

Lattice Cryptography for the Internet

Chris Peikert

Georgia Institute of Technology

Post-Quantum Cryptography

2 October 2014

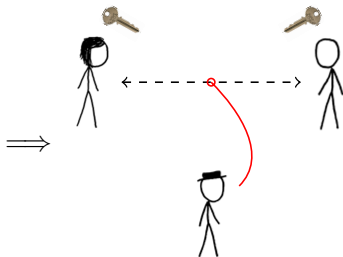
Lattice-Based Cryptography

$$y = g^x \bmod p$$

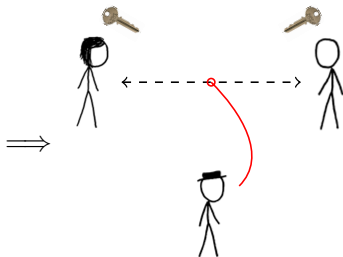
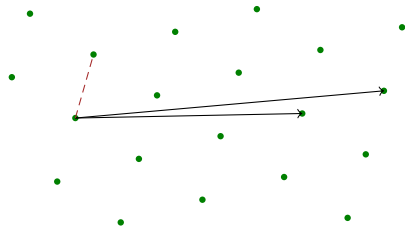
$$m^e \bmod N$$

$$e(g^a, g^b)$$

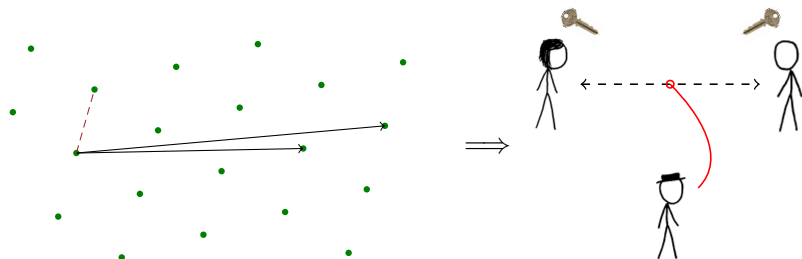
$$N = p \cdot q$$



Lattice-Based Cryptography



Lattice-Based Cryptography



Amazing!

- ▶ Simple, efficient, and highly parallel crypto schemes
- ▶ Resists attacks by quantum algorithms (so far)
- ▶ Security from worst-case complexity assumptions
- ▶ Solves “holy grail” problems in crypto: FHE, obfuscation, ...

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption
- ▶ Fully homomorphic encryption

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption
- ▶ Fully homomorphic encryption
- ▶ Functional encryption

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption
- ▶ Fully homomorphic encryption
- ▶ Functional encryption
- ▶ General-purpose obfuscation
- ▶ ...

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption
- ▶ Fully homomorphic encryption
- ▶ Functional encryption
- ▶ General-purpose obfuscation
- ▶ ...

Meanwhile, in the Real World...

A Decade of Lattice Crypto

- ▶ Trapdoor functions and CCA-secure encryption (w/o ROM)
- ▶ Signatures schemes (w/ and w/o ROM)
- ▶ (Hierarchical) identity-based encryption
- ▶ Attribute-based encryption
- ▶ Fully homomorphic encryption
- ▶ Functional encryption
- ▶ General-purpose obfuscation
- ▶ ...

Meanwhile, in the Real World...

- ▶ The vast majority of (public-key) crypto used in practice: **signatures** and **key exchange/transport**, over the Internet.

This Work

- ▶ A first step towards **Internet standards** for lattice cryptography.

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ **AKE** from *any* passively secure KEM (à la IKEv2, RFC 5996)

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient **KEMs** from **ring-LWE** (à la RSA-KEM, RFC 5990)

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new 'reconciliation' mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new ‘reconciliation’ mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.
 - ★ **Bit-for-bit** encryption, plus fixed-size ‘prelude’

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new ‘reconciliation’ mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.
 - ★ Bit-for-bit encryption, plus fixed-size ‘prelude’
 - ★ Improves prior ciphertext sizes by up to **2x**, at essentially **no cost** (in security, runtime, key sizes, etc.)

