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 $\mathbb{Z}^n = \{(x_1, \ldots, x_n): x_i \in \mathbb{Z}\}$
Point Lattices

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

$$B: x = (x_1, \ldots, x_n) \mapsto Bx = x_1 \cdot b_1 + \cdots + x_n \cdot b_n$$
Definition (One-Way Function (Informal))

An injective function \( f : X \rightarrow Y \) 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 Solution (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: \( A \in \mathbb{Z}_q^{n \times m} \)
- Input: \( x \in \{0, 1\}^m \)

Theorem (A’96)

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

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

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

Theorem (A’96)

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

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

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

**Theorem (A’96)**

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

Applications: OWF [A’96], Hashing [GGH’97], Commit [KTX’08], 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**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

Main security parameter: \( n \). (Security largely independent of \( m \).)
**SIS Property 1: Compression**

**SIS Function**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

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

- \( f_A \): \( m \) bits \( \rightarrow \) \( n \log q \) bits.

\[ \{0, 1\}^m \xrightarrow{f_A} \mathbb{Z}_q^n \]

- \( m \) bits
- \( n \log q \) bits
SIS Property 1: Compression

**SIS Function**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

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

- \( f_A \): \( m \) bits \( \rightarrow \) \( n \log q \) bits.
- When \( (m > n \log q) \), \( f_A \) is a compression function.

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

Daniele Micciancio (UCSD)  Lattice Cryptography: an introduction  May 2015  7 / 32
SIS Property 1: Compression

SIS Function

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

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

- \( f_A \): \( m \) bits \( \rightarrow \) \( n \log q \) bits.
- When \( (m > n \log q) \), \( f_A \) is a compression function.
- E.g., \( m = 2n \log q \):
  - \( f_A : \{0, 1\}^m \rightarrow \{0, 1\}^{m/2} \).
SIS Property 1: Compression

**SIS Function**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

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

- \( f_A \): \( m \) bits \( \rightarrow n \log q \) bits.
- When \( (m > n \log q) \), \( f_A \) is a compression function.
- E.g., \( m = 2n \log q \):
  \[ f_A : \{0, 1\}^m \rightarrow \{0, 1\}^{m/2} \]

Ajtai’s theorem requires \( (m > n \log q) \)
Collision Resistant Hashing

Keyed function family $f_A : X \rightarrow Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A : Y^2 \rightarrow Y$.)
Collision Resistant Hashing

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

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 $f_A : X \rightarrow Y$ with $|X| > |Y|$
(E.g., $X = Y^2$ and $f_A : Y^2 \rightarrow Y$.)

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)
\[ f_A : X \rightarrow Y. \] 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)

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

Theorem

If \( f_A : \{0, \pm 1\}^m \rightarrow \mathbb{Z}_q^n \) is one-way, then \( f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n \) is collision resistant.
SIS Application 1: Collision Resistant Hashing

Definition (Collision Resistance)

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

Theorem

If $f_A : \{0, \pm 1\}^m \rightarrow \mathbb{Z}_q^n$ is one-way, then $f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n$ is collision resistant.

- Assume can find collisions to $f_A$
Definition (Collision Resistance)

\[ f_A : X \to Y. \text{ No adversary, given a random } A, \text{ can efficiently find } x \neq x' \in X \text{ such that } f_A(x) = f_A(x') \]

Theorem

If \( f_A : \{0, \pm 1\}^m \to \mathbb{Z}_q^n \) is one-way, then \( f_A : \{0,1\}^m \to \mathbb{Z}_q^n \) is collision resistant.

- Assume can find collisions to \( f_A \)
- Goal: Given random \( A \) and \( y \), find \( f_A(x) = y \)
SIS Application 1: Collision Resistant Hashing

Definition (Collision Resistance)

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

Theorem

*If \( f_A : \{0, \pm 1\}^m \rightarrow \mathbb{Z}_q^n \) is one-way, then \( f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n \) is collision resistant.*

- Assume can find collisions to \( f_A \)
- Goal: Given random \( A \) and \( y \), find \( f_A(x) = y \)
- Add \( y \) to random column \( a_i' = a_i + y \).
SIS Application 1: Collision Resistant Hashing

Definition (Collision Resistance)

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

Theorem

If \( f_A : \{0, \pm 1\}^m \rightarrow \mathbb{Z}_q^n \) is one-way, then \( f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n \) is collision resistant.

