similar to LWE, but using the gadget vector $\mathbf{g}$ instead of scaling factor Δ to encode the message.

- Plaintext modulus $p = q$
- $\mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = [\mathbf{A}, \mathbf{As} + \mathbf{e} + \mathbf{g} \cdot m]$
- $\mathtt{Dec}_{\mathbf{s}}^{\mathbf{g}}([A, \mathbf{c}]) = \lceil \mathbf{c} - \mathbf{As} \rfloor_{\mathbf{g}}$

The "Gadget-LWE" encryption scheme

Similar to LWE, but using the gadget vector $\mathbf{g}$ instead of scaling factor Δ to encode the message.

- Plaintext modulus $p = q$
- $\mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = [\mathbf{A}, \mathbf{As} + \mathbf{e} + \mathbf{g} \cdot m]$
- $\mathtt{Dec}_{\mathbf{s}}^{\mathbf{g}}([A, \mathbf{c}]) = \lceil \mathbf{c} - \mathbf{As} \rfloor_{\mathbf{g}}$

Bonus: we can multiply ciphertexts by arbitrary constants (or linear transformations)

$$\begin{aligned} c \odot \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) &= \mathbf{g}^{-1}(c)^t \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}m) \\ &= \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}^{-1}(c)^t \cdot \mathbf{g} \cdot m) \\ &= \mathtt{Enc}_{\mathbf{s}}(c \cdot m) \end{aligned}$$

The "Gadget-LWE" encryption scheme

Similar to LWE, but using the gadget vector $\mathbf{g}$ instead of scaling factor Δ to encode the message.

- Plaintext modulus $p = q$
- $\mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = [\mathbf{A}, \mathbf{As} + \mathbf{e} + \mathbf{g} \cdot m]$
- $\mathtt{Dec}_{\mathbf{s}}^{\mathbf{g}}([A, \mathbf{c}]) = \lceil \mathbf{c} - \mathbf{As} \rfloor_{\mathbf{g}}$

Bonus: we can multiply ciphertexts by arbitrary constants (or linear transformations)

$$\begin{aligned} c \odot \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) &= \mathbf{g}^{-1}(c)^t \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}m) \\ &= \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}^{-1}(c)^t \cdot \mathbf{g} \cdot m) \\ &= \mathtt{Enc}_{\mathbf{s}}(c \cdot m) \end{aligned}$$

Similarly: $(\mathbf{g} \cdot c) \odot \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}cm) = \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(cm)$

The "Gadget-LWE" encryption scheme

Similar to LWE, but using the gadget vector $\mathbf{g}$ instead of scaling factor Δ to encode the message.

- Plaintext modulus $p = q$
- $\mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = [\mathbf{A}, \mathbf{As} + \mathbf{e} + \mathbf{g} \cdot m]$
- $\mathtt{Dec}_{\mathbf{s}}^{\mathbf{g}}([A, \mathbf{c}]) = \lceil \mathbf{c} - \mathbf{As} \rfloor_{\mathbf{g}}$

Bonus: we can multiply ciphertexts by arbitrary constants (or linear transformations)

$$\begin{aligned} c \odot \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) &= \mathbf{g}^{-1}(c)^t \cdot \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}m) \\ &= \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}^{-1}(c)^t \cdot \mathbf{g} \cdot m) \\ &= \mathtt{Enc}_{\mathbf{s}}(c \cdot m) \end{aligned}$$

Similarly: $(\mathbf{g} \cdot c) \odot \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(m) = \mathtt{Enc}_{\mathbf{s}}(\mathbf{g}cm) = \mathtt{Enc}_{\mathbf{s}}^{\mathbf{g}}(cm)$
Remark: error grows by $\ell = \log_2 q$ instead of q because $\mathbf{g}^{-1}(c) \in \{0,1\}^{\ell}$

Other gadgets

- The "powers-of-two" gadget is just the most common example.
- Easily generalize to arbitrary base:
  $\mathbf{g}^t = [1, B, B^2, \ldots, B^{\log_B q - 1}] \in \mathbb{Z}^\ell$, gives a trade-off between storage $\ell = \log_B q$ and error growth $B \cdot \ell$

Other gadgets

- The "powers-of-two" gadget is just the most common example.
- Easily generalize to arbitrary base:
  $\mathbf{g}^t = [1, B, B^2, \ldots, B^{\log_B q - 1}] \in \mathbb{Z}^\ell$, gives a trade-off between storage $\ell = \log_B q$ and error growth $B \cdot \ell$
- Another useful option is the "Chinese Remainder Theorem" (CRT) gadget $\mathbf{g}^t = [q/p_1, \ldots, q/p_\ell]$ where $q = \prod_i p_i$
- You can even combine a scaling factor $\Delta = q/p$ and a gadget $\mathbf{g}$ (mod p), and encode $m \in \mathbb{Z}_p$ as $\Delta \cdot (\mathbf{g}m) \in \mathbb{Z}_q^\ell$.
- Many optimizations and variants in lattice-based cryptography can be seen simply as a different choice of gadget $\mathbf{g}$

