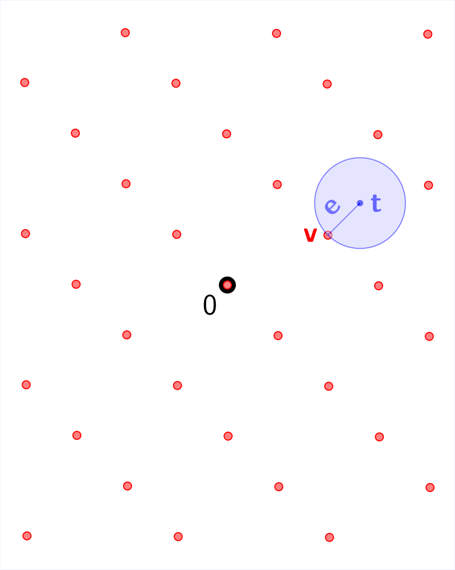
The SIS Problem and Cryptographic Applications

Daniele Micciancio

January 2020

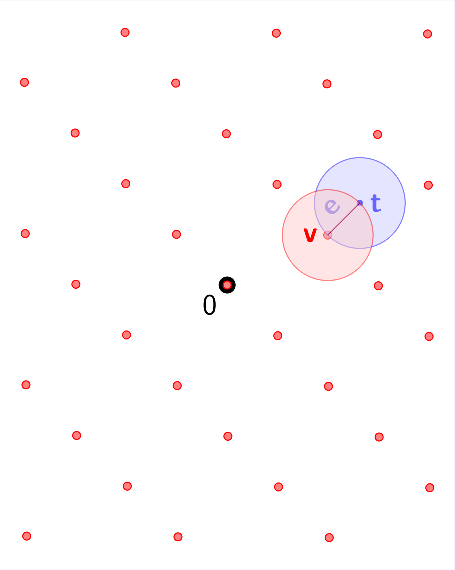
Outline

CVP and dual lattice



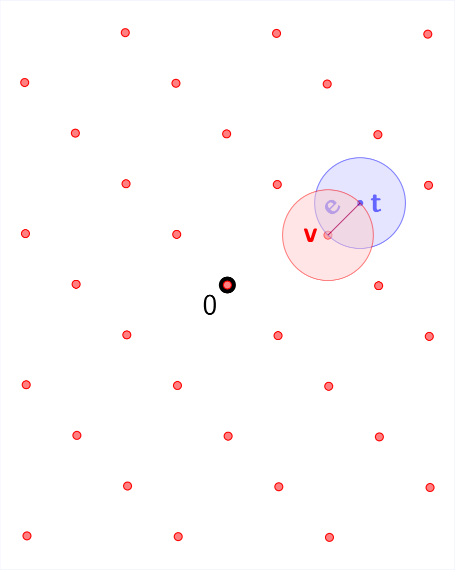
- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$

CVP and dual lattice



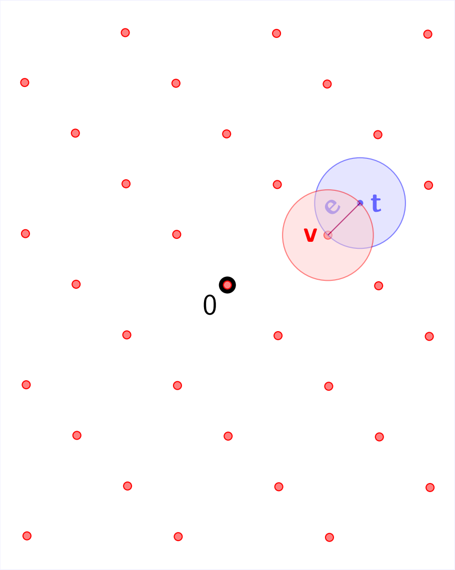
- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$

CVP and dual lattice



- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$
- Dual lattice $\Lambda^* = \mathcal{L}(\mathbf{D})$.

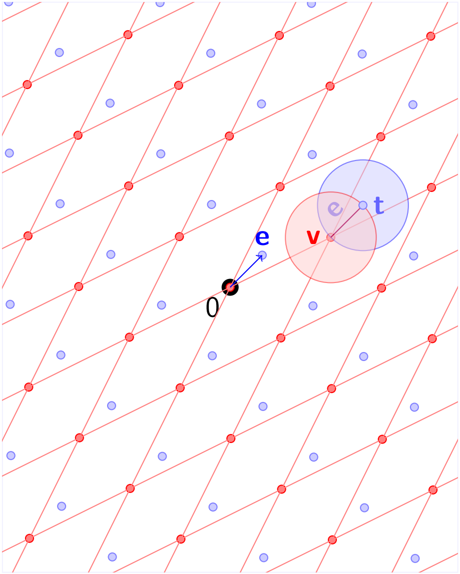
CVP and dual lattice



- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$
- Dual lattice $\Lambda^* = \mathcal{L}(\mathbf{D})$.
- Syndrome of $\mathbf{t}$:

$$\begin{aligned} \mathbf{s} &= \langle \mathbf{D}, \mathbf{t} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{v} \rangle + \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1. \end{aligned}$$

CVP and dual lattice

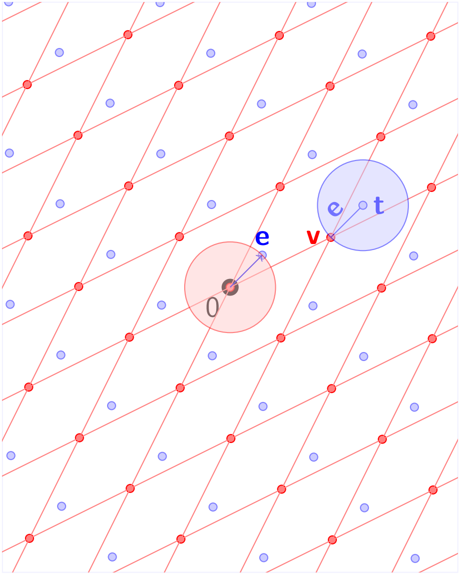


- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$
- Dual lattice $\Lambda^* = \mathcal{L}(\mathbf{D})$.
- Syndrome of $\mathbf{t}$:

$$\begin{aligned} \mathbf{s} &= \langle \mathbf{D}, \mathbf{t} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{v} \rangle + \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1. \end{aligned}$$

- $\mathbf{e}$ belongs to coset $\mathbf{t} + \Lambda = \{\mathbf{x} : \langle \mathbf{D}, \mathbf{x} \rangle = \mathbf{s} \bmod 1\}$

CVP and dual lattice



- Lattice Λ, target $\mathbf{t} = \mathbf{v} + \mathbf{e}$
- Dual lattice $\Lambda^* = \mathcal{L}(\mathbf{D})$.
- Syndrome of $\mathbf{t}$:

$$\begin{aligned} \mathbf{s} &= \langle \mathbf{D}, \mathbf{t} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{v} \rangle + \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1 \\ &= \langle \mathbf{D}, \mathbf{e} \rangle \bmod 1. \end{aligned}$$