- Assume can find collisions to \( f_A \)
- Goal: Given random \( A \) and \( y \), find \( f_A(x) = y \)
- Add \( y \) to random column \( a'_i = a_i + y \)
- Find collision \((x, x')\) for \( A' : A'x = A'x' \)
SIS Application 1: Collision Resistant Hashing

Definition (Collision Resistance)

\[ f_A : X \rightarrow Y. \] No adversary, given a random \( A \), can efficiently find \( x \neq x' \in X \) such that \( f_A(x) = f_A(x') \)

Theorem

*If \( f_A : \{0, \pm1\}^m \rightarrow \mathbb{Z}_q^n \) is one-way, then \( f_A : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n \) is collision resistant.*

- Assume can find collisions to \( f_A \)
- Goal: Given random \( A \) and \( y \), find \( f_A(x) = y \)
- Add \( y \) to random column \( a'_i = a_i + y \).
- Find collision \((x, x')\) for \( A' : A'x = A'x' \)
- If \( x'_i = 1 \) and \( x_i = 0 \), then \( A(x - x') = y \)
SIS Property 2: Regularity

\[ f : X \to Y \text{ is regular if all } y \in Y \text{ have same } |f^{-1}(y)|. \]
SIS Property 2: Regularity

\[ f : X \rightarrow Y \text{ is regular if all } y \in Y \text{ have same } |f^{-1}(y)|. \]

**SIS Function**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]
SIS Property 2: Regularity

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

**SIS Function**

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

**Pairwise independence:**

- Fix \( x_1 \neq x_2 \in \{0, 1\}^m \),
- Random \( A \)
- \( f_A(x_1) \) and \( f_A(x_2) \) are independent.
**SIS Property 2: Regularity**

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

**SIS Function**

\[
A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n
\]

Pairwise independence:

- Fix \( x_1 \neq x_2 \in \{0, 1\}^m \),
- Random \( A \)
- \( f_A(x_1) \) and \( f_A(x_2) \) are independent.

**Lemma (Leftover Hash Lemma)**

Pairwise Independence + Compression \( \Rightarrow \) Regular
SIS Property 2: Regularity

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

**SIS Function**

\[
A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n
\]

Pairwise independence:

- Fix \( x_1 \neq x_2 \in \{0, 1\}^m \),
- Random \( A \)
- \( f_A(x_1) \) and \( f_A(x_2) \) are independent.

**Lemma (Leftover Hash Lemma)**

*Pairwise Independence + Compression \(\implies\) Regular*

\( f_A : (U(\{0, 1\}^n)) \approx U(\mathbb{Z}_q^n) \) maps uniform to uniform.
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 \( C(m; r) = C(m'; r') \).
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 $A_1, A_2$ at random
SIS Application 2: Commitment

- Choose \( A_1, A_2 \) at random
- message \( m \in \{0, 1\}^m \) and randomness \( r \in \{0, 1\}^m \)
SIS Application 2: Commitment

- Choose $A_1, A_2$ at random
- message $m \in \{0, 1\}^m$ and randomness $r \in \{0, 1\}^m$
- Commitment: $C(m, r) = f_{[A_1, A_2]}(m, r) = A_1m + A_2r$. 
SIS Application 2: Commitment

- Choose $A_1, A_2$ at random
- message $m \in \{0, 1\}^m$ and randomness $r \in \{0, 1\}^m$
- Commitment: $C(m, r) = f_{[A_1, A_2]}(m, r) = A_1m + A_2r$.
- Hiding Property: $C(m)$ hides the message because $A_2r = f_{A_2}(r) \approx U(\mathbb{Z}_q^n)$
Choose $A_1, A_2$ at random