Application: Key-Switching

The key-switching problem: $\mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) \mapsto \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m)$

- You have a ciphertext $\mathbf{C} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) = [\mathbf{A}, \mathbf{b}]$ under a key $\mathbf{s}$
- You want to publish some information $\mathbf{W}$ that can be used to change the encryption key to some other $\mathbf{z}$, without leaking any information

Application: Key-Switching

The key-switching problem: $\mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) \mapsto \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m)$

- You have a ciphertext $\mathbf{C} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) = [\mathbf{A}, \mathbf{b}]$ under a key $\mathbf{s}$
- You want to publish some information $\mathbf{W}$ that can be used to change the encryption key to some other $\mathbf{z}$, without leaking any information

Solution: $\mathbf{W} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(\begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix})$

Theorem (Key-Switching)

$\mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}} \odot \mathbf{W} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m)$

Application: Key-Switching

The key-switching problem: $\mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) \mapsto \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m)$

- You have a ciphertext $\mathbf{C} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}}(m) = [\mathbf{A}, \mathbf{b}]$ under a key $\mathbf{s}$
- You want to publish some information $\mathbf{W}$ that can be used to change the encryption key to some other $\mathbf{z}$, without leaking any information

Solution: $\mathbf{W} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(\begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix})$

Theorem (Key-Switching)

$\mathtt{Enc}^{\mathbf{g}}_{\mathbf{s}} \odot \mathbf{W} = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m)$

$$\begin{aligned} \mathbf{C} \odot \mathbf{W} &= \mathbf{C} \odot \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(\begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}) = \mathtt{Enc}_{\mathbf{z}}(\mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}) \\ &= \mathtt{Enc}_{\mathbf{z}}(\mathbf{b} - \mathbf{A}\mathbf{s}) = \mathtt{Enc}_{\mathbf{z}}(\mathbf{g}m + \mathbf{e}) = \mathtt{Enc}^{\mathbf{g}}_{\mathbf{z}}(m) \end{aligned}$$

So far: summary

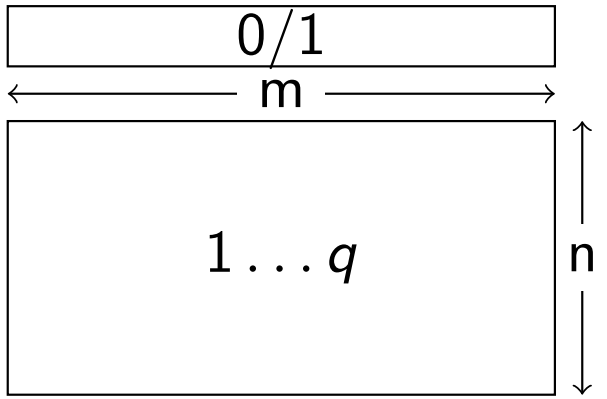
- SIS, LWE: hard lattice problems on random q-ary lattices $\Lambda_q(\mathbf{A})$
- Technical tool: Lattice gadgets
- Strong linear homomorphic properties
- Typical applications:
  - SIS: collision resistant hashing
  - LWE: encryption
- Main issue: Efficiency!

Part 2: Algebraic Lattices

- Structured matrices and Polynomial Rings
- RingSIS and RingLWE
- Algorithms: FFT and NTT
- Example: Kyber (ML-KEM)

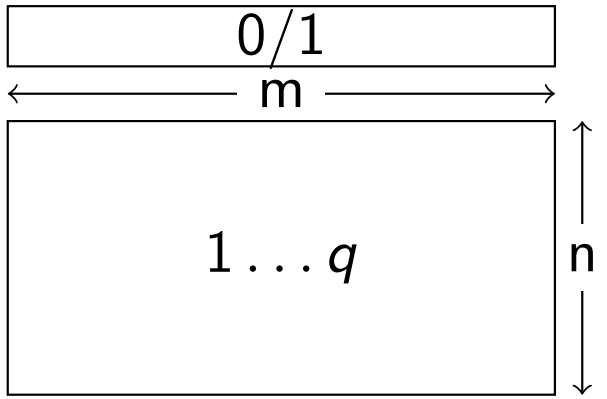
Efficiency of Ajtai's function

- $q = n^{O(1)}$, $m = O(n \log n) > n \log_2 q$
- E.g., $n = 64$, $q = 2^8$, $m = 1024$
- $f_{\mathbf{A}}$ maps 1024 bits to 512.



