Lattice Cryptography: an introduction

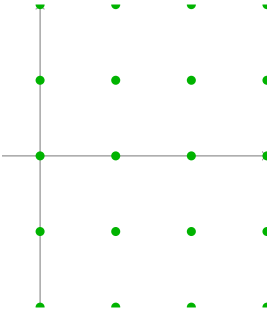
Daniele Micciancio

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

May 2015

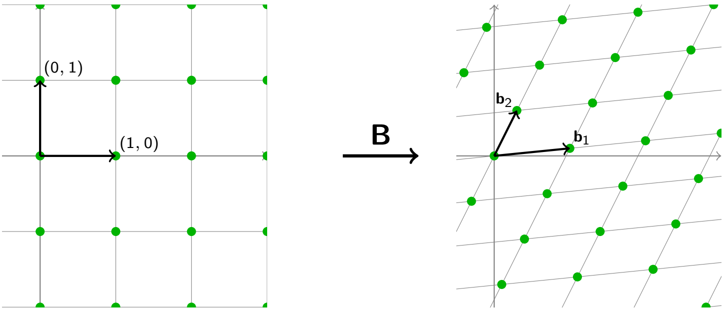
Point Lattices

- The simplest example of lattice is $\mathbb{Z}^n = \{(x_1, \ldots, x_n) \colon x_i \in \mathbb{Z}\}$



- The simplest example of lattice is $\mathbb{Z}^n = \{(x_1, \ldots, x_n) \colon x_i \in \mathbb{Z}\}$
- Other lattices are obtained by applying a linear transformation

$$\mathbf{B} \colon \mathbf{x} = (x_1, \ldots, x_n) \mapsto \mathbf{B}\mathbf{x} = x_1 \cdot \mathbf{b}_1 + \cdots + x_n \cdot \mathbf{b}_n$$

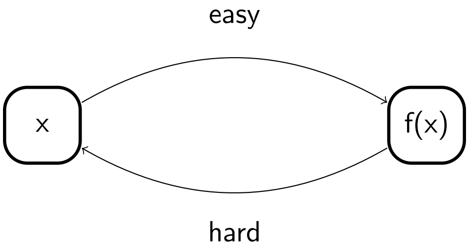


One-way Functions

Definition (One-Way Function (Informal))

An injective function $f : X \to Y$ is one-way if

- It is easy to compute, i.e., there is an efficient algorithm that on input x outputs $f(x)$
- It is hard to invert, i.e., there is no efficient algorithm that on input $f(x)$ outputs x



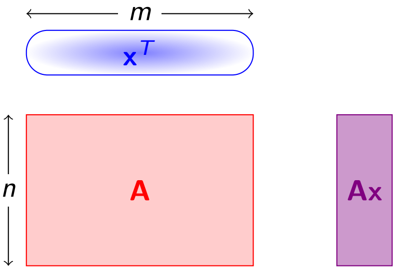
Outline

Modern Lattice Cryptography:

- The Short Integer Solusion (SIS) Function
  - Properties
  - Cryptographic Applications
- The Learning With Errors (LWE) Function
  - Properties
  - Cryptographic Applications
- Efficiency Considerations

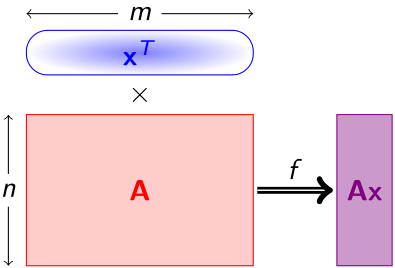
Ajtai's one-way function (SIS)

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



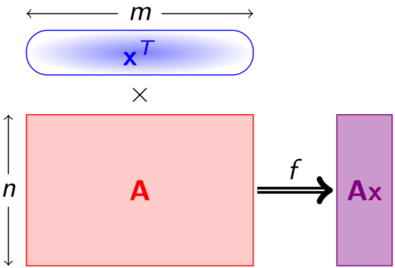
Ajtai's one-way function (SIS)

- 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] . . .

SIS: Properties and Applications

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

SIS Property 1: Compression

SIS Function

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

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

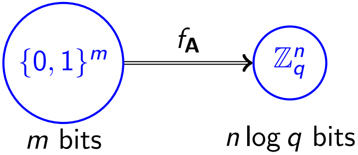
SIS Property 1: Compression

SIS Function

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

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

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



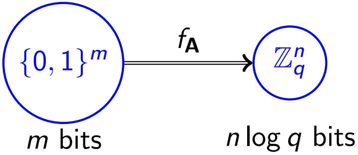
SIS Property 1: Compression

SIS Function

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

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.