- message $m \in \{0, 1\}^m$ and randomness $r \in \{0, 1\}^m$
- Commitment: $C(m, r) = f_{[A_1, A_2]}(m, r) = A_1m + A_2r$.
- Hiding Property: $C(m)$ hides the message because $A_2r = f_{A_2}(r) \approx U(\mathbb{Z}_q^n)$
- Binding Property: Finding $(m, r) \neq (m', r')$ such that $C(m, r) = C(m', r')$ breaks the collision resistance of $f_{[A_1, A_2]}$
SIS Property 3: (Approximate) Linear Homomorphism

SIS Function

\( A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \)

- The SIS function is linearly homomorphic:

\[ f_A(x_1) + f_A(x_2) = f_A(x_1 + x_2) \]
SIS Property 3: (Approximate) Linear Homomorphism

SIS Function

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

- 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

\[ A \in \mathbb{Z}_q^{n \times m}, \quad x \in \{0, 1\}^m, \quad f_A(x) = Ax \mod q \in \mathbb{Z}_q^n \]

- The SIS function is linearly homomorphic:

\[ f_A(x_1) + f_A(x_2) = f_A(x_1 + x_2) \]

- Homomorphism is only approximate:
  - If \( x_1, x_2 \) are small, then also \( x_1 + x_2 \) is small
  - However, \( x_1 + x_2 \) can be slightly larger than \( x_1, x_2 \)
  - Domain of \( f_A \) is not closed under +

- \( f_A \) is also key-homomorphic:

\[ f_{A_1}(x) + f_{A_2}(x) = f_{A_1 + A_2}(x) \]
(One-Time) Digital Signatures

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

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

- (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: \((pk, sk) \leftarrow \text{KeyGen}\)
  - Signing Algorithm: \(\text{Sign}(sk, m) = \sigma\)
  - Verification Algorithm: \(\text{Verify}(pk, m, \sigma)\)

- (One-Time) Security:
  - 1. Generate keys \((pk, sk) \leftarrow \text{KeyGen}\)
  - 2. Adversary \(m \leftarrow \text{Adv}(pk)\) 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: \((pk, sk) \leftarrow \text{KeyGen}\)
- Signing Algorithm: \(\text{Sign}(sk, m) = \sigma\)
- Verification Algorithm: \(\text{Verify}(pk, m, \sigma)\)

(One-Time) Security:
1. Generate keys \((pk, sk) \leftarrow \text{KeyGen}\)
2. Adversary \(m \leftarrow \text{Adv}(pk)\) chooses message query
3. ... receives signature \(\sigma \leftarrow \text{Sign}(s, m)\),
4. ... and outputs forgery \((m', \sigma') \leftarrow \text{Adv}(\sigma)\).
(One-Time) Digital Signatures

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

(One-Time) Security:
1. Generate keys \((pk, sk) \leftarrow \text{KeyGen}\)
2. Adversary \(m \leftarrow \text{Adv}(pk)\) chooses message query
3. ... receives signature \(\sigma \leftarrow \text{Sign}(s, m)\),
4. ... and outputs forgery \((m', \sigma') \leftarrow \text{Adv}(\sigma)\).
5. Adversary wins if \(\text{Verify}(m', \sigma')\) 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. \(\ldots\) receives signature \(\sigma \leftarrow Sign(s, m)\),
  4. \(\ldots\) 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 $f_A$ to matrices $X = [x_1, \ldots, x_l]$:

  $$f_A(X) = [f_A(x_1), \ldots, f_A(x_l)] = AX \pmod q$$
SIS Application 3: One-Time Signatures

- Extend $f_A$ to matrices $X = [x_1, \ldots, x_l]$:

  $$f_A(X) = [f_A(x_1), \ldots, f_A(x_l)] = AX \pmod{q}$$

- Key Generation:
  
  - Public Parameter: SIS function key $A$
  - Secret Key: $sk = (X, x)$ two (small) inputs to $f_A$
  - Public Key: $pk = (Y = f_A(X), y = f_A(x))$ image of $sk$ under $f_A$
SIS Application 3: One-Time Signatures