This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new ‘reconciliation’ mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.
 - ★ Bit-for-bit encryption, plus fixed-size ‘prelude’
 - ★ Improves prior ciphertext sizes by up to 2x, at essentially no cost (in security, runtime, key sizes, etc.)
 - ★ Applies to **all (ring-)LWE-based encryption** schemes

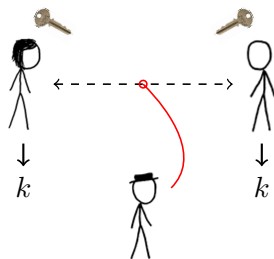
This Work

- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new ‘reconciliation’ mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.
 - ★ Bit-for-bit encryption, plus fixed-size ‘prelude’
 - ★ Improves prior ciphertext sizes by up to 2x, at essentially no cost (in security, runtime, key sizes, etc.)
 - ★ Applies to all (ring-)LWE-based encryption schemes
- ▶ **Not in this work:** parameters, security estimates, implementation

This Work

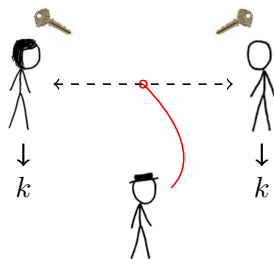
- ▶ A first step towards Internet standards for lattice cryptography.
 - ★ AKE from *any* passively secure KEM (à la IKEv2, RFC 5996)
 - ★ New, efficient KEMs from ring-LWE (à la RSA-KEM, RFC 5990)
- ▶ Technical contribution: a new ‘reconciliation’ mechanism yielding additive (not multiplicative) ciphertext overhead for lattice encryption.
 - ★ Bit-for-bit encryption, plus fixed-size ‘prelude’
 - ★ Improves prior ciphertext sizes by up to 2x, at essentially no cost (in security, runtime, key sizes, etc.)
 - ★ Applies to all (ring-)LWE-based encryption schemes
- ▶ Not in this work: parameters, security estimates, implementation
 - ★ Follow-up [BCNS’14]: TLS/SSL suite (in C) using these components, with estimated > 128 bit security: practical!

Authenticated Key Exchange



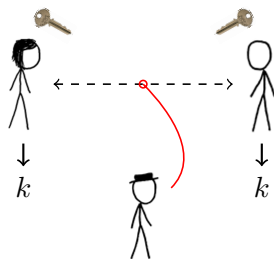
- ▶ **Basic Goal:** mutually authenticate parties and provide them a “consistent view” of completed session (including key k).

Authenticated Key Exchange



- ▶ Basic Goal: mutually authenticate parties and provide them a “consistent view” of completed session (including key k).
- ▶ Adversary controls network, session initiation, may corrupt parties.

Authenticated Key Exchange



- ▶ Basic Goal: mutually authenticate parties and provide them a “consistent view” of completed session (including key k).
- ▶ Adversary controls network, session initiation, may corrupt parties.
- ▶ Many intricate models and definitions, offering strong guarantees. [BR'93, BR'95, Kra'96, BCK'98, Sho'99, CK'01-02, LMQ'03, Kra'05, ...]

Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK'02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities

Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK'02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities
- ▶ A version of Krawczyk’s “SIGn-and-MAC” (SIGMA) protocol [Kra'03] satisfies this security definition, assuming DDH

Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK'02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities
- ▶ A version of Krawczyk’s “SIGn-and-MAC” (SIGMA) protocol [Kra'03] satisfies this security definition, assuming DDH
- ▶ Internet Key Exchange (RFC 5996) based on this model & protocol

Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK'02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities
- ▶ A version of Krawczyk’s “SIGn-and-MAC” (SIGMA) protocol [Kra'03] satisfies this security definition, assuming DDH
- ▶ Internet Key Exchange (RFC 5996) based on this model & protocol

Our Results

- ▶ We generalize SIGMA, replacing its underlying DH mechanism with *any passively secure KEM*.

Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK'02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities
- ▶ A version of Krawczyk’s “SIGn-and-MAC” (SIGMA) protocol [Kra'03] satisfies this security definition, assuming DDH
- ▶ Internet Key Exchange (RFC 5996) based on this model & protocol

Our Results

- ▶ We generalize SIGMA, replacing its underlying DH mechanism with *any passively secure KEM*.
- ▶ This works because:
 - ★ SIGMA has ‘initiator’ who speaks first: can send KEM public key
 - ★ Security proof uses only ‘KEM-like’ properties of DH