- $\mathbf{e}$ belongs to coset $\mathbf{t} + \Lambda = \{\mathbf{x} : \langle \mathbf{D}, \mathbf{x} \rangle = \mathbf{s} \bmod 1\}$

Problem (Syndrome Decoding)

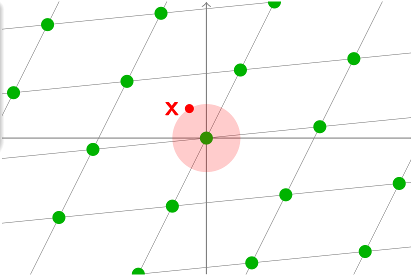
Find shortest $\mathbf{e}$ *such that* $\langle \mathbf{D}, \mathbf{e} \rangle = \mathbf{s} \bmod 1$

SIS/LWE as CVP

Candidate OWF

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

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



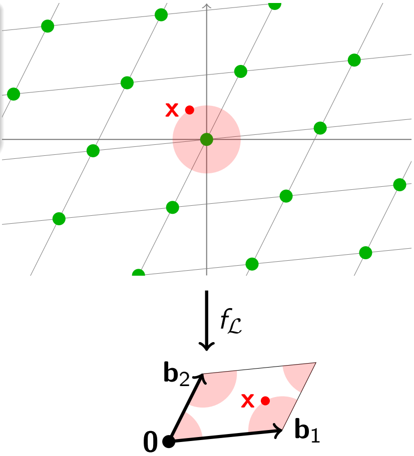
SIS/LWE as CVP

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

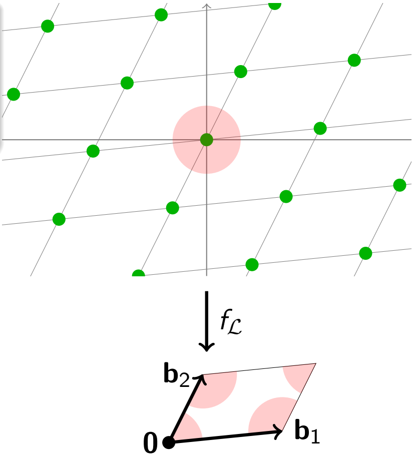
Candidate OWF

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

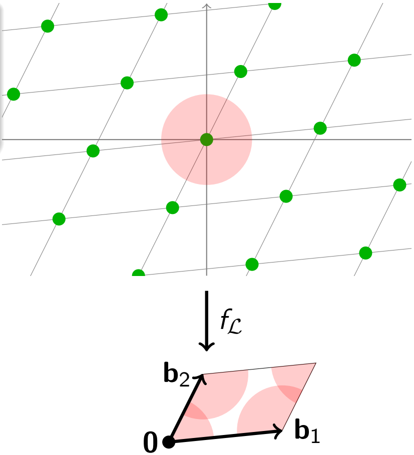


SIS/LWE as CVP

Candidate OWF

Key: a hard lattice $\mathcal{L}$
Input: $\mathbf{x}$, $\|\mathbf{x}\| \le \beta$
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

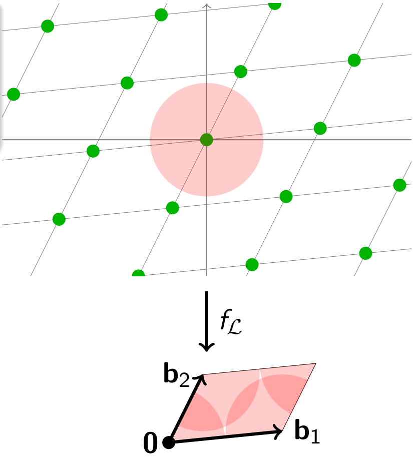


SIS/LWE as CVP

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

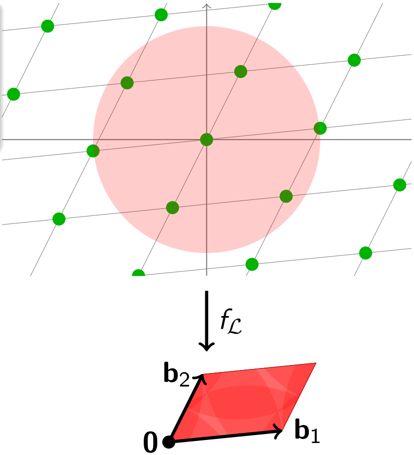


SIS/LWE as CVP

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

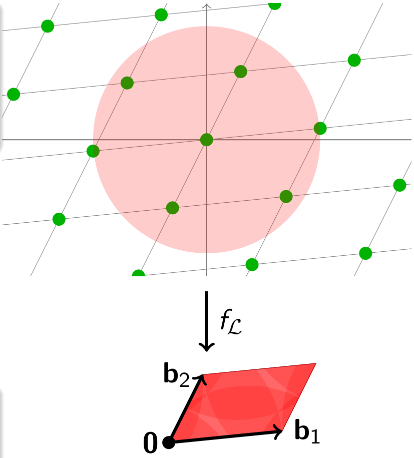
Candidate OWF

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

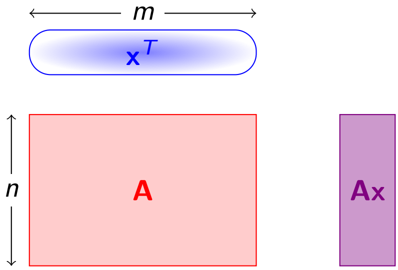
Question

Are these functions cryptographically hard to invert?



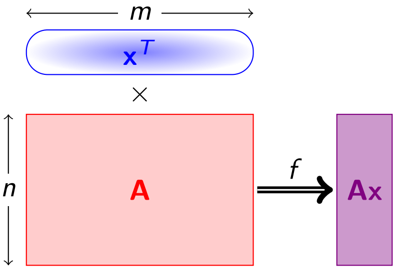
Ajtai's one-way function (SIS)

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



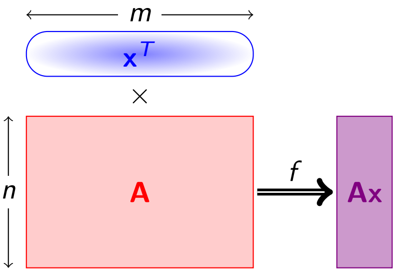
Ajtai's one-way function (SIS)

- 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] . . .