- Extend $f_A$ to matrices $X = [x_1, \ldots, x_l]$:
  
  $$f_A(X) = [f_A(x_1), \ldots, f_A(x_l)] = AX \pmod{q}$$

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

- Message: short vector $m \in \{0, 1\}^l$

- $Sign(sk, m) = Xm + x$, linear combination of secret key
SIS Application 3: One-Time Signatures

- Extend \( f_A \) to matrices \( X = [x_1, \ldots, x_l] \):

\[
f_A(X) = [f_A(x_1), \ldots, f_A(x_l)] = AX \pmod{q}
\]

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

- Message: short vector \( m \in \{0, 1\}^l \)

- \( \text{Sign}(sk, m) = Xm + x \), linear combination of secret key

- \( \text{Verify}(pk, m, \sigma) \) uses homomorphic properties to check that

\[
f_A(\sigma) = f_A(Xm + x) = f_A(X)m + f_A(x) = Ym + y
\]
Learning with errors (LWE)

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

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

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

**Theorem (Regev’05)**

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

\[ A \in \mathbb{Z}_q^{m \times n}, \ s \in \mathbb{Z}_q^n, \ x \leftarrow \mathcal{E}^m, \quad g_A(s, x) = As + x \mod q \in \mathbb{Z}_q^m \]

Main security parameter: \( n \). (Security largely independent of \( m \).)
LWE Property 1: Injectivity

LWE Function

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

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

\[ n \log q + m \log |\mathcal{E}| \text{ bits} \quad m \log q \text{ bits} \]

- Regev’s theorem requires error \( |\mathcal{E}| > \sqrt{n} \) and follow a certain nonuniform (Gaussian) distribution
LWE Property 1: Injectivity

LWE Function

\[ \mathbf{A} \in \mathbb{Z}_q^{m \times n}, \; \mathbf{s} \in \mathbb{Z}_q^n, \; \mathbf{x} \leftarrow \mathcal{E}^m, \; \quad g\mathbf{A}(\mathbf{s}, \mathbf{x}) = \mathbf{A}s + \mathbf{x} \mod q \in \mathbb{Z}_q^m \]

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

\[ \mathbb{Z}_q^n \times \mathcal{E}^m \xrightarrow{g\mathbf{A}} \mathbb{Z}_q^m \]

- \( n \log q + m \log |\mathcal{E}| \) bits
- \( m \log q \) bits

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

- \( g\mathbf{A} \): \( n \log q + m \log |\mathcal{E}| \) bits \( \rightarrow \) \( m \log q \) bits.
LWE Property 1: Injectivity

LWE Function

\[
A \in \mathbb{Z}_q^{m \times n}, \quad s \in \mathbb{Z}_q^n, \quad x \leftarrow \mathcal{E}^m, \quad g_A(s, x) = As + x \mod q \in \mathbb{Z}_q^m
\]

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

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

LWE Function

\[ \mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathcal{E}^m, \quad g_A(s, x) = \mathbf{A}s + \mathbf{x} \mod q \in \mathbb{Z}_q^m \]

Each row of \( \mathbf{A} \) and \( \mathbf{x} \) gives a pair \((a_i, a_is + x_i)\)
LWE: Learning Formulation

**LWE Function**

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

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

**Definition (Learning With Errors (search version))**

Given samples \((a_i, a_i \mathbf{s} + x_i)\) for fixed \( \mathbf{s} \), and random \( a_i \in \mathbb{Z}_q^n, \, x_i \leftarrow \mathcal{E} \), learn \( \mathbf{s} \).
Pseudorandomness

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

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

\[
f : X \rightarrow Y
\]

\[
g : X \rightarrow Y \times Y
\]

\[
g(x) = (f(x), f(x))
\]
Pseudorandomness

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

$$f : X \rightarrow Y$$

$$g : X \rightarrow Y \times Y$$

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

- If $f$ is one-way, then $g$ is also one-way.
Pseudorandomness

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

\[ f : X \rightarrow Y \]

\[ g : X \rightarrow Y \times Y \]

\[ g(x) = (f(x), f(x)) \]