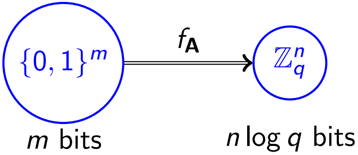
SIS Property 1: Compression

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., $m = 2n \lg q$: $f_{\mathbf{A}} : \{0,1\}^m \to \{0,1\}^{m/2}$.



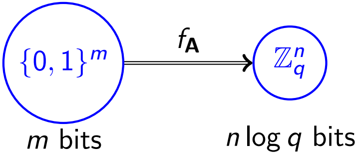
SIS Property 1: Compression

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.
- 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

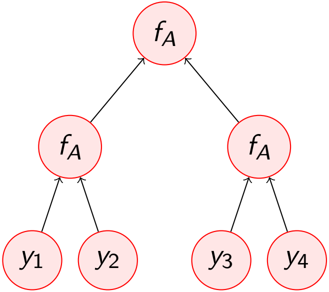
Keyed function family $f_A\colon X \to Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A\colon Y^2 \to Y$.)

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 1: 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 1: 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 1: 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 1: 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 1: 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 1: 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 (x, x') for $\mathbf{A}'$: $\mathbf{A}'\mathbf{x} = \mathbf{A}'\mathbf{x}'$

SIS Application 1: 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 (x, x') 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 2: Regularity

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

SIS Property 2: 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 2: Regularity

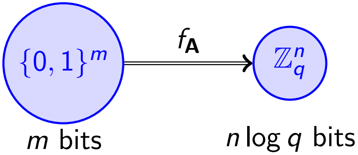
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

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 2: Regularity

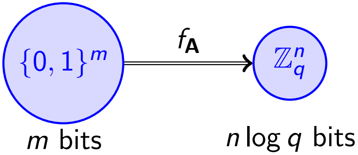
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

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 2: Regularity

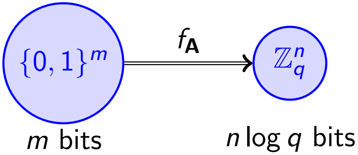
$f : X \to Y$ is regular if all $y \in Y$ have same $|f^{-1}(y)|$.

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.

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 2: Commitment

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

SIS Application 2: 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 2: 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 2: 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 2: 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 3: (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 3: (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 3: (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 3: 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 3: 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 3: 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 3: One-Time Signatures

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

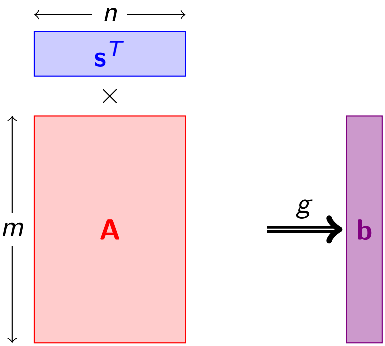
$$f_{\mathbf{A}}(\mathbf{X}) = [f_{\mathbf{A}}(\mathbf{x}_1), \ldots, f_{\mathbf{A}}(\mathbf{x}_l)] = \mathbf{AX} \pmod q$$

- 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 homomoprhic properties to check that

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

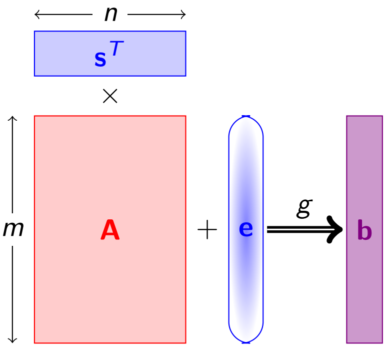
Learning with errors (LWE)

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



Learning with errors (LWE)

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

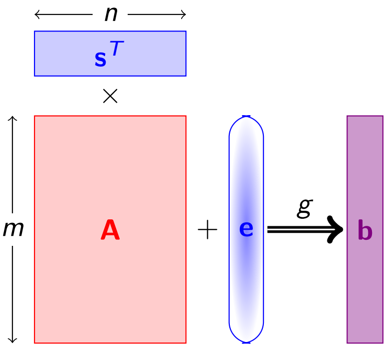


Learning with errors (LWE)

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

Theorem (Regev'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 even for quantum computers.