Cryptographic functions

Definition (Ajtai's function)

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

$\mathbf{x} \in \{0,1\}^m$: $\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$ $\qquad (q = 10)$

$\longleftarrow$ m $\longrightarrow$

$$\mathbf{A} \in \mathbb{Z}_q^{n\times m}: \begin{bmatrix} 1 & 4 & 5 & 9 & 3 & 0 & 2 \\ 4 & 2 & 8 & 6 & 2 & 4 & 3 \\ 7 & 5 & 5 & 4 & 7 & 8 & 0 \\ 2 & 7 & 0 & 1 & 4 & 6 & 9 \end{bmatrix} \; \updownarrow n \qquad \begin{bmatrix} 2 \\ 2 \\ 7 \\ 1 \end{bmatrix} \; \mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$$

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find $\mathbf{x} \in \{0,1\}^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{y}$

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{A}\mathbf{x} = \mathbf{y} \pmod{q}$

Inverting Ajtai's function can be formulated as a lattice problem.

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{Ax} = \mathbf{y} \pmod q$

Inverting Ajtai's function can be formulated as a lattice problem.

- Easy problem: find (arbitrary) integer solution $\mathbf{t}$ to system of linear equations $\mathbf{At} = \mathbf{y} \pmod q$

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{Ax} = \mathbf{y} \pmod q$

Inverting Ajtai's function can be formulated as a lattice problem.

- Easy problem: find (arbitrary) integer solution $\mathbf{t}$ to system of linear equations $\mathbf{At} = \mathbf{y} \pmod q$
- All solutions to $\mathbf{Ax} = \mathbf{y}$ are of the form $\mathbf{t} + \Lambda^\perp$ where

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \colon \mathbf{Ax} = \mathbf{0} \pmod q\}$$

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{A}\mathbf{x} = \mathbf{y} \pmod q$

Inverting Ajtai's function can be formulated as a lattice problem.

- Easy problem: find (arbitrary) integer solution $\mathbf{t}$ to system of linear equations $\mathbf{A}\mathbf{t} = \mathbf{y} \pmod q$
- All solutions to $\mathbf{A}\mathbf{x} = \mathbf{y}$ are of the form $\mathbf{t} + \Lambda^\perp$ where

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \colon \mathbf{A}\mathbf{x} = \mathbf{0} \pmod q\}$$

- Cryptanalysis problem: find a small vector in $\mathbf{t} + \Lambda^\perp(\mathbf{A})$

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{Ax} = \mathbf{y} \pmod{q}$

Inverting Ajtai's function can be formulated as a lattice problem.

- Easy problem: find (arbitrary) integer solution $\mathbf{t}$ to system of linear equations $\mathbf{At} = \mathbf{y} \pmod{q}$
- All solutions to $\mathbf{Ax} = \mathbf{y}$ are of the form $\mathbf{t} + \Lambda^\perp$ where

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \colon \mathbf{Ax} = \mathbf{0} \pmod{q}\}$$

- Cryptanalysis problem: find a small vector in $\mathbf{t} + \Lambda^\perp(\mathbf{A})$
- Equivalently: find a lattice vector $\mathbf{v} \in \Lambda^\perp(\mathbf{A})$ close to $\mathbf{t}$

Ajtai's function and lattice problems

Cryptanalysis (Inversion)

Given $\mathbf{A}$ and $\mathbf{y}$, find small solution $\mathbf{x} \in \{0,1\}^m$ to inhomogeneous linear system $\mathbf{A}\mathbf{x} = \mathbf{y} \pmod q$

Inverting Ajtai's function can be formulated as a lattice problem.

- Easy problem: find (arbitrary) integer solution $\mathbf{t}$ to system of linear equations $\mathbf{A}\mathbf{t} = \mathbf{y} \pmod q$
- All solutions to $\mathbf{A}\mathbf{x} = \mathbf{y}$ are of the form $\mathbf{t} + \Lambda^\perp$ where

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m \colon \mathbf{A}\mathbf{x} = \mathbf{0} \pmod q\}$$

- Cryptanalysis problem: find a small vector in $\mathbf{t} + \Lambda^\perp(\mathbf{A})$
- Equivalently: find a lattice vector $\mathbf{v} \in \Lambda^\perp(\mathbf{A})$ close to $\mathbf{t}$

Inverting Ajtai's function is an average case instance of the Closest Vector Problem where the lattice is chosen according to $\Lambda^\perp(\mathbf{A})$

Ajtai's function: collision resistance

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

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

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

$$\mathbf{z} = \quad \mathbf{x} - \quad \mathbf{y}$$

Ajtai's function: collision resistance

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

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

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

$$\mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{y} = \mathbf{0} \bmod q$$

Ajtai's function: collision resistance

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

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

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

$$\mathbf{Az} = \mathbf{Ax} - \mathbf{Ay} = \mathbf{0} \bmod q$$

- Collisions are lattice vectors $\mathbf{z} \in \Lambda^\perp(\mathbf{A})$ with small norm $\|\mathbf{z}\|_\infty = \max_i |z_i| = 1$.

Ajtai's function: collision resistance

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

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

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

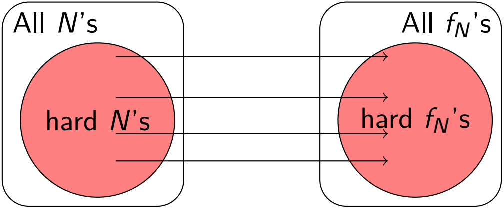
$$\mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{y} = \mathbf{0} \bmod q$$

- Collisions are lattice vectors $\mathbf{z} \in \Lambda^\perp(\mathbf{A})$ with small norm $\|\mathbf{z}\|_\infty = \max_i |z_i| = 1$.
- ... there is a much deeper and interesting relation between breaking $f_{\mathbf{A}}$ and lattice problems.

Provable security (from average case hardness)

Example 1: (Rabin) modular squaring

- $f_N(x) = x^2 \bmod N$, where $N = p \cdot q$
- Inverting f_N is at least as hard as factoring N



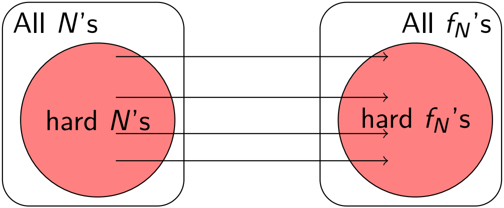
Provable security (from average case hardness)

Example 1: (Rabin) modular squaring

- $f_N(x) = x^2 \bmod N$, where $N = p \cdot q$
- Inverting f_N is at least as hard as factoring N

Theorem

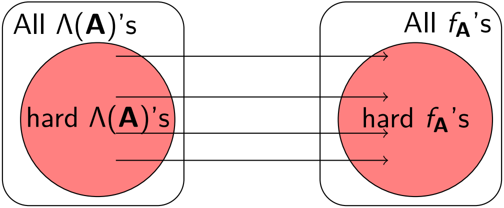
f_N is cryptographically hard to invert, provided most $N = p \cdot q$ are hard to factor



Provable security (from average case hardness)