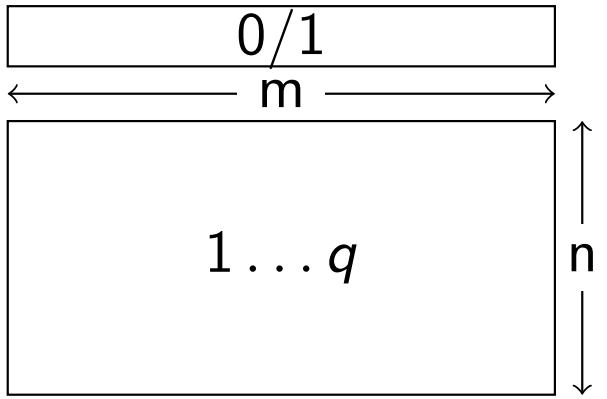
Efficiency of Ajtai's function

- $q = n^{O(1)}$, $m = O(n \log n) > n \log_2 q$
- E.g., $n = 64$, $q = 2^8$, $m = 1024$
- $f_{\mathbf{A}}$ maps 1024 bits to 512.
- Key size:
  $nm \log q = O(n^2 \log^2 n) = 2^{19} = 64KB$
- Runtime: $nm = O(n^2 \log n) = 2^{16}$
  arithmetic operations



Efficiency of Ajtai's function

- $q = n^{O(1)}$, $m = O(n \log n) > n \log_2 q$
- E.g., $n = 64$, $q = 2^8$, $m = 1024$
- $f_{\mathbf{A}}$ maps 1024 bits to 512.
- Key size:
  $nm \log q = O(n^2 \log^2 n) = 2^{19} = 64KB$
- Runtime: $nm = O(n^2 \log n) = 2^{16}$
  arithmetic operations
- Usable, but inefficient
  - Source of inefficiency: quadratic dependency in n



Problem

Can we do better than $O(n^2)$ complexity?

Efficient lattice based hashing

Idea: use structured matrix $\mathbf{A} = [\mathbf{A}^{(1)} \mid \ldots \mid \mathbf{A}^{(m/n)}]$, $\quad \mathbf{A}^{(i)} \in \mathbb{Z}_q^{n \times n}$.

Theorem (Micciancio 2002)

$f_{\mathbf{A}}$ is a one-way function, provided SVP and SIVP are hard to approximate within $\gamma \approx n$ in the worst case over cyclic lattices

$$\mathbf{A}^{(i)} = \begin{bmatrix} a_1^{(i)} & a_n^{(i)} & \cdots & a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \cdots & a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ a_n^{(i)} & a_{n-1}^{(i)} & \cdots & a_1^{(i)} \end{bmatrix}$$

- Similar idea first used by NTRU (1998)
- Novelty in Micciancio (2002): provably hard to invert

Efficient lattice based hashing

Idea: use structured matrix $\mathbf{A} = [\mathbf{A}^{(1)} \mid \ldots \mid \mathbf{A}^{(m/n)}]$, $\quad \mathbf{A}^{(i)} \in \mathbb{Z}_q^{n \times n}$.

Theorem (Micciancio 2002)

$f_{\mathbf{A}}$ is a one-way function, provided SVP and SIVP are hard to approximate within $\gamma \approx n$ in the worst case over cyclic lattices

$$\mathbf{A}^{(i)} = \begin{bmatrix} a_1^{(i)} & a_n^{(i)} & \cdots & a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \cdots & a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ a_n^{(i)} & a_{n-1}^{(i)} & \cdots & a_1^{(i)} \end{bmatrix}$$

- Similar idea first used by NTRU (1998)
- Novelty in Micciancio (2002): provably hard to invert

Can you find a collision? (mod 10)

1	4	3	8	6	4	9	0	2	6	4	5	3	2	7	1
8	1	4	3	0	6	4	9	5	2	6	4	1	3	2	7
3	8	1	4	9	0	6	4	4	5	2	6	7	1	3	2
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3

Can you find a collision? (mod 10)

1	0	0	-1	-1	1	1	0	0	0	1	1	1	0	-1	0	
1	4	3	8	6	4	9	0	2	6	4	5	3	2	7	1	5
8	1	4	3	0	6	4	9	5	2	6	4	1	3	2	7	4
3	8	1	4	9	0	6	4	4	5	2	6	7	1	3	2	8
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3	6

Can you find a collision? (mod 10)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	
1	4	3	8	6	4	9	0	2	6	4	5	3	2	7	1	0
8	1	4	3	0	6	4	9	5	2	6	4	1	3	2	7	0
3	8	1	4	9	0	6	4	4	5	2	6	7	1	3	2	0
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3	0

Can you find a collision? (mod 10)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
1	4	3	8	6	4	9	0	2	6	4	5	3	2	7	1		
8	1	4	3	0	6	4	9	5	2	6	4	1	3	2	7		
3	8	1	4	9	0	6	4	4	5	2	6	7	1	3	2		
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3		