LWE: Properties and Applications

- Properties
  1. Injectivity
  2. Pseudorandomness
  3. Homomorphism
- Applications
  1. Symmetric Key Encryption
  2. Public Key Encryption

LWE Property 1: Injectivity

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

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

LWE Property 1: Injectivity

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s},\mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

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

$$\boxed{\mathbb{Z}_q^n \times \mathcal{E}^m} \xrightarrow{g_{\mathbf{A}}} \mathbb{Z}_q^m$$

$n \lg q + m \log |\mathcal{E}|$ bits $\qquad$ $m \log q$ bits

- Regev's theorem requires error $|\mathcal{E}| > \sqrt{n}$ and follow a certain nonuniform (Gaussian) distribution

LWE Property 1: Injectivity

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

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

$$\boxed{\mathbb{Z}_q^n \times \mathcal{E}^m} \xrightarrow{g_{\mathbf{A}}} \mathbb{Z}_q^m$$

$n \lg q + m \log |\mathcal{E}|$ bits $\qquad$ $m \log q$ bits

- Regev's theorem requires error $|\mathcal{E}| > \sqrt{n}$ and follow a certain nonuniform (Gaussian) distribution
- $g_{\mathbf{A}}$: $n \lg q + m \lg |\mathcal{E}|$ bits $\rightarrow$ $m \lg q$ bits.

LWE Property 1: Injectivity

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

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

$$\boxed{\mathbb{Z}_q^n \times \mathcal{E}^m} \xrightarrow{\quad g_{\mathbf{A}} \quad} \mathbb{Z}_q^m$$

$n \lg q + m \log |\mathcal{E}|$ bits $\qquad\qquad$ $m \log q$ bits

- Regev's theorem requires error $|\mathcal{E}| > \sqrt{n}$ and follow a certain nonuniform (Gaussian) distribution
- $g_{\mathbf{A}}$: $n \lg q + m \lg |\mathcal{E}|$ bits $\rightarrow$ $m \lg q$ bits.
- $g_{\mathbf{A}}$ expands the input roughly by a factor $\log q / \log |\mathcal{E}|$, and is injective with high probability

LWE: Learning Formulation

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

Each row of $\mathbf{A}$ and $\mathbf{x}$ gives a pair $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$

LWE: Learning Formulation

LWE Function

$\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{x} \leftarrow \mathcal{E}^m$, $\quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{x}) = \mathbf{A}\mathbf{s} + \mathbf{x} \bmod q \in \mathbb{Z}_q^m$

Each row of $\mathbf{A}$ and $\mathbf{x}$ gives a pair $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$

Definition (Learning With Errors (search version))

Given samples $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$ for fixed $\mathbf{s}$, and random $\mathbf{a}_i \in \mathbf{Z}_q^n$, $\mathbf{x}_i \leftarrow \mathcal{E}$, learn $\mathbf{s}$.

Pseudorandoness

- One-wayness is not usually enough for cryptographic security. Typically, one expects $f(x)$ to "look" random.

Pseudorandoness

- One-wayness is not usually enough for cryptographic security. Typically, one expects $f(x)$ to "look" random.

$$f : X \to Y$$

$$g : X \to Y \times Y$$

$$g(x) = (f(x), f(x))$$

Pseudorandoness

- One-wayness is not usually enough for cryptographic security. Typically, one expects $f(x)$ to "look" random.

$$f : X \to Y$$

$$g : X \to Y \times Y$$

$$g(x) = (f(x), f(x))$$

- If f is one-way, then g is also one-way

Pseudorandoness

- One-wayness is not usually enough for cryptographic security. Typically, one expects $f(x)$ to "look" random.

$$f : X \to Y$$

$$g : X \to Y \times Y$$

$$g(x) = (f(x), f(x))$$

- If f is one-way, then g is also one-way
- The output of $g(x)$ does not look random at all!

Pseudorandoness

- One-wayness is not usually enough for cryptographic security. Typically, one expects $f(x)$ to "look" random.

$$f : X \to Y$$

$$g : X \to Y \times Y$$

$$g(x) = (f(x), f(x))$$

- If f is one-way, then g is also one-way
- The output of $g(x)$ does not look random at all!