Example 2: Ajtai's function

- $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$
- Finding collisions in $f_{\mathbf{A}}$ is as hard as ℓ_∞-SVP in $\Lambda(\mathbf{A})$



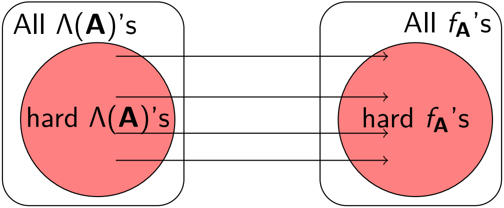
Provable security (from average case hardness)

Example 2: Ajtai's function

- $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$
- Finding collisions in $f_{\mathbf{A}}$ is as hard as ℓ_∞-SVP in $\Lambda(\mathbf{A})$

Theorem

$f_{\mathbf{A}}$ is collision resistant, provided ℓ_∞-SVP is hard for most lattices $\Lambda(\mathbf{A})$



Average-case Complexity

Average-case complexity depends on input distribution

Example (Factoring problem)

Given a number N, output $a, b > 1$ such that $N = ab$

Average-case Complexity

Average-case complexity depends on input distribution

Example (Factoring problem)

Given a number N, output $a, b > 1$ such that $N = ab$

Factoring can be easy on average

if N is uniformly random, then $N = 2 \cdot \frac{N}{2}$ with probability 50%!

Average-case Complexity

Average-case complexity depends on input distribution

Example (Factoring problem)

Given a number N, output $a, b > 1$ such that $N = ab$

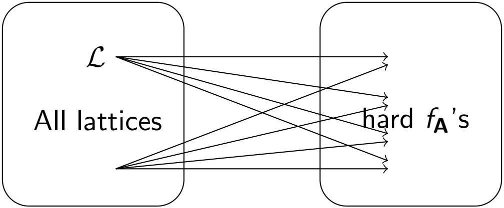
Factoring can be easy on average

if N is uniformly random, then $N = 2 \cdot \frac{N}{2}$ with probability 50%!

- Factoring $N = pq$ is believed to be hard when p, q are randomly chosen primes
- How do we know $\Lambda^{\perp}(\mathbf{A})$ is a hard distribution for SVP?

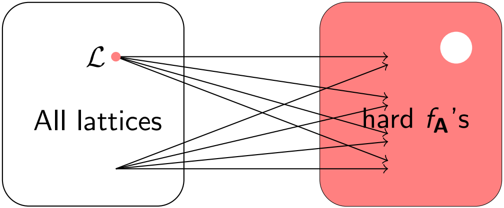
Provable security (from worst case hardness)

- Any fixed lattice $\mathcal{L}$ is mapped to a random $\mathbf{A}$
- Finding collisions in $f_{\mathbf{A}}$ allows to find (relatively) short vectors in $\mathcal{L}$.



Provable security (from worst case hardness)

- Any fixed lattice $\mathcal{L}$ is mapped to a random $\mathbf{A}$
- Finding collisions in $f_{\mathbf{A}}$ allows to find (relatively) short vectors in $\mathcal{L}$.

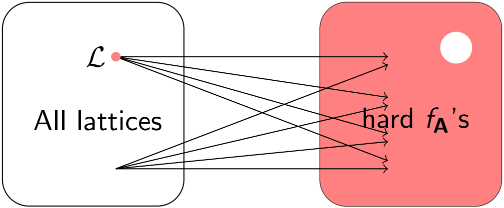


Provable security (from worst case hardness)

- Any fixed lattice $\mathcal{L}$ is mapped to a random $\mathbf{A}$
- Finding collisions in $f_{\mathbf{A}}$ allows to find (relatively) short vectors in $\mathcal{L}$.

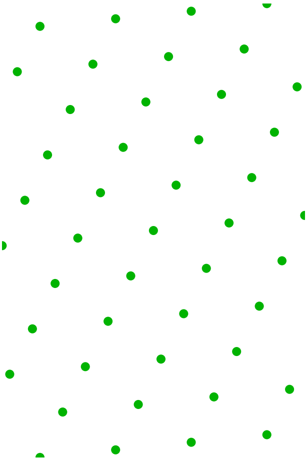
Theorem (Ajtai,...,Micciancio&Regev)

$f_{\mathbf{A}}$ is collision resistant, provided SIVP is hard to approximate (within $\gamma = n$) for some $\mathcal{L}$



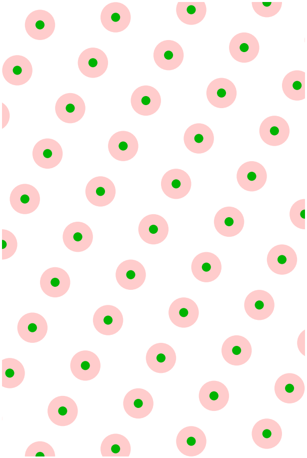
Blurring a lattice

Consider a lattice Λ, and



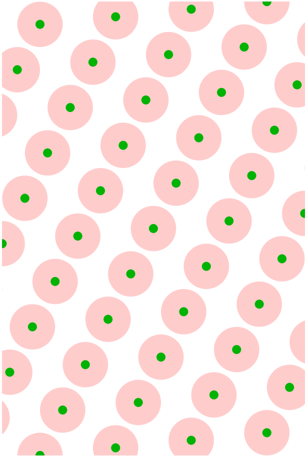
Blurring a lattice

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



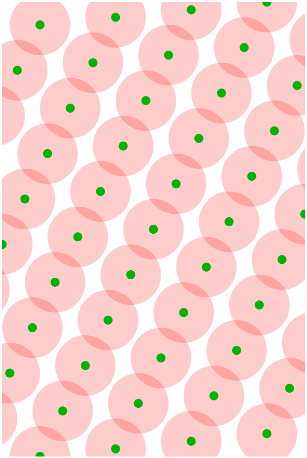
Blurring a lattice

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



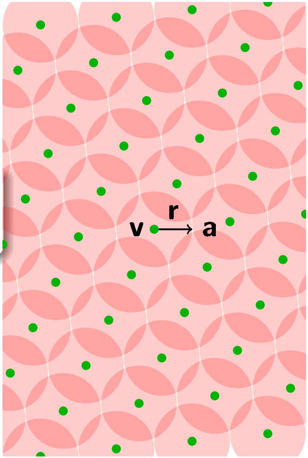
Blurring a lattice

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?

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

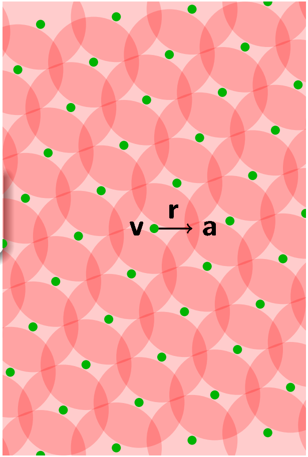
- Each point in $\mathbf{a} \in \mathbb{R}^n$ can be written $\mathbf{a} = \mathbf{v} + \mathbf{r}$ where $\mathbf{v} \in \mathcal{L}$ and $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.