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

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

$$f : X \rightarrow Y$$

$$g : X \rightarrow Y \times Y$$

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

- 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 $f : X \rightarrow Y$ is a pseudorandom generator if for every efficient algorithm $\mathcal{D}$, $\Pr_{x \in X}\{\mathcal{D}(f(x)) = 1\} \approx \Pr_{y \in Y}\{\mathcal{D}(y) = 1\}$.
Theorem (Pseudorandomness of LWE)

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

Easy proof using learning formulation:
LWE Property 2: Pseudorandomness

**Theorem (Pseudorandomness of LWE)**

*If (search) LWE is hard, then $g_A(s, x)$ is pseudorandom.*

Easy proof using learning formulation:

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

Daniele Micciancio (UCSD)

Lattice Cryptography: an introduction

May 2015 21 / 32
Theorem (Pseudorandomness of LWE)

*If (search) LWE is hard, then* $g_A(s, x)$ *is pseudorandom.*

Easy proof using learning formulation:

- Assume small prime $q$, and very large $m$. Fix secret $s \in \mathbb{Z}_q^n$.
- Assume $D$ can distinguish $(a_i, a_i s + x_i)$ from random
LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_A(s, x)$ is pseudorandom.

Easy proof using learning formulation:

- Assume small prime $q$, and very large $m$. Fix secret $s \in \mathbb{Z}_q^n$.
- Assume $D$ can distinguish $(a_i, a_is + x_i)$ from random
- Task: given many $(a_i, b_i = a_i \cdot s + x_i)$, find $s$
LWE Property 2: Pseudorandomness

**Theorem (Pseudorandomness of LWE)**

If (search) LWE is hard, then $g_A(s, x)$ is pseudorandom.

Easy proof using learning formulation:

- Assume small prime $q$, and very large $m$. Fix secret $s \in \mathbb{Z}_q^n$.
- Assume $D$ can distinguish $(a_i, a_is + x_i)$ from random
- Task: given many $(a_i, b_i = a_i \cdot s + x_i)$, find $s$
- Recover $s$ one piece at a time:
LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_A(s, x)$ is pseudorandom.

Easy proof using learning formulation:

- Assume small prime $q$, and very large $m$. Fix secret $s \in \mathbb{Z}_q^n$.
- Assume $D$ can distinguish $(a_i, a_i s + x_i)$ from random
- Task: given many $(a_i, b_i = a_i \cdot s + x_i)$, find $s$
- Recover $s$ one piece at a time:
  - Pick random $r \in \mathbb{Z}_q^n$, and guess $v = r \cdot s \in \mathbb{Z}_q$
LWE Property 2: Pseudorandomness

Theorem (Pseudorandomness of LWE)

If (search) LWE is hard, then $g_A(s, x)$ is pseudorandom.

Easy proof using learning formulation:

- Assume small prime $q$, and very large $m$. Fix secret $s \in \mathbb{Z}_q^n$.
- Assume $D$ can distinguish $(a_i, a_i s + x_i)$ from random
- Task: given many $(a_i, b_i = a_i \cdot s + x_i)$, find $s$
- Recover $s$ one piece at a time:
  1. Pick random $r \in \mathbb{Z}_q^n$, and guess $v = r \cdot s \in \mathbb{Z}_q$
  2. Call $D(a_i + r, b_i + v)$ to check if guess $v = r \cdot s$ 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: $sk \leftarrow \text{KeyGen}$
  - (Randomized) Encryption Algorithm: $c \leftarrow \text{Enc}(sk, m)$
  - Decryption Algorithm: $m \leftarrow \text{Dec}(sk, m)$

- **Security**
  1. Pick secret key $sk \leftarrow \text{KeyGen}$
  2. Adversary makes encryption queries $m_1, m_2, \ldots \leftarrow \mathcal{A}$
  3. Adversary cannot distinguish $\text{Enc}(sk, m_i)$ from $\text{Enc}(sk, 0)$
LWE Application 1: Symmetric Encryption

- Secret Key: \( s \in \mathbb{Z}_q^n \). Assume \( m \in \{0, 1\} \).
LWE Application 1: Symmetric Encryption

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

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

\[
b_i - a_i \cdot s = x_i + E(m)
\]

and rounds to 0 or \( q/2 \).
LWE Application 1: Symmetric Encryption

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

- Encryption: \( Enc(s, m) = (a_i, b_i = g_{a_i}(s, x_i) + E(m)) \) where \( E(m) = \frac{q}{2} m \)

- Decryption: \( Dec(s, (a_i, b_i)) \) computes

\[
    b_i - a_i \cdot s = x_i + E(m)
\]

and rounds to 0 or \( q/2 \).