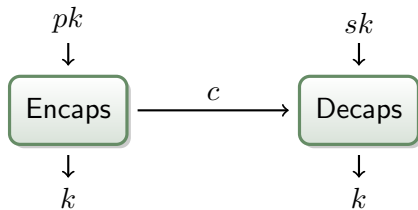
Authenticated Key Exchange

- ▶ We focus on “SK-security with post-specified peer” model [CK’02a]
 - ★ Designed explicitly with Internet in mind
 - ★ ‘Responder’ identity is discovered *during* protocol; can conceal identities
- ▶ A version of Krawczyk’s “SIGn-and-MAC” (SIGMA) protocol [Kra’03] satisfies this security definition, assuming DDH
- ▶ Internet Key Exchange (RFC 5996) based on this model & protocol

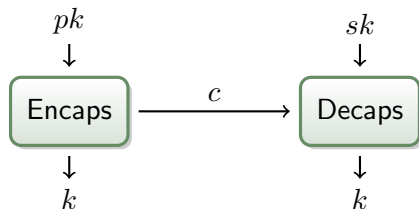
Our Results

- ▶ We generalize SIGMA, replacing its underlying DH mechanism with *any passively secure KEM*.
- ▶ This works because:
 - ★ SIGMA has ‘initiator’ who speaks first: can send KEM public key
 - ★ Security proof uses only ‘KEM-like’ properties of DH
- ▶ Bottom line: minor changes to protocol design should make standardization and implementation easier.

Key Encapsulation/Transport (KEM)

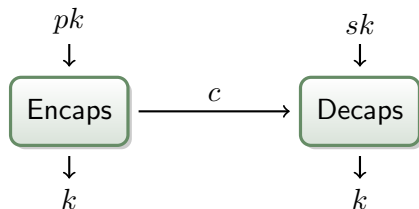


Key Encapsulation/Transport (KEM)



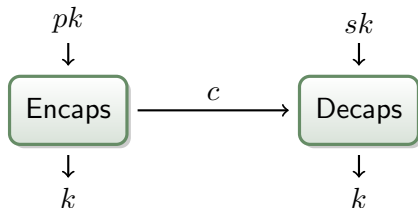
- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.

Key Encapsulation/Transport (KEM)



- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Key Encapsulation/Transport (KEM)

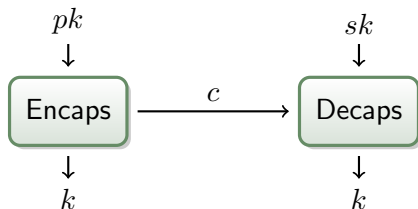


- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Some **practical** actively secure KEMs:

- 1 RSA-OAEP [BR'94, Shoup'01]: security from RSA (in ROM)

Key Encapsulation/Transport (KEM)

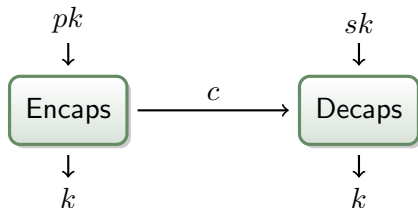


- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Some **practical** actively secure KEMs:

- 1 RSA-OAEP [BR'94, Shoup'01]: security from RSA (in ROM)
- 2 RSA-KEM (RFC 5990): from RSA (in ROM)

Key Encapsulation/Transport (KEM)

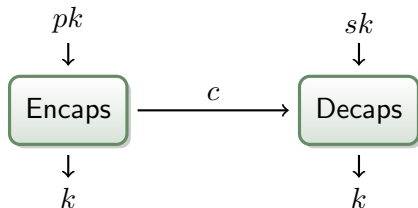


- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Some **practical** actively secure KEMs:

- 1 RSA-OAEP [BR'94, Shoup'01]: security from RSA (in ROM)
- 2 RSA-KEM (RFC 5990): from RSA (in ROM)
- 3 "Hashed ElGamal" [BR'97, CS'98, ABR'01]: from DDH (in ROM)

Key Encapsulation/Transport (KEM)

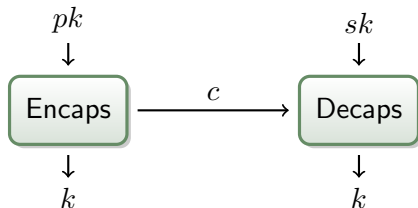


- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Some **practical** actively secure KEMs:

- 1 RSA-OAEP [BR'94, Shoup'01]: security from RSA (in ROM)
- 2 RSA-KEM (RFC 5990): from RSA (in ROM)
- 3 "Hashed ElGamal" [BR'97, CS'98, ABR'01]: from DDH (in ROM)
- 4 REACT transform [OP'01]: from inj. trapdoor function (in ROM)

Key Encapsulation/Transport (KEM)



- ▶ Passive security: $(pk, c, k) \stackrel{c}{\approx} (pk, c, k^*)$ where k^* uniform & independ.
- ▶ Active (CCA) security: same, even with $\text{Decaps}_{sk}(\cdot)$ oracle

Some **practical** actively secure KEMs:

- 1 RSA-OAEP [BR'94, Shoup'01]: security from RSA (in ROM)
- 2 RSA-KEM (RFC 5990): from RSA (in ROM)
- 3 "Hashed ElGamal" [BR'97, CS'98, ABR'01]: from DDH (in ROM)
- 4 REACT transform [OP'01]: from inj. trapdoor function (in ROM)
- 5 FO transforms [FO'99a,b]: from any CPA-secure encryption (in ROM)

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.
- ▶ Most are **inapplicable**, **inefficient**, or **insecure** for our KEM!

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.
- ▶ Most are inapplicable, inefficient, or insecure for our KEM!
 - ✗ "Hashed ElGamal:" **insecure** due to ring-LWE search/decision equiv.

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.
- ▶ Most are inapplicable, inefficient, or insecure for our KEM!
 - ✗ "Hashed ElGamal:" insecure due to ring-LWE search/decision equiv.
 - ✗ REACT: **insecure**: ring-LWE is not OW-PCA, for same reason

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.
- ▶ Most are inapplicable, inefficient, or insecure for our KEM!
 - ✗ "Hashed ElGamal:" insecure due to ring-LWE search/decision equiv.
 - ✗ REACT: insecure: ring-LWE is not OW-PCA, for same reason
 - ✗ OAEP, REACT: **inapplicable**: need inj. trapdoor funcs (Use [MP'12]?)

Key Encapsulation/Transport: Our Results

Passive Security

- ▶ We construct a KEM from ring-LWE (à la [LPR'10,LPR'13]), with a new 'error reconciliation' mechanism: $\approx 2x$ smaller ciphertexts

Active Security

- ▶ There are many construction paradigms, and (semi)generic transformations, for actively secure KEM/encryption schemes.
- ▶ Most are inapplicable, inefficient, or insecure for our KEM!
 - ✗ "Hashed ElGamal:" insecure due to ring-LWE search/decision equiv.
 - ✗ REACT: insecure: ring-LWE is not OW-PCA, for same reason
 - ✗ OAEP, REACT: inapplicable: need inj. trapdoor funcs (Use [MP'12]?)
 - ✓ FO transforms: these work!
 - Prefer [FO'99b] because it maintains plaintext length
 - Subtlety: RO yields random coins for encryption (Gaussian sampling)

New Reconciliation Mechanism

- ▶ In most prior lattice encryption schemes:

New Reconciliation Mechanism

- ▶ In most prior lattice encryption schemes:
 - ★ Sender encodes each msg bit $\mu \in \mathbb{Z}_2 = \{0, 1\}$ as $v = \mu \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.

New Reconciliation Mechanism

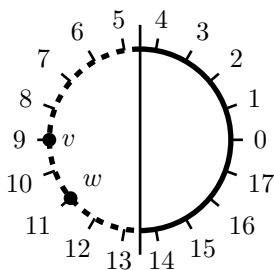
- ▶ In most prior lattice encryption schemes:
 - ★ Sender encodes each msg bit $\mu \in \mathbb{Z}_2 = \{0, 1\}$ as $v = \mu \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.
 - ★ Receiver recovers $w \approx \mu \cdot \lfloor \frac{q}{2} \rfloor$, where \approx comes from LWE error.

New Reconciliation Mechanism

- ▶ In most prior lattice encryption schemes:
 - ★ Sender encodes each msg bit $\mu \in \mathbb{Z}_2 = \{0, 1\}$ as $v = \mu \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.
 - ★ Receiver recovers $w \approx \mu \cdot \lfloor \frac{q}{2} \rfloor$, where \approx comes from LWE error.
 - ★ Receiver computes μ by 'rounding:' $\mu = \lfloor v \rfloor_2 := \lfloor \frac{2}{q} \cdot v \rfloor \in \mathbb{Z}_2$.