Definition (Pseudorandom Generator (PRG))

A function $f : X \to Y$ is a pseudorandom generator if for every efficient algorithm $\mathcal{D}$, $\Pr_{x \in X}\{\mathcal{D}(f(x)) = 1\} \approx \Pr_{y \in Y}\{\mathcal{D}(y) = 1\}$.

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ *is pseudorandomn.*

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.
- Assume $\mathcal{D}$ can distinguish $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$ from random

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.
- Assume $\mathcal{D}$ can distinguish $(\mathbf{a}_i, \mathbf{a}_i \mathbf{s} + x_i)$ from random
- Task: given many $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + x_i)$, find $\mathbf{s}$

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.
- Assume $\mathcal{D}$ can distinguish $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$ from random
- Task: given many $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + x_i)$, find $\mathbf{s}$
- Recover $\mathbf{s}$ one piece at a time:

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.
- Assume $\mathcal{D}$ can distinguish $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$ from random
- Task: given many $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + x_i)$, find $\mathbf{s}$
- Recover $\mathbf{s}$ one piece at a time:
  1. Pick random $\mathbf{r} \in \mathbb{Z}_q^n$, and guess $v \stackrel{?}{=} \mathbf{r} \cdot \mathbf{s} \in \mathbb{Z}_q$

LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_{\mathbf{A}}(\mathbf{s}, \mathbf{x})$ is pseudorandomn.

Easy proof using learning formulation:

- Assume small prime q, and very large m. Fix secret $\mathbf{s} \in \mathbb{Z}_q^n$.
- Assume $\mathcal{D}$ can distinguish $(\mathbf{a}_i, \mathbf{a}_i\mathbf{s} + x_i)$ from random
- Task: given many $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + x_i)$, find $\mathbf{s}$
- Recover $\mathbf{s}$ one piece at a time:
  1. Pick random $\mathbf{r} \in \mathbb{Z}_q^n$, and guess $v \stackrel{?}{=} \mathbf{r} \cdot \mathbf{s} \in \mathbb{Z}_q$
  2. Call $\mathcal{D}(\mathbf{a}_i + \mathbf{r}, b_i + v)$ to check if guess $v = \mathbf{r} \cdot \mathbf{s}$ was correct

Symmetric Encryption

- Definition
  - Key Generation: $sk \leftarrow KeyGen$
  - (Randomized) Encryption Algorithm: $c \leftarrow Enc(sk, m)$
  - Decryption Algorithm: $m \leftarrow Dec(sk, m)$

Symmetric Encryption

- Definition
  - Key Generation: $sk \leftarrow KeyGen$
  - (Randomized) Encryption Algorithm: $c \leftarrow Enc(sk, m)$
  - Decryption Algorithm: $m \leftarrow Dec(sk, m)$
- Security
  1. Pick secret key $sk \leftarrow KeyGen$
  2. Adversary makes encryption queries $m_1, m_2, \ldots \leftarrow \mathcal{A}$
  3. Adversary cannot distinguish $Enc(sk, m_i)$ from $Enc(sk, 0)$

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- Encryption: $Enc(\mathbf{s}, m) = (\mathbf{a}_i, b_i = g_{\mathbf{a}_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- Encryption: $Enc(\mathbf{s}, m) = (\mathbf{a}_i, b_i = g_{\mathbf{a}_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$
- Decryption: $Dec(\mathbf{s}, (\mathbf{a}_i, b_i))$ computes

$$b_i - \mathbf{a}_i \cdot s = x_i + E(m)$$

and rounds to 0 or $q/2$.

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- Encryption: $Enc(\mathbf{s}, m) = (\mathbf{a}_i, b_i = g_{\mathbf{a}_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$
- Decryption: $Dec(\mathbf{s}, (\mathbf{a}_i, b_i))$ computes

$$b_i - \mathbf{a}_i \cdot s = x_i + E(m)$$

  and rounds to 0 or $q/2$.
- Correctness: if $|x_i| < q/4$, decryption is correct

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- Encryption: $Enc(\mathbf{s}, m) = (\mathbf{a}_i, b_i = g_{\mathbf{a}_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$
- Decryption: $Dec(\mathbf{s}, (\mathbf{a}_i, b_i))$ computes

$$b_i - \mathbf{a}_i \cdot s = x_i + E(m)$$

  and rounds to 0 or $q/2$.
- Correctness: if $|x_i| < q/4$, decryption is correct
- Notice: if $g_{\mathbf{a}_i}(\mathbf{s}, x_i)$ were unformly random, b_i would also be random and independent of m