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?

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



- 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 $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.

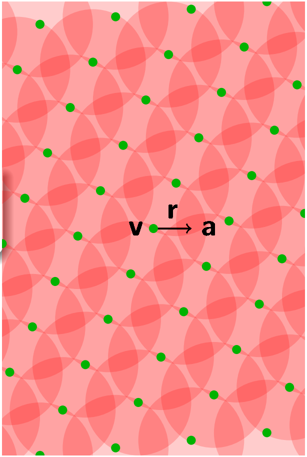
Blurring a lattice

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?

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

- 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 $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.

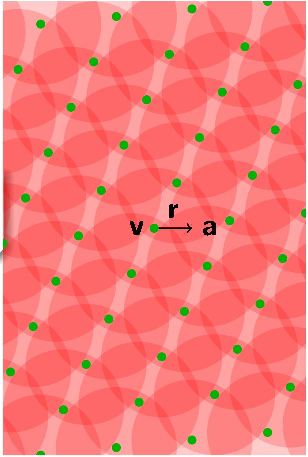


Blurring a lattice

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 $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.

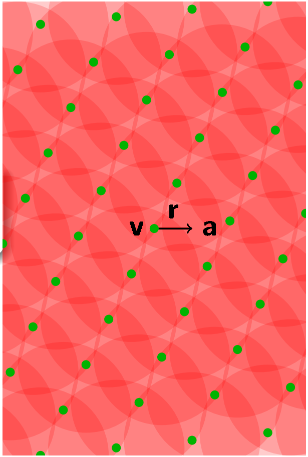
Blurring a lattice

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?

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

- 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 $\|\mathbf{r}\| \approx \sqrt{n}\lambda_n$.

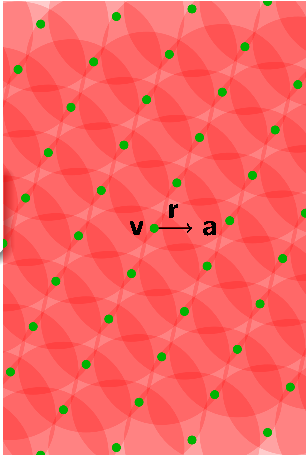


Blurring a lattice

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.

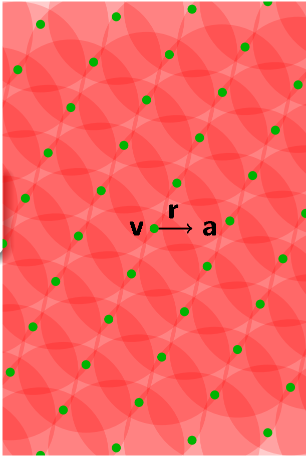
Blurring a lattice

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

1. The Short Integer Solution (SIS) Problem

2. Average Case Hardness

3. Efficiency and RingSIS
   - Small modulus
   - Ideal Lattices

4. Cryptographic Applications
   - 1: Compression and Hashing
   - 2: Regularity and Commitment Schemes
   - 3: Linearity and Digital Signatures

Ajtai's connection

Theorem (A'96)

For large enough m, n, q*, the function* $f_{\mathbf{A}}$ *is collision resistant*

Ajtai's connection

Theorem (A'96)

For large enough m, n, q, the function $f_{\mathbf{A}}$ is collision resistant

- Original proof required $q = n^{O(1)}$ to be a large polynomial
- Improved to $q \approx n^{2.5}$ in [MR'04]
- Further improved in [GPV'08] to $q \approx n$, making seemingly optimal use of known techniques
- Question: How can we prove hardness for smaller values of q?

Ajtai's connection

Theorem (A'96)

For large enough m, n, q, the function $f_{\mathbf{A}}$ is collision resistant

- Original proof required $q = n^{O(1)}$ to be a large polynomial
- Improved to $q \approx n^{2.5}$ in [MR'04]
- Further improved in [GPV'08] to $q \approx n$, making seemingly optimal use of known techniques
- Question: How can we prove hardness for smaller values of q?

Theorem (MP'13)

If one can break $f_{\mathbf{A}}$ for some $\sqrt{n} < q < n$, then one can also break it for larger $q' = q^c$, $c > 1$.

Reducing q in SIS (proof sketch, toy version)

- For simplicity, assume $f_{\mathbf{A}}$ takes binary inputs $\mathbf{x} \in \{0,1\}^m$.

Reducing q in SIS (proof sketch, toy version)

- For simplicity, assume $f_{\mathbf{A}}$ takes binary inputs $\mathbf{x} \in \{0,1\}^m$.
- Say we can solve SIS for some n, m, q. $\boxed{\mathbf{A}'(\mathbb{Z}_q^{n\times m})}$

Reducing q in SIS (proof sketch, toy version)

- For simplicity, assume $f_{\mathbf{A}}$ takes binary inputs $\mathbf{x} \in \{0,1\}^m$.
- Say we can solve SIS for some n, m, q. $\boxed{\mathbf{A}'(\mathbb{Z}_q^{n\times m})}$
- We solve SIS with parameters n, m^2, q^2 as follows:

$$\boxed{\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})}$$

Reducing q in SIS (proof sketch, toy version)

- For simplicity, assume $f_{\mathbf{A}}$ takes binary inputs $\mathbf{x} \in \{0,1\}^m$.
- Say we can solve SIS for some n, m, q. $\mathbf{A}'(\mathbb{Z}_q^{n\times m})$
- We solve SIS with parameters n, m^2, q^2 as follows:

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$

$\mathbf{A}_1$	$\mathbf{A}_2$	$\cdots$	$\mathbf{A}_m$

Reducing q in SIS (proof sketch, toy version)

- For simplicity, assume $f_{\mathbf{A}}$ takes binary inputs $\mathbf{x} \in \{0,1\}^m$.
- Say we can solve SIS for some n, m, q. $\boxed{\mathbf{A}'(\mathbb{Z}_q^{n\times m})}$
- We solve SIS with parameters n, m^2, q^2 as follows:

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$			
$\mathbf{A}_1$	$\mathbf{A}_2$	$\cdots$	$\mathbf{A}_m$
$\mathbf{A}'_1 + q\mathbf{A}''_1$	$\mathbf{A}'_2 + q\mathbf{A}''_2$	$\cdots$	$\mathbf{A}'_m + q\mathbf{A}''_m$

- $\mathbf{A}'_i, \mathbf{A}''_i \in \mathbb{Z}_q^{n\times m}$ for all i

Reducing q in SIS (toy version, cont.)

$$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$$

$$\mathbf{A}'_1 + q\mathbf{A}''_1 \quad \mathbf{A}'_2 + q\mathbf{A}''_2 \quad \cdots \quad \mathbf{A}'_m + q\mathbf{A}''_m$$

Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n \times m^2})$			
$\mathbf{A}'_1 + q\mathbf{A}''_1$	$\mathbf{A}'_2 + q\mathbf{A}''_2$	$\cdots$	$\mathbf{A}'_m + q\mathbf{A}''_m$

- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$

Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n \times m^2})$			
$\mathbf{A}'_1 + q\mathbf{A}''_1$	$\mathbf{A}'_2 + q\mathbf{A}''_2$	$\cdots$	$\mathbf{A}'_m + q\mathbf{A}''_m$

- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i$

Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$			
$\mathbf{A}'_1 + q\mathbf{A}''_1$	$\mathbf{A}'_2 + q\mathbf{A}''_2$	$\cdots$	$\mathbf{A}'_m + q\mathbf{A}''_m$

- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i = \frac{1}{q}(\mathbf{A}'_i\mathbf{z}_i) + \frac{q}{q}(\mathbf{A}''_i\mathbf{z}_i) \in \mathbb{Z}_q^n$

Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n \times m^2})$

$\mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_m$

- Find SIS(n,m,q) collisions $\mathbf{A}'_i \mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i = \frac{1}{q}(\mathbf{A}'_i \mathbf{z}_i) + \frac{q}{q}(\mathbf{A}''_i \mathbf{z}_i) \in \mathbb{Z}_q^n$

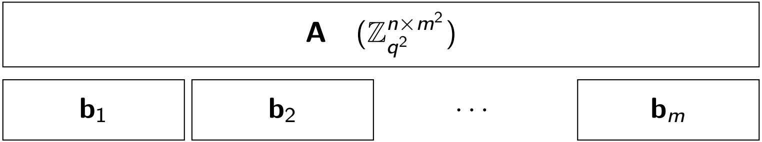
Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$

$\mathbf{b}_1$ | $\mathbf{b}_2$ | $\cdots$ | $\mathbf{b}_m$

- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i = \frac{1}{q}(\mathbf{A}'_i\mathbf{z}_i) + \frac{q}{q}(\mathbf{A}''_i\mathbf{z}_i) \in \mathbb{Z}_q^n$
- Solve SIS(n,m,q) instance $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$ to find collision $\mathbf{w}$

Reducing q in SIS (toy version, cont.)



- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i = \frac{1}{q}(\mathbf{A}'_i\mathbf{z}_i) + \frac{q}{q}(\mathbf{A}''_i\mathbf{z}_i) \in \mathbb{Z}_q^n$
- Solve SIS(n,m,q) instance $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$ to find collision $\mathbf{w}$
- Output collision $\mathbf{A}(\mathbf{w} \otimes \mathbf{z}_*) \equiv_{q^2} \mathbf{0}$

$$(\mathbf{w} \otimes \mathbf{z}_*) = (w_1 \cdot \mathbf{z}_1, \ldots, w_m \cdot \mathbf{z}_m) \in \{-1, 0, +1\}^{m^2}$$

Reducing q in SIS (toy version, cont.)

$\mathbf{A} \quad (\mathbb{Z}_{q^2}^{n\times m^2})$			
$\mathbf{b}_1$	$\mathbf{b}_2$	$\cdots$	$\mathbf{b}_m$

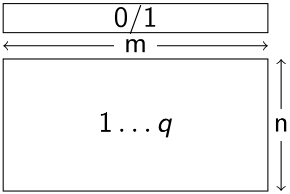
- Find SIS(n,m,q) collisions $\mathbf{A}'_i\mathbf{z}_i \equiv_q 0$, $\mathbf{z}_i \in \{0, \pm 1\}$
- Compute $\mathbf{b}_i = \frac{1}{q}(\mathbf{A}'_i + q\mathbf{A}''_i)\mathbf{z}_i = \frac{1}{q}(\mathbf{A}'_i\mathbf{z}_i) + \frac{q}{q}(\mathbf{A}''_i\mathbf{z}_i) \in \mathbb{Z}_q^n$
- Solve SIS(n,m,q) instance $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]$ to find collision $\mathbf{w}$
- Output collision $\mathbf{A}(\mathbf{w} \otimes \mathbf{z}_*) \equiv_{q^2} \mathbf{0}$

$$(\mathbf{w} \otimes \mathbf{z}_*) = (w_1 \cdot \mathbf{z}_1, \ldots, w_m \cdot \mathbf{z}_m) \in \{-1, 0, +1\}^{m^2}$$

- Actual proof used discrete gaussian sampling (DGS $\leq$ DGS)

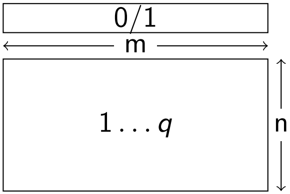
Efficiency of Ajtai's function

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



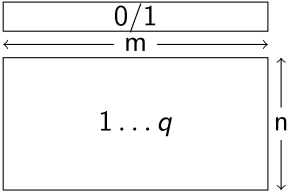
Efficiency of Ajtai's function

- $q = n^{O(1)}$, $m = O(n \log n) > n \log_2 q$
- 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

$$\mathbf{A} = [\mathbf{A}^{(1)} \mid \ldots \mid \mathbf{A}^{(m/n)}]$$

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

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

Efficient lattice based hashing

Idea

Use structured matrix

$$\mathbf{A} = [\mathbf{A}^{(1)} \mid \ldots \mid \mathbf{A}^{(m/n)}]$$

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

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

- Proposed by [M02], where it is proved that $f_{\mathbf{A}}$ is one-way under plausible complexity assumptions
- Similar idea first used by NTRU public key cryptosystem (1998), but with no proof of security
- Wishful thinking: finding short vectors in $\Lambda_q^{\perp}(\mathbf{A})$ is hard, and therefore $f_{\mathbf{A}}$ is collision resistant

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	9	7	3
6	9	7	3
6	9	7	3
6	9	7	3

- $x^n - 1 = (x-1) \cdot (x^{n-1} + \cdots + 1)$

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

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

- $x^n - 1 = (x-1) \cdot (x^{n-1} + \cdots + 1)$

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.