6
6
6
6

9
9
9
9

7
7
7
7

3
3
3
3

Can you find a collision? (mod 10)

1	1	1	1	-1	-1	-1	-1	0	0	0	0	1	1	1	1		
1	4	3	8	6	4	9	0	2	6	4	5	3	2	7	1		0
8	1	4	3	0	6	4	9	5	2	6	4	1	3	2	7		0
3	8	1	4	9	0	6	4	4	5	2	6	7	1	3	2		0
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3		0

$$+1 \times \begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \end{bmatrix} \quad -1 \times \begin{bmatrix} 9 \\ 9 \\ 9 \\ 9 \end{bmatrix} \quad +0 \times \begin{bmatrix} 7 \\ 7 \\ 7 \\ 7 \end{bmatrix} \quad +1 \times \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

Remarks about proofs of security

- This function is essentially the compression function of hash function LASH, modeled after NTRU
- You can still "prove" security based on average case assumption: Breaking the above hash function is as hard as finding short vectors in a random lattice $\Lambda([\mathbf{A}^{(1)}|\ldots|\mathbf{A}^{(m/n)}])$
- . . . but we know the function is broken: The underlying random lattice distribution is weak!
- Conclusion: Assuming that a problem is hard on average-case is a really tricky business!

Can you find a collision now? (mod 10)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
1	-4	-3	-8	6	-4	-9	-0	2	-6	-4	-5	3	-2	-7	-1
8	1	-4	-3	0	6	-4	-9	5	2	-6	-4	1	3	-2	-7
3	8	1	-4	9	0	6	-4	4	5	2	-6	7	1	3	-2
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3

Can you find a collision now? (mod 10)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
1	-4	-3	-8	6	-4	-9	-0	2	-6	-4	-5	3	-2	-7	-1
8	1	-4	-3	0	6	-4	-9	5	2	-6	-4	1	3	-2	-7
3	8	1	-4	9	0	6	-4	4	5	2	-6	7	1	3	-2
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3

Theorem (trivial)

Finding collisions on the average is at least as hard as finding short vectors in the corresponding random lattices

Can you find a collision now? (mod 10)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
1	-4	-3	-8	6	-4	-9	-0	2	-6	-4	-5	3	-2	-7	-1
8	1	-4	-3	0	6	-4	-9	5	2	-6	-4	1	3	-2	-7
3	8	1	-4	9	0	6	-4	4	5	2	-6	7	1	3	-2
4	3	8	1	4	9	0	6	6	4	5	2	2	7	1	3

Theorem (trivial)

Finding collisions on the average is at least as hard as finding short vectors in the corresponding random lattices

Theorem (LM'07,PR'07)

Provably collision resistant, assuming the worst case hardness of approximating SVP and SIVP over anti-cyclic lattices.

Polynomial rings

- $\mathbb{Z}[X]$: set of polynomials with integer coefficients
- Monic polynomial $f(X) = X^d + f_{d-1}X^{d-1} + \cdot + f_1 \cdot X + f_0 \in \mathbb{Z}[X]$
- $a(X) \pmod{f(X)}$ has degree $< d$
- $R = \mathbb{Z}[X]/f(X) \equiv \mathbb{Z}^d$

Polynomial rings

- $\mathbb{Z}[X]$: set of polynomials with integer coefficients
- Monic polynomial $f(X) = X^d + f_{d-1}X^{d-1} + \cdot + f_1 \cdot X + f_0 \in \mathbb{Z}[X]$
- $a(X) \pmod{f(X)}$ has degree $< d$
- $R = \mathbb{Z}[X]/f(X) \equiv \mathbb{Z}^d$
- Matrix representation

$$M_{a(X)} = \left[(a(X)), (X \cdot a(X)), \ldots, (X^{d-1} \cdot a(X))\right] \in \mathbb{Z}^{d \times d}$$

  where $X^i \cdot a(X)$ is reduced $\pmod{f(X)}$
- $c(X) = a(X) \cdot b(X) \bmod f(X)$ then

$$M_{a(X)} \cdot b(X) = c(X)$$

$$M_{a(X)} \cdot M_{b(X)} = M_{c(X)}$$

Examples

- If $f(X) = X^d - 1$, then $X^d \equiv 1$ and

$$M_a = \begin{bmatrix} a_0 & a_{d-1} & \cdots & a_1 \\ a_1 & a_0 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & \cdots & a_0 \end{bmatrix}$$

Examples

- If $f(X) = X^d - 1$, then $X^d \equiv 1$ and

$$M_a = \begin{bmatrix} a_0 & a_{d-1} & \cdots & a_1 \\ a_1 & a_0 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & \cdots & a_0 \end{bmatrix}$$

- If $f(X) = X^d + 1$, then $X^d \equiv -1$ and

$$M_a = \begin{bmatrix} a_0 & -a_{d-1} & \cdots & -a_1 \\ a_1 & a_0 & \cdots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & \cdots & a_0 \end{bmatrix}$$

- What makes $X^d + 1$ better than $X^d - 1$?