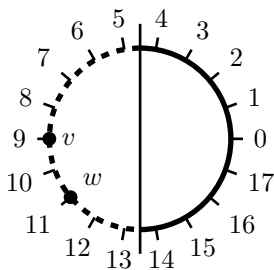
New Reconciliation Mechanism

- ▶ In most prior lattice encryption schemes:
 - ★ Sender encodes each msg bit $\mu \in \mathbb{Z}_2 = \{0, 1\}$ as $v = \mu \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.
 - ★ Receiver recovers $w \approx \mu \cdot \lfloor \frac{q}{2} \rfloor$, where \approx comes from LWE error.
 - ★ Receiver computes μ by 'rounding:' $\mu = \lfloor v \rfloor_2 := \lfloor \frac{2}{q} \cdot v \rfloor \in \mathbb{Z}_2$.



New Reconciliation Mechanism

- ▶ In most prior lattice encryption schemes:
 - ★ Sender encodes each msg bit $\mu \in \mathbb{Z}_2 = \{0, 1\}$ as $v = \mu \cdot \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$.
 - ★ Receiver recovers $w \approx \mu \cdot \lfloor \frac{q}{2} \rfloor$, where \approx comes from LWE error.
 - ★ Receiver computes μ by 'rounding:' $\mu = \lfloor v \rfloor_2 := \lfloor \frac{2}{q} \cdot v \rfloor \in \mathbb{Z}_2$.



- ▶ Problem: $\log q$ factor overhead per msg bit (plus ctext 'preamble').

New Reconciliation Mechanism

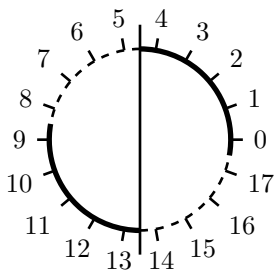
- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.

New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.

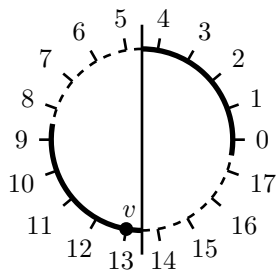
New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.



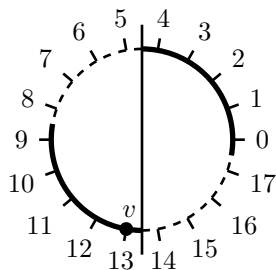
New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.
Encoding of $\lfloor v \rfloor_2$ is $\langle v \rangle_2 \in \{0, 1\}$. (Maybe biased; doesn't matter!)



New Reconciliation Mechanism

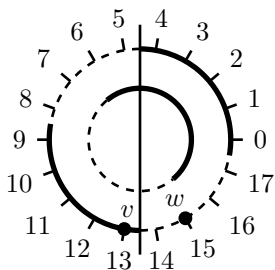
- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.
Encoding of $\lfloor v \rfloor_2$ is $\langle v \rangle_2 \in \{0, 1\}$. (Maybe biased; doesn't matter!)



Claim 1: if $v \in \mathbb{Z}_q$ is uniform, then $\lfloor v \rfloor_2$ is uniform given $\langle v \rangle_2$.

New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.
Encoding of $\lfloor v \rfloor_2$ is $\langle v \rangle_2 \in \{0, 1\}$. (Maybe biased; doesn't matter!)

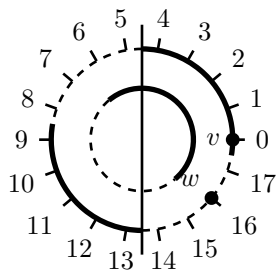


Claim 1: if $v \in \mathbb{Z}_q$ is uniform, then $\lfloor v \rfloor_2$ is uniform given $\langle v \rangle_2$.

Claim 2: given $\langle v \rangle_2$ and any $w \approx v$, can recover $\lfloor v \rfloor_2$.

New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.
Encoding of $\lfloor v \rfloor_2$ is $\langle v \rangle_2 \in \{0, 1\}$. (Maybe biased; doesn't matter!)

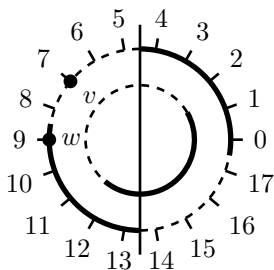


Claim 1: if $v \in \mathbb{Z}_q$ is uniform, then $\lfloor v \rfloor_2$ is uniform given $\langle v \rangle_2$.