- $x^n + 1$ is irreducible (for $n = 2^k$)

Efficiency of anti-cyclic hashing

- Key size: $(m/n) \cdot n \log q = m \cdot \log q = \tilde{O}(n)$ bits
- Anti-cyclic matrix-vector multiplication can be computed in quasi-linear time $\tilde{O}(n)$ using FFT
- The resulting hash function can also be computed in $\tilde{O}(n)$ time
- For appropriate choice of parameters, this can be very practical (SWIFFT [LMPR])
- The hash function is linear: $\mathbf{A}(\mathbf{x}+\mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}$
- This can be a feature rather than a weakness

Ideal Lattices and Algebraic number theory

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

$$f_{\mathbf{a}_1,\ldots,\mathbf{a}_k}(\mathbf{u}_1, \ldots, \mathbf{u}_k) = \sum_i \mathbf{a}_i(X) \cdot \mathbf{u}_i(X) \pmod{X^n - 1}$$

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

1. The Short Integer Solution (SIS) Problem
2. Average Case Hardness
3. Efficiency and RingSIS
   - Small modulus
   - Ideal Lattices
4. Cryptographic Applications
   - 1: Compression and Hashing
   - 2: Regularity and Commitment Schemes
   - 3: Linearity and Digital Signatures

SIS: Properties and Applications

- Properties:
  1. Compression
  2. Regularity
  3. Homomorphism
- Applications:
  1. Collision Resistant Hashing
  2. Commitment Schemes
  3. Digital Signatures

SIS Property: Compression

SIS Function

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

Main security parameter: n. (Security largely independent of m.)

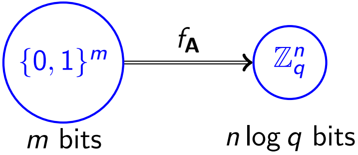
SIS Property: Compression

SIS Function

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

Main security parameter: n. (Security largely independent of m.)

- $f_{\mathbf{A}}$: m bits $\rightarrow$ $n \lg q$ bits.



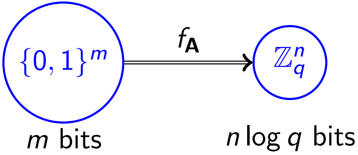
SIS Property: Compression

SIS Function

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

Main security parameter: n. (Security largely independent of m.)

- $f_{\mathbf{A}}$: m bits $\to n \lg q$ bits.
- When $(m > n \lg q)$, $f_{\mathbf{A}}$ is a compression function.



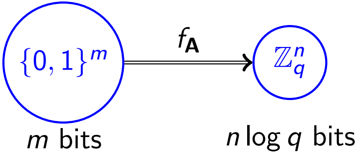
SIS Property: Compression

SIS Function

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

Main security parameter: n. (Security largely independent of m.)

- $f_{\mathbf{A}}$: m bits $\rightarrow n \lg q$ bits.
- When $(m > n \lg q)$, $f_{\mathbf{A}}$ is a compression function.
- E.g., $m = 2n \lg q$: $f_{\mathbf{A}} : \{0,1\}^m \rightarrow \{0,1\}^{m/2}$.



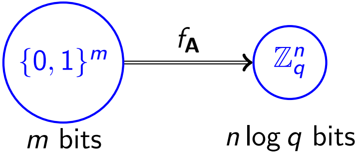
SIS Property: Compression

SIS Function

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

Main security parameter: n. (Security largely independent of m.)

- $f_{\mathbf{A}}$: m bits $\to n \lg q$ bits.
- When $(m > n \lg q)$, $f_{\mathbf{A}}$ is a compression function.
- E.g., $m = 2n \lg q$: $f_{\mathbf{A}} : \{0,1\}^m \to \{0,1\}^{m/2}$.



Ajtai's theorem requires $(m > n \lg q)$

Collision Resistant Hashing

Keyed function family $f_A \colon X \to Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A \colon Y^2 \to Y$.)

Collision Resistant Hashing

Keyed function family $f_A \colon X \to Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A \colon Y^2 \to Y$.)

Definition (Collision Resistance)

Finding $x_1 \neq x_2 \in X$ such that $f_A(x_1) = f_A(x_2)$ is hard.

Collision Resistant Hashing

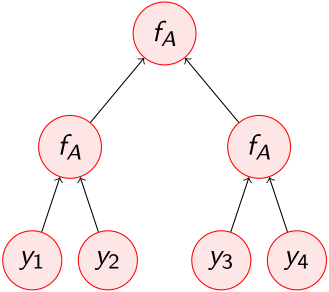
Keyed function family $f_A\colon X \to Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A\colon Y^2 \to Y$.)

Definition (Collision Resistance)

Finding $x_1 \neq x_2 \in X$ such that $f_A(x_1) = f_A(x_2)$ is hard.

Classic application: Merkle Trees

- Leaves are user data
- Each internal node is the hash of its children
- Root r commits to all $y_1, \ldots, y_n$
- Each y_i can be shown to be consistent with r by revealing $\log(n)$ values



SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A\colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}}\colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ *is one-way, then* $f_{\mathbf{A}}\colon \{0, 1\}^m \to \mathbb{Z}_q^n$ *is collision resistant.*

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}} \colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ *is one-way, then* $f_{\mathbf{A}} \colon \{0, 1\}^m \to \mathbb{Z}_q^n$ *is collision resistant.*

- Assume can find collisions to $f_{\mathbf{A}}$

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}} \colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ is one-way, then $f_{\mathbf{A}} \colon \{0, 1\}^m \to \mathbb{Z}_q^n$ is collision resistant.

- Assume can find collisions to $f_{\mathbf{A}}$
- Goal: Given random $\mathbf{A}$ and $\mathbf{y}$, find $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{y}$

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}} \colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ is one-way, then $f_{\mathbf{A}} \colon \{0, 1\}^m \to \mathbb{Z}_q^n$ is collision resistant.

- Assume can find collisions to $f_{\mathbf{A}}$
- Goal: Given random $\mathbf{A}$ and $\mathbf{y}$, find $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{y}$
- Add $\mathbf{y}$ to random column $\mathbf{a_i'} = \mathbf{a_i} + \mathbf{y}$.

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}} \colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ is one-way, then $f_{\mathbf{A}} \colon \{0, 1\}^m \to \mathbb{Z}_q^n$ is collision resistant.

- Assume can find collisions to $f_{\mathbf{A}}$
- Goal: Given random $\mathbf{A}$ and $\mathbf{y}$, find $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{y}$
- Add $\mathbf{y}$ to random column $\mathbf{a}'_{\mathbf{i}} = \mathbf{a}_{\mathbf{i}} + \mathbf{y}$.
- Find collision for $\mathbf{A}'$: $\mathbf{A}'\mathbf{x} = \mathbf{A}'\mathbf{x}'$

SIS Application: Collision Resistant Hashing

Definition (Collision Resistance)

$f_A \colon X \to Y$. No adversary, given a random A, can efficiently find $x \neq x' \in X$ such that $f_A(x) = f_A(x')$

Theorem

If $f_{\mathbf{A}} \colon \{0, \pm 1\}^m \to \mathbb{Z}_q^n$ is one-way, then $f_{\mathbf{A}} \colon \{0, 1\}^m \to \mathbb{Z}_q^n$ is collision resistant.