Choosing $f(X)$

- $f(X) = X^d - 1$
  - Factors into $X^d - 1 = (X - 1) \cdot (X^{d-1} + X^{d-2} + \cdots X + 1)$
  - Ability to find collisions is closely related to existence of linear factor $(X - 1)$

Choosing $f(X)$

- $f(X) = X^d - 1$
  - Factors into $X^d - 1 = (X - 1) \cdot (X^{d-1} + X^{d-2} + \cdots X + 1)$
  - Ability to find collisions is closely related to existence of linear factor $(X - 1)$
- $f(X) = X^d + 1$
  - When $d = 2^k$, the polynomial $f(X)$ is irreducible
  - This is the most common choice in cryptography
  - Efficient, easier to implement
  - These are called "power-of-two" cyclotomic rings

Choosing $f(X)$

- $f(X) = X^d - 1$
  - Factors into $X^d - 1 = (X - 1) \cdot (X^{d-1} + X^{d-2} + \cdots X + 1)$
  - Ability to find collisions is closely related to existence of linear factor $(X - 1)$
- $f(X) = X^d + 1$
  - When $d = 2^k$, the polynomial $f(X)$ is irreducible
  - This is the most common choice in cryptography
  - Efficient, easier to implement
  - These are called "power-of-two" cyclotomic rings
- One may use other irreducible $f(X)$
  - For example, $f(X) = X^{p-1} + X^{p-2} + \cdots + X + 1$
  - Less convenient, harder to implement
- From now on assume $R = \mathbb{Z}[X]/(X^d + 1) \equiv \mathbb{Z}^d$ with $d = \dim(R) = 2^k$

Algebraic Number Theory

- $K = \mathbb{Q}[X]/f(X) \equiv \mathbb{Q}^d$ is called an Algebraic Number Field
  - May be defined for any irreducible $f(X) \in \mathbb{Z}[X]$
  - Here we focus on $f(X) = X^d + 1$ for $d = 2^k$
- $R \subset K$ is the "Ring of integers" of K
- Very special case: $d = 1 = 2^0$
  - $\mathbb{Q}[X]/(X+1) = \mathbb{Q}$
  - $\mathbb{Z}[X]/(X+1) = \mathbb{Z}$
- R generalizes $\mathbb{Z} \subset \mathbb{Q}$ to higher dimension $d = \dim(R)$
  - Ring element $a(X) \in R$ can be stored as vector in $\mathbb{Z}^d$
  - It represent a (structured) matrix in $\mathbb{Z}^{d \times d}$.
  - For $d = 256$ or higher, this is a huge saving in storage
- You don't need to known Algebraic Number Theory to understand Lattice-based cryptography

RingSIS

- Just like SIS, but using R instead of $\mathbb{Z}$
- Let $R_q = R/qR = \mathbb{Z}_q[X]/f(X)$
- Given $\mathbf{A} \in R^{n \times m}$, find "short" $\mathbf{x} \in R^m$ such that $\mathbf{Ax} = \mathbf{0} \pmod{q}$
- What is a "short" ring element $a(X) \in R$?

RingSIS

- Just like SIS, but using R instead of $\mathbb{Z}$
- Let $R_q = R/qR = \mathbb{Z}_q[X]/f(X)$
- Given $\mathbf{A} \in R^{n \times m}$, find "short" $\mathbf{x} \in R^m$ such that $\mathbf{Ax} = \mathbf{0} \pmod{q}$
- What is a "short" ring element $a(X) \in R$?
  - For power-of-two cyclotomic rings R, just take the norm of the coefficient vector $\|\mathbf{a}\|$

RingSIS

- Just like SIS, but using R instead of $\mathbb{Z}$
- Let $R_q = R/qR = \mathbb{Z}_q[X]/f(X)$
- Given $\mathbf{A} \in R^{n \times m}$, find "short" $\mathbf{x} \in R^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{0} \pmod q$
- What is a "short" ring element $a(X) \in R$?
  - For power-of-two cyclotomic rings R, just take the norm of the coefficient vector $\|\mathbf{a}\|$
- Effective dimension $\mathbf{A} \in \mathbb{Z}_q^{nd \times md}$
  - For large d, it is enough to take $n = 1$ and $m = 2n \log_2 q = 2 \log_2 q$
  - For $d = 256$, $q = 2^{16}$, now $\mathbf{A}$ only takes $64d$ $\mathtt{B} = 16$ $\mathtt{KB}$, instead of $64d^2$ $\mathtt{B} = 4$ $\mathtt{MB}$.