LWE Application 1: Symmetric Encryption

- Secret Key: $\mathbf{s} \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- Encryption: $Enc(\mathbf{s}, m) = (\mathbf{a}_i, b_i = g_{\mathbf{a}_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$
- Decryption: $Dec(\mathbf{s}, (\mathbf{a}_i, b_i))$ computes

$$b_i - \mathbf{a}_i \cdot s = x_i + E(m)$$

  and rounds to 0 or $q/2$.
- Correctness: if $|x_i| < q/4$, decryption is correct
- Notice: if $g_{\mathbf{a}_i}(\mathbf{s}, x_i)$ were unformly random, b_i would also be random and independent of m
- Security: If can distinguish $E(sk, m)$ from $E(sk, 0)$, then can distinguish $g_{\mathbf{a}_i}(\mathbf{s}, x_i)$ from random.

LWE Property 3: Homomorphism

- The LWE function is linearly homomorphic

$$g_{\mathbf{A}_1}(\mathbf{s}, \mathbf{x}_1) + g_{\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_2) = g_{\mathbf{A}_1+\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_1 + \mathbf{x}_2)$$

LWE Property 3: Homomorphism

- The LWE function is linearly homomorphic

$$g_{\mathbf{A}_1}(\mathbf{s}, \mathbf{x}_1) + g_{\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_2) = g_{\mathbf{A}_1+\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_1 + \mathbf{x}_2)$$

- LWE encryption inherits omomorphic property:

$$Enc(sk, m_1) + Enc(sk, m_2) \approx Enc(sk, m_1 + m_2)$$

$$(\mathbf{a}_1, g_{\mathbf{a}_1}(\mathbf{s}, x_1) + \frac{q}{2} m_1) + (\mathbf{a}_2, g_{\mathbf{a}_2}(\mathbf{s}, x_2) + \frac{q}{2} m_2)$$

$$= (\mathbf{a}_1 + \mathbf{a}_2, g_{\mathbf{a}_1+\mathbf{a}_2}(\mathbf{s}, x_1 + x_2) + \frac{q}{2}(m_1 + m_2))$$

LWE Property 3: Homomorphism

- The LWE function is linearly homomorphic

$$g_{\mathbf{A}_1}(\mathbf{s}, \mathbf{x}_1) + g_{\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_2) = g_{\mathbf{A}_1+\mathbf{A}_2}(\mathbf{s}, \mathbf{x}_1 + \mathbf{x}_2)$$

- LWE encryption inherits omomorphic property:

$$Enc(sk, m_1) + Enc(sk, m_2) \approx Enc(sk, m_1 + m_2)$$

$$(\mathbf{a}_1, g_{\mathbf{a}_1}(\mathbf{s}, x_1) + \frac{q}{2} m_1) + (\mathbf{a}_2, g_{\mathbf{a}_2}(\mathbf{s}, x_2) + \frac{q}{2} m_2)$$

$$= (\mathbf{a}_1 + \mathbf{a}_2, g_{\mathbf{a}_1+\mathbf{a}_2}(\mathbf{s}, x_1 + x_2) + \frac{q}{2}(m_1 + m_2))$$

- The errors x_i add up. Still, if initial x_i are small, and few ciphertexts are added, result is decryptable.

LWE Application 2: Public Key Encryption

- Use homomorphic properties to transform symmetric *Enc* into public key encryption scheme

LWE Application 2: Public Key Encryption

- Use homomorphic properties to transform symmetric *Enc* into public key encryption scheme
- Key Generation:
  1. Pick secret key $sk \leftarrow KeyGen$ for Enc
  2. Public key $pk = (p_1, \ldots, p_n)$ equals $p_i = Enc(sk, 0)$

LWE Application 2: Public Key Encryption

- Use homomorphic properties to transform symmetric *Enc* into public key encryption scheme
- Key Generation:
  1. Pick secret key $sk \leftarrow KeyGen$ for Enc
  2. Public key $pk = (p_1, \ldots, p_n)$ equals $p_i = Enc(sk, 0)$
- Encryption of m: pick small random r_i and output

$$\begin{aligned}\sum_i r_i \cdot p_i + m &= \sum_i r_i \cdot Enc(sk, 0) + m \\ &= Enc(sk, \sum_i r_i \cdot 0 + m) = Enc(sk, m)\end{aligned}$$