- Assume can find collisions to $f_{\mathbf{A}}$
- Goal: Given random $\mathbf{A}$ and $\mathbf{y}$, find $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{y}$
- Add $\mathbf{y}$ to random column $\mathbf{a_i'} = \mathbf{a_i} + \mathbf{y}$.
- Find collision for $\mathbf{A}'$: $\mathbf{A}'\mathbf{x} = \mathbf{A}'\mathbf{x}'$
- If $x_i' = 1$ and $x_i = 0$, then $\mathbf{A}(\mathbf{x} - \mathbf{x}') = \mathbf{y}$

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

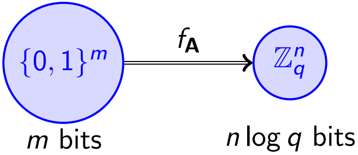
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

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

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

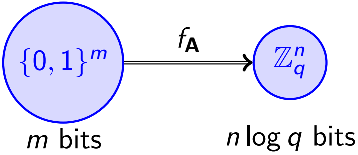
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

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

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 $\implies$ Regular

SIS Property: Regularity

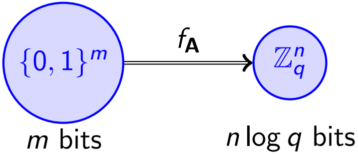
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

SIS Function

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

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.

Perfectly Hiding Commitments

Perfectly Hiding Commitments

- Analogy:
  - Lock message in a box
  - Give box, keep key
  - Later: give key to open box

Perfectly Hiding Commitments

- Analogy:
  - Lock message in a box
  - Give box, keep key
  - Later: give key to open box
- Implementation
  - Randomized function $C(m;r)$
  - Commit(m): give $c = C(m;r)$ for random $r \leftarrow \$$
  - Open: reveal m, r such that $C(m;r) = c$.

Perfectly Hiding Commitments

- Analogy:
  - Lock message in a box
  - Give box, keep key
  - Later: give key to open box
- Implementation
  - Randomized function $C(m;r)$
  - Commit(m): give $c = C(m;r)$ for random $r \leftarrow \$$
  - Open: reveal m, r such that $C(m;r) = c$.
- Security properties:
  - Hiding: $c = C(m;\$)$ is independent of m
  - Binding: hard to find $m \neq m'$ and r, r' such that $C(m;r) = C(m';r')$.

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: $C(\mathbf{m}, \mathbf{r}) = f_{[\mathbf{A_1},\mathbf{A_2}]}(\mathbf{m}, \mathbf{r}) = \mathbf{A_1}\mathbf{m} + \mathbf{A_2}\mathbf{r}$.

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

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

- 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

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

- 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

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

- 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:

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

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$
  2. Adversary $m \leftarrow Adv(pk)$ chooses message query

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$
  2. Adversary $m \leftarrow Adv(pk)$ chooses message query
  3. ... receives signature $\sigma \leftarrow Sign(s, m)$,

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$
  2. Adversary $m \leftarrow Adv(pk)$ chooses message query
  3. ...receives signature $\sigma \leftarrow Sign(s, m)$,
  4. ...and outputs forgery $(m', \sigma') \leftarrow Adv(\sigma)$.

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$
  2. Adversary $m \leftarrow Adv(pk)$ chooses message query
  3. ... receives signature $\sigma \leftarrow Sign(s, m)$,
  4. ... and outputs forgery $(m', \sigma') \leftarrow Adv(\sigma)$.
  5. Adversary wins if $Verify(m', \sigma')$ and $m \neq m'$.

(One-Time) Digital Signatures

- Digital Signature Scheme:
  - Key Generation Algorithm: $(pk, sk) \leftarrow KeyGen$
  - Signing Algorithm: $Sign(sk, m) = \sigma$
  - Verification Algorithm: $Verify(pk, m, \sigma)$
- (One-Time) Security:
  1. Generate keys $(pk, sk) \leftarrow KeyGen$
  2. Adversary $m \leftarrow Adv(pk)$ chooses message query
  3. . . . receives signature $\sigma \leftarrow Sign(s, m)$,
  4. . . . and outputs forgery $(m', \sigma') \leftarrow Adv(\sigma)$.
  5. Adversary wins if $Verify(m', \sigma')$ and $m \neq m'$.
- General Signatures: Adversary is allowed an arbitrary number of signature queries

SIS Application: One-Time Signatures

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

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

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 $\mathbf{m} \in \{0, 1\}^l$
- $\mathit{Sign}(sk, \mathbf{m}) = \mathbf{Xm} + \mathbf{x}$, linear combination of secret key
- $\mathit{Verify}(pk, \mathbf{m}, \sigma)$ uses homomoprhic properties to check that

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

One-time signatures from anti-cyclic lattices

Fix hash function key $\mathbf{A} = [\mathbf{A}^{(1)} | \ldots | \mathbf{A}^{(m/n)}]$

Definition (Secret signing key)

$$\begin{aligned} \mathbf{x} &= [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m/n)}] \\ \mathbf{y} &= [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m/n)}] \end{aligned}$$

Definition (Public verif. key)

$$\begin{aligned} X &= h_{\mathbf{A}}(\mathbf{x}) = \sum \mathbf{A}^{(i)}\mathbf{x}^{(i)} \\ Y &= h_{\mathbf{A}}(\mathbf{y}) = \sum \mathbf{A}^{(i)}\mathbf{y}^{(i)} \end{aligned}$$

- Signing $\mathbf{m} \in \{0,1\}^n$:

$$\begin{aligned} \sigma_i &= \mathbf{x}^{(i)}\mathbf{M} + \mathbf{y}^{(i)} \\ \sigma &= (\sigma_1, \ldots, \sigma_{m/n}) \end{aligned}$$

- Verification:

Check if $h_{\mathbf{A}}(\sigma) = X\mathbf{M} + Y$

$$\mathbf{M} = \begin{bmatrix} m_1 & -m_n & \cdots & -m_2 \\ m_2 & m_1 & \cdots & -m_3 \\ \vdots & \vdots & \ddots & \vdots \\ m_n & m_{n-1} & \cdots & m_1 \end{bmatrix}$$

Efficiency and security

- Key generation, signing and verifying all require just 1 or 2 hash function computations in $\tilde{O}(n)$ time
- Secret key, public key and signature size are also $\tilde{O}(n)$ bits

Theorem (Lyubashevsky&Micciancio)

The one-time signature scheme is secure based on the worst-case hardness of approximating SVP/SIVP on anti-cyclic lattices within a factor $\gamma = n^2$

- Forgery $(\mathbf{M}, \sigma)$: $h_{\mathbf{A}}(\sigma) = X\mathbf{M} + Y$
- Use $\mathbf{x}, \mathbf{y}$ to sign $\mathbf{M}$: $h_{\mathbf{A}}(\sigma') = X\mathbf{M} + Y$
- If $\sigma \neq \sigma'$, then $h_{\mathbf{A}}(\sigma) = X\mathbf{M} + Y = h_{\mathbf{A}}(\sigma')$ is a collision!

That's all folks!

Later today:

- LWE: injective version of SIS, many more applications
- RingLWE: efficient version of LWE