RingLWE

- Similar, like LWE but using R instead of $\mathbb{Z}$
- Let $R_q = R/qR = \mathbb{Z}_q[X]/f(X)$
- Distribution $[\mathbf{A}, \mathbf{As} + \mathbf{e}]$ where $\mathbf{A} \leftarrow R_q^{m \times k}$, $\mathbf{s} \leftarrow R_q^k$, and $\mathbf{e} \leftarrow \chi^{dm}$
  - Search RingLWE: Find $\mathbf{s}$ and $\mathbf{e} = \mathbf{b} - \mathbf{As}$
  - Decisional RingLWE: distinguish $[\mathbf{A}, \mathbf{b}]$ from uniformly random $R_q^{m \times (k+1)}$
  - Security level is the effective secret dimension dn

RingLWE

- Similar, like LWE but using R instead of $\mathbb{Z}$
- Let $R_q = R/qR = \mathbb{Z}_q[X]/f(X)$
- Distribution $[\mathbf{A}, \mathbf{As} + \mathbf{e}]$ where $\mathbf{A} \leftarrow R_q^{m \times k}$, $\mathbf{s} \leftarrow R_q^k$, and $\mathbf{e} \leftarrow \chi^{dm}$
  - Search RingLWE: Find $\mathbf{s}$ and $\mathbf{e} = \mathbf{b} - \mathbf{As}$
  - Decisional RingLWE: distinguish $[\mathbf{A}, \mathbf{b}]$ from uniformly random $R_q^{m \times (k+1)}$
  - Security level is the effective secret dimension dn
- Can set $k = 1$: distinguish $[\mathbf{a}, \mathbf{a} \cdot s + \mathbf{e}_i]$ from uniform $R_q^{m \times 2}$
  - The secret $s \in R_q$ is just a ring element
  - d serves as a security parameter
- When $k > 1$, this is sometime called ModuleLWE.

Running time

- Using $a(X)$ instead of $M_{a(x)}$ reduces storage (for keys, ciphertexts, etc.) from $O(d^2)$ to $O(d)$.
- What about running time?

Running time

- Using $a(X)$ instead of $M_{a(x)}$ reduces storage (for keys, ciphertexts, etc.) from $O(d^2)$ to $O(d)$.
- What about running time?
  - Computing the product $a(X) \cdot b(X)$ using the naive algorithm still takes time $O(d^2)$
  - This can be reduced to $O(d \log d)$

Running time

- Using $a(X)$ instead of $M_{a(x)}$ reduces storage (for keys, ciphertexts, etc.) from $O(d^2)$ to $O(d)$.
- What about running time?
  - Computing the product $a(X) \cdot b(X)$ using the naive algorithm still takes time $O(d^2)$
  - This can be reduced to $O(d \log d)$
- Approach:
  - Convert ring elements $a(X) \in R$ between their "coefficient representation $[a_0, \ldots, a_{d-1}]$ and their evaluation representation

$$(a(\omega_0), \ldots, a(\omega_{d-1}))$$

  - Note that in the evaluation representation, multiplication is componentwise

$$(a \cdot b)(\omega_i) = a(\omega_i) \cdot b(\omega_i)$$

    and can be computed with d scalar multiplications

FFT based multiplication

- The polynomial $f(X)$ has d complex roots $\omega_j = \exp((2j+1)\pi\imath/n)$

FFT based multiplication

- The polynomial $f(X)$ has d complex roots $\omega_j = \exp((2j+1)\pi\imath/n)$
- Using these $\omega_0, \ldots, \omega_{d-1}$ as evaluation points has two advantages
  1. Polynomial multiplication happens modulo $f(X)$ because $f(\omega_i) = 0$
  2. The mapping between the coefficient and evaluation representation can be computed in time $O(d \log d)$ using the Fast Fourier Transform (FFT) algorithm

FFT based multiplication

- The polynomial $f(X)$ has d complex roots $\omega_j = \exp((2j+1)\pi\imath/n)$
- Using these $\omega_0, \ldots, \omega_{d-1}$ as evaluation points has two advantages
  1. Polynomial multiplication happens modulo $f(X)$ because $f(\omega_i) = 0$
  2. The mapping between the coefficient and evaluation representation can be computed in time $O(d \log d)$ using the Fast Fourier Transform (FFT) algorithm
- Compute the polynomial product $a(X) \cdot b(X) \pmod{f(X)}$ as

$$a(X) \cdot b(X) = \mathbf{fft}^{-1}(\mathbf{fft}(a) \circ \mathbf{fft}(b))$$

  where $\circ$ is componentwise multiplication.
- Total running time is $2 \cdot O(d \log d) + O(d) = O(d \log d)$ arithmetic operations on *complex numbers*

The Number Theoretic Transform (NTT)