LWE Application 2: Public Key Encryption

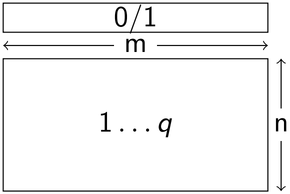
- Use homomorphic properties to transform symmetric *Enc* into public key encryption scheme
- Key Generation:
  1. Pick secret key $sk \leftarrow KeyGen$ for *Enc*
  2. Public key $pk = (p_1, \ldots, p_n)$ equals $p_i = Enc(sk, 0)$
- Encryption of m: pick small random r_i and output

$$
\begin{aligned}
\sum_i r_i \cdot p_i + m &= \sum_i r_i \cdot Enc(sk, 0) + m \\
&= Enc(sk, \sum_i r_i \cdot 0 + m) = Enc(sk, m)
\end{aligned}
$$

- Decryption: same as before
- if p_i has error x_i, then $E(pk, m)$ has error $\sum_i r_i x_i$

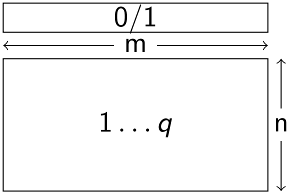
Efficiency of Ajtai's function

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



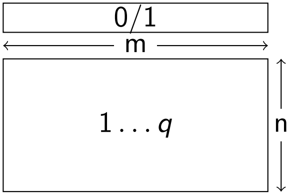
Efficiency of Ajtai's function

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

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?

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?

$$\begin{array}{|cccc|cccc|cccc|cccc|}
? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? & ? \\
\hline
1 & 4 & 3 & 8 & 6 & 4 & 9 & 0 & 2 & 6 & 4 & 5 & 3 & 2 & 7 & 1 \\
8 & 1 & 4 & 3 & 0 & 6 & 4 & 9 & 5 & 2 & 6 & 4 & 1 & 3 & 2 & 7 \\
3 & 8 & 1 & 4 & 9 & 0 & 6 & 4 & 4 & 5 & 2 & 6 & 7 & 1 & 3 & 2 \\
4 & 3 & 8 & 1 & 4 & 9 & 0 & 6 & 6 & 4 & 5 & 2 & 2 & 7 & 1 & 3 \\
\hline
\end{array}
\quad
\begin{array}{|c|}
\\
\hline
0 \\
0 \\
0 \\
0 \\
\hline
\end{array}$$

Can you find a collision?

$$
\left|\begin{array}{cccc|cccc|cccc|cccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 4 & 3 & 8 & 6 & 4 & 9 & 0 & 2 & 6 & 4 & 5 & 3 & 2 & 7 & 1 \\
8 & 1 & 4 & 3 & 0 & 6 & 4 & 9 & 5 & 2 & 6 & 4 & 1 & 3 & 2 & 7 \\
3 & 8 & 1 & 4 & 9 & 0 & 6 & 4 & 4 & 5 & 2 & 6 & 7 & 1 & 3 & 2 \\
4 & 3 & 8 & 1 & 4 & 9 & 0 & 6 & 6 & 4 & 5 & 2 & 2 & 7 & 1 & 3
\end{array}\right|
\qquad
\left|\begin{array}{c} \\ \hline \\ \\ \\ \end{array}\right|
$$

$$
\boxed{\begin{array}{c} 6 \\ 6 \\ 6 \\ 6 \end{array}}
\qquad
\boxed{\begin{array}{c} 9 \\ 9 \\ 9 \\ 9 \end{array}}
\qquad
\boxed{\begin{array}{c} 7 \\ 7 \\ 7 \\ 7 \end{array}}
\qquad
\boxed{\begin{array}{c} 3 \\ 3 \\ 3 \\ 3 \end{array}}
$$

Can you find a collision?

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?

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
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?

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
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?

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
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 (Lyubashevsky&Micciancio)

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

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 approximate 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

Conclusion

- Simple SIS/LWE functions
- Useful homomorphic properties $\Rightarrow$ Cryptographic applications
- Cyclic/Anticycic matrices (RingSIS/RingLWE):
  - key to efficiency in practice
  - technique pervasively used by all practical instantiations of lattice cryptography
- Question: Are these functions secure?
  - We think so, and that's where lattices come into the picture
  - ... but that's another story