- Correctness: if \( |x_i| < q/4 \), decryption is correct
LWE Application 1: Symmetric Encryption

- **Secret Key:** $s \in \mathbb{Z}_q^n$. Assume $m \in \{0, 1\}$.
- **Encryption:** $Enc(s, m) = (a_i, b_i = g_{a_i}(s, x_i) + E(m))$ where $E(m) = \frac{q}{2} m$
- **Decryption:** $Dec(s, (a_i, b_i))$ computes
  \[ b_i - a_i \cdot s = x_i + E(m) \]
  and rounds to 0 or $q/2$.
- **Correctness:** if $|x_i| < q/4$, decryption is correct
- **Notice:** if $g_{a_i}(s, x_i)$ were uniformly 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: $\text{Enc}(\mathbf{s}, m) = (a_i, b_i = g_{a_i}(\mathbf{s}, x_i) + E(m))$ where $E(m) = \frac{q}{2}m$
- Decryption: $\text{Dec}(\mathbf{s}, (a_i, b_i))$ computes
  \[ b_i - a_i \cdot \mathbf{s} = x_i + E(m) \]
  and rounds to 0 or $q/2$.
- Correctness: if $|x_i| < q/4$, decryption is correct
- Notice: if $g_{a_i}(\mathbf{s}, x_i)$ were uniformly 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 $g_{a_i}(\mathbf{s}, x_i)$ from random.
LWE Property 3: Homomorphism

- The LWE function is linearly homomorphic

\[ g_{A_1}(s, x_1) + g_{A_2}(s, x_2) = g_{A_1+A_2}(s, x_1 + x_2) \]
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 homomorphic property:

\[ \text{Enc}(sk, m_1) + \text{Enc}(sk, m_2) \approx \text{Enc}(sk, m_1 + m_2) \]

\[ (a_1, g_{a_1}(s, x_1) + \frac{q}{2} m_1) + (a_2, g_{a_2}(s, x_2) + \frac{q}{2} m_2) \]

\[ = (a_1 + a_2, g_{a_1 + a_2}(s, x_1 + x_2) + \frac{q}{2} (m_1 + m_2)) \]
LWE Property 3: Homomorphism

- The LWE function is linearly homomorphic

\[ g_{A_1}(s, x_1) + g_{A_2}(s, x_2) = g_{A_1 + A_2}(s, x_1 + x_2) \]

- LWE encryption inherits homomorphic property:

\[
\begin{align*}
Enc(sk, m_1) + Enc(sk, m_2) & \approx Enc(sk, m_1 + m_2) \\
(a_1, g_{a_1}(s, x_1) + \frac{q}{2}m_1) + (a_2, g_{a_2}(s, x_2) + \frac{q}{2}m_2) & = (a_1 + a_2, g_{a_1 + a_2}(s, x_1 + x_2) + \frac{q}{2}(m_1 + m_2))
\end{align*}
\]

- 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

$$\sum_i r_i \cdot p_i + m = \sum_i r_i \cdot Enc(sk, 0) + m$$

$$= Enc(sk, \sum_i r_i \cdot 0 + m) = Enc(sk, m)$$
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

$$\sum_i r_i \cdot p_i + m = \sum_i r_i \cdot Enc(sk, 0) + m$$

$$= Enc(sk, \sum_i r_i \cdot 0 + m) = Enc(sk, m)$$

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., \( n = 64, q = 2^8, m = 1024 \)
- \( f_A \) 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., \( n = 64, q = 2^8, m = 1024 \)
- \( f_A \) 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., $n = 64$, $q = 2^8$, $m = 1024$
- $f_A$ maps 1024 bits to 512.
- Key size: $nm \log q = O(n^2 \log^2 n) = 2^{19} = 64\, KB$
- 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