- We want to multiply polynomials in $R_q = \mathbb{Z}_q[X]/f(X)$
- Choose prime q such that $q \equiv 1 \pmod{2d}$
  - Since $2d$ divides $q - 1 = |\mathbb{Z}_q^*|$, there is an element $\omega \in \mathbb{Z}_q^*$ of order $2d$
  - The values $\omega_j = \omega^{2j+1} \in \mathbb{Z}_q$ are roots of $f(\omega_j) = 0 \pmod{q}$
  - We can perform the FFT computation in $\mathbb{Z}_q$, using ω
- This is called the Number Theoretic Transform

$$\textbf{ntt}: \mathbb{Z}_q^d \to \mathbb{Z}_q^d$$

- The effective use of NTT to efficiently implement (Ring) lattice cryptography first introduced and popularized by the SWIFFT hash function (Lyubashevsky,M.,Peikert,Rosen, 2008)

Applications

- All cryptographic applications of SIS and LWE are immediately adapted to the RingSIS/RingLWE setting: just use R instead of $\mathbb{Z}$

Applications

- All cryptographic applications of SIS and LWE are immediately adapted to the RingSIS/RingLWE setting: just use R instead of $\mathbb{Z}$
- How do you build lattice gadgets in the ring setting? e.g., how do you adapt $\mathbf{g} = (1, 2, 4, \ldots) \in \mathbb{Z}^w$ to R?

Applications

- All cryptographic applications of SIS and LWE are immediately adapted to the RingSIS/RingLWE setting: just use R instead of $\mathbb{Z}$
- How do you build lattice gadgets in the ring setting? e.g., how do you adapt $\mathbf{g} = (1, 2, 4, \ldots) \in \mathbb{Z}^w$ to R?
  - Simply note that $\mathbb{Z} \subset R$, so any $\mathbf{g} \in \mathbb{Z}^w$ is also a vector $\mathbf{g} \in R^w$
  - Scalars $a(X) = a_0 \in \mathbb{Z}$ are mapped to diagonal matrices $M_a = a_0 \cdot \mathbf{I}$
  - For example, the powers-of-two gadget becomes

$$\mathbf{G} = [\mathbf{I}, 2\mathbf{I}, 4\mathbf{I}, \ldots]^t$$

  - All gadget algorithms (inversion, decomposition) work on ring elements componentwise, on the coefficients of the input polynomial $a(X) = [a_0, \ldots, a_{d-1}]$

Applications

- All cryptographic applications of SIS and LWE are immediately adapted to the RingSIS/RingLWE setting: just use R instead of $\mathbb{Z}$
- How do you build lattice gadgets in the ring setting? e.g., how do you adapt $\mathbf{g} = (1, 2, 4, \ldots) \in \mathbb{Z}^w$ to R?
  - Simply note that $\mathbb{Z} \subset R$, so any $\mathbf{g} \in \mathbb{Z}^w$ is also a vector $\mathbf{g} \in R^w$
  - Scalars $a(X) = a_0 \in \mathbb{Z}$ are mapped to diagonal matrices $M_a = a_0 \cdot \mathbf{I}$
  - For example, the powers-of-two gadget becomes

$$\mathbf{G} = [\mathbf{I}, 2\mathbf{I}, 4\mathbf{I}, \ldots]^t$$

  - All gadget algorithms (inversion, decomposition) work on ring elements componentwise, on the coefficients of the input polynomial $a(X) = [a_0, \ldots, a_{d-1}]$
- Error is measured in the coefficient representation:
  - Gadget operations should always be performed on the coefficient representation
  - Can use $\mathbf{ntt}$ and $\mathbf{ntt}^{-1}$ to convert between coefficient and evaluation representation

Coefficient vs Evaluation representation

- Ring elements can be equivalently represented as
  - coefficient vectors $[a_0, \ldots, a_{d-1}] \in \mathbb{Z}_q^d$, or
  - evaluation vectors $[a(\omega_0), \ldots, a(\omega_{d-1})] \in \mathbb{Z}_q^d$
- Both representations are equally compact, but
  - the coefficient representation supports the evaluation of gadget operations
  - the evaluation representation allows to perform ring arithmetics (additions and multiplications) in linear time $O(d)$
- One can convert between the two representations in time $O(d \log d)$ using $\mathbf{ntt}, \mathbf{ntt}^{-1}$, and this is often the bottleneck in practical implementations of lattice cryptography

Kyber PKE – Parameters

- Core component of ML-KEM
- Parameters:
  - Ring dimension $d = 2^8 = 256$
  - 12-bit prime modulus $q = 3329 = 13 \cdot 256 + 1$
  - secret dimension: $k = 2, 3, 4$
- Technically, a full NTT would have required $2d = 512$ to divide $q - 1$, but 256 is good enough for efficient implementation

Kyber PKE – Parameters

- Core component of ML-KEM
- Parameters:
  - Ring dimension $d = 2^8 = 256$
  - 12-bit prime modulus $q = 3329 = 13 \cdot 256 + 1$
  - secret dimension: $k = 2, 3, 4$
- Technically, a full NTT would have required $2d = 512$ to divide $q - 1$, but 256 is good enough for efficient implementation
- Ring $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$
- Storage (per ring element): $d \cdot \log_2 q = 256 \cdot 12 = 3072$ bits
- Security level: $k \cdot d = 512, 768, 1024$

Kyber PKE – Algorithms

- Same as [LP'11] in the ring setting, but precompute **ntt** when useful/possible

Kyber PKE – Algorithms

- Same as [LP'11] in the ring setting, but precompute **ntt** when useful/possible
- Key generation: $\texttt{Gen} = (\hat{\mathbf{s}}, [\hat{\mathbf{A}}, \hat{\mathbf{b}}])$
  - $\hat{\mathbf{A}} = \mathbf{ntt}(\mathbf{A}) \in R_q^{k \times k}$, generated from a short seed $\rho \in \{0,1\}^{256}$
  - $\mathbf{s}, \mathbf{e} \in R_q^k$, chosen from distribution χ_1 on small elements
  - $\hat{\mathbf{s}} = \mathbf{ntt}(\mathbf{s})$, $\hat{\mathbf{e}} = \mathbf{ntt}(\mathbf{e})$
  - $\hat{\mathbf{b}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}} \in R_q^k$

Kyber PKE – Algorithms

- Same as [LP'11] in the ring setting, but precompute **ntt** when useful/possible
- Key generation: $\mathtt{Gen} = (\hat{\mathbf{s}}, [\hat{\mathbf{A}}, \hat{\mathbf{b}}])$
  - $\hat{\mathbf{A}} = \mathbf{ntt}(\mathbf{A}) \in R_q^{k \times k}$, generated from a short seed $\rho \in \{0,1\}^{256}$
  - $\mathbf{s}, \mathbf{e} \in R_q^k$, chosen from distribution χ_1 on small elements
  - $\hat{\mathbf{s}} = \mathbf{ntt}(\mathbf{s})$, $\hat{\mathbf{e}} = \mathbf{ntt}(\mathbf{e})$
  - $\hat{\mathbf{b}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}} \in R_q^k$
- Encryption $\mathtt{Enc}_{[\hat{\mathbf{A}}, \hat{\mathbf{b}}]}(m) = (\mathbf{u}, v)$:
  - $\mathbf{r} \leftarrow \chi_1^k$, $\mathbf{e} \leftarrow \chi_2^{k+1}$
  - $$\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} = \mathbf{ntt}^{-1}\left(\begin{bmatrix} \hat{\mathbf{A}}^t \\ \hat{\mathbf{b}}^t \end{bmatrix} \circ \mathbf{ntt}(\mathbf{r}) \right) + \mathbf{e} + \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{m} \end{bmatrix}$$

Kyber PKE – Algorithms

- Same as [LP'11] in the ring setting, but precompute **ntt** when useful/possible
- Key generation: $\mathtt{Gen} = (\hat{\mathbf{s}}, [\hat{\mathbf{A}}, \hat{\mathbf{b}}])$
  - $\hat{\mathbf{A}} = \mathbf{ntt}(\mathbf{A}) \in R_q^{k \times k}$, generated from a short seed $\rho \in \{0,1\}^{256}$
  - $\mathbf{s}, \mathbf{e} \in R_q^k$, chosen from distribution χ_1 on small elements
  - $\hat{\mathbf{s}} = \mathbf{ntt}(\mathbf{s})$, $\hat{\mathbf{e}} = \mathbf{ntt}(\mathbf{e})$
  - $\hat{\mathbf{b}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}} \in R_q^k$
- Encryption $\mathtt{Enc}_{[\hat{\mathbf{A}}, \hat{\mathbf{b}}]}(m) = (\mathbf{u}, v)$:
  - $\mathbf{r} \leftarrow \chi_1^k$, $\mathbf{e} \leftarrow \chi_2^{k+1}$
  - $\begin{bmatrix} \mathbf{u} \\ v \end{bmatrix} = \mathbf{ntt}^{-1}\left(\begin{bmatrix} \hat{\mathbf{A}}^t \\ \hat{\mathbf{b}}^t \end{bmatrix} \circ \mathbf{ntt}(\mathbf{r}) \right) + \mathbf{e} + \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{m} \end{bmatrix}$
- Decryption: $\mathtt{Dec}_{\hat{\mathbf{s}}}(\mathbf{u}, v) = \left\lceil (v - \mathbf{ntt}^{-1}(\hat{\mathbf{s}} \circ \mathbf{ntt}(\mathbf{u})))/\Delta \right\rfloor$

Conclusion

- Lattice cryptography: mature area building on +20 years of research
- Algebraic lattices: key to efficient implementation
- Tools and techniques: linear algebra, gadget lattices
- Natural homomorphic properties: rich set of applications
- Ready to use Public Key Encryption (and Digital Signatures)
- Any Questions?