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

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

\[
A^{(i)} = \begin{bmatrix}
    a^{(i)}_1 & a^{(i)}_n & \ldots & a^{(i)}_2 \\
    a^{(i)}_2 & a^{(i)}_1 & \ldots & a^{(i)}_3 \\
    \vdots & \vdots & \ddots & \vdots \\
    a^{(i)}_n & a^{(i)}_{n-1} & \ldots & a^{(i)}_1
\end{bmatrix}
\]
Efficient lattice based hashing

Idea

Use structured matrix

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

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

- Proposed by [M02], where it is proved that \( f_A \) 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 \( \Lambda_q^\perp(A) \) is hard, and therefore \( f_A \) is collision resistant
Can you find a collision?

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Can you find a collision?

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>-1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Sum of non-diagonal elements: 5 + 4 + 8 + 6 = 23
Can you find a collision?

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Can you find a collision?

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
Can you find a collision?

\[
\begin{array}{cccc|cccc|cccc|c}
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 4 & 3 & 8 & 6 & 4 & 9 & 0 & 2 & 6 & 4 & 5 & 3 & 2 & 7 & 1 & 0 \\
8 & 1 & 4 & 3 & 0 & 6 & 4 & 9 & 5 & 2 & 6 & 4 & 1 & 3 & 2 & 7 & 0 \\
3 & 8 & 1 & 4 & 9 & 0 & 6 & 4 & 4 & 5 & 2 & 6 & 7 & 1 & 3 & 2 & 0 \\
4 & 3 & 8 & 1 & 4 & 9 & 0 & 6 & 6 & 4 & 5 & 2 & 2 & 7 & 1 & 3 & 0 \\
\end{array}
\]

\[
\begin{array}{c}
+1 \times \\
6 \\
6 \\
6 \\
6 \\
\end{array}
\begin{array}{c}
-1 \times \\
9 \\
9 \\
9 \\
\end{array}
\begin{array}{c}
+0 \times \\
7 \\
7 \\
7 \\
\end{array}
\begin{array}{c}
+1 \times \\
3 \\
3 \\
3 \\
\end{array}
\]
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 $\Lambda([A^{(1)}|\ldots|A^{(m/n)}])$
- ...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?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td>-8</td>
<td>6</td>
<td>-4</td>
<td>-9</td>
<td>0</td>
<td>2</td>
<td>-6</td>
<td>-4</td>
<td>-5</td>
<td>3</td>
<td>-2</td>
<td>-7</td>
<td>-1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td>-9</td>
<td>5</td>
<td>2</td>
<td>-6</td>
<td>-4</td>
<td>1</td>
<td>3</td>
<td>-2</td>
<td>-7</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>-4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>-6</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Can you find a collision now?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td>-8</td>
<td>6</td>
<td>-4</td>
<td>-9</td>
<td>-0</td>
<td>2</td>
<td>-6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td>-9</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>-4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

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?

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td>-8</td>
<td></td>
<td>6</td>
<td>-4</td>
<td>-9</td>
<td>-0</td>
<td></td>
<td>2</td>
<td>-6</td>
<td>-4</td>
<td>-5</td>
<td></td>
<td>3</td>
<td>-2</td>
<td>-7</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>-4</td>
<td>-3</td>
<td></td>
<td>6</td>
<td>4</td>
<td>-4</td>
<td>9</td>
<td></td>
<td>5</td>
<td>2</td>
<td>-6</td>
<td>-4</td>
<td></td>
<td>1</td>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
<td>-4</td>
<td></td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td></td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>-6</td>
<td></td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td></td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td></td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

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: \(A(x + y) = Ax + Ay\)
- This can be a feature rather than a weakness
Conclusion

- Simple SIS/LWE functions
- Useful homomorphic properties $\Rightarrow$ Cryptographic applications
- Cyclic/Anticyclic 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