Claim 2: given $\langle v \rangle_2$ and any $w \approx v$, can recover $\lfloor v \rfloor_2$.

New Reconciliation Mechanism

- ▶ Contribution: **bit-for-bit** error-tolerant reconciliation.
- ▶ Sender has (pseudo)random $v \in \mathbb{Z}_q$, receiver can recover $w \approx v$.
- ▶ For even q , define 'cross-rounding' function $\langle v \rangle_2 := \lfloor \frac{4}{q} \cdot v \rfloor \bmod 2$.
Encoding of $\lfloor v \rfloor_2$ is $\langle v \rangle_2 \in \{0, 1\}$. (Maybe biased; doesn't matter!)



Claim 1: if $v \in \mathbb{Z}_q$ is uniform, then $\lfloor v \rfloor_2$ is uniform given $\langle v \rangle_2$.

Claim 2: given $\langle v \rangle_2$ and any $w \approx v$, can recover $\lfloor v \rfloor_2$.

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are **deg $< n$ polynomials** with **integer (mod q) coeffs.**
 - ★ 'Errors' in R are polynomials with **small** (Gaussian) integer coefficients.
 - ★ Operations in R_q are **very efficient** using FFT-like algorithms.

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are $\deg < n$ polynomials with integer (mod q) coeffs.
 - ★ 'Errors' in R are polynomials with small (Gaussian) integer coefficients.
 - ★ Operations in R_q are very efficient using FFT-like algorithms.
- ▶ Public key is $(a, b \approx a \cdot s) \in R_q \times R_q$ for secret error $s \in R$.

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are $\deg < n$ polynomials with integer (mod q) coeffs.
 - ★ 'Errors' in R are polynomials with small (Gaussian) integer coefficients.
 - ★ Operations in R_q are very efficient using FFT-like algorithms.
- ▶ Public key is $(a, b \approx a \cdot s) \in R_q \times R_q$ for secret error $s \in R$.
- ▶ Encaps: choose error $r \in R$, let $u \approx r \cdot a$, $v \approx r \cdot b \in R_q$, output

$$c = (u, \langle v \rangle_2) \in R_q \times R_2, \quad \text{key } k = \lfloor v \rfloor_2 \in R_2.$$

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are $\deg < n$ polynomials with integer (mod q) coeffs.
 - ★ 'Errors' in R are polynomials with small (Gaussian) integer coefficients.
 - ★ Operations in R_q are very efficient using FFT-like algorithms.
- ▶ Public key is $(a, b \approx a \cdot s) \in R_q \times R_q$ for secret error $s \in R$.

- ▶ Encaps: choose error $r \in R$, let $u \approx r \cdot a$, $v \approx r \cdot b \in R_q$, output

$$c = (u, \langle v \rangle_2) \in R_q \times R_2, \quad \text{key } k = \lfloor v \rfloor_2 \in R_2.$$

- ▶ Decaps: recover $k = \lfloor v \rfloor_2$ from $\langle v \rangle_2$ and

$$w = u \cdot s \approx r \cdot a \cdot s \approx r \cdot b \approx v.$$

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are $\deg < n$ polynomials with integer (mod q) coeffs.
 - ★ 'Errors' in R are polynomials with small (Gaussian) integer coefficients.
 - ★ Operations in R_q are very efficient using FFT-like algorithms.
- ▶ Public key is $(a, b \approx a \cdot s) \in R_q \times R_q$ for secret error $s \in R$.

- ▶ Encaps: choose error $r \in R$, let $u \approx r \cdot a$, $v \approx r \cdot b \in R_q$, output

$$c = (u, \langle v \rangle_2) \in R_q \times R_2, \quad \text{key } k = \lfloor v \rfloor_2 \in R_2.$$

- ▶ Decaps: recover $k = \lfloor v \rfloor_2$ from $\langle v \rangle_2$ and

$$w = u \cdot s \approx r \cdot a \cdot s \approx r \cdot b \approx v.$$

- ▶ Passively secure under ring-LWE [LPR'10].

The Ring-LWE KEM

- ▶ Let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of two, and $R_q = R/qR$
 - ★ Elts of R (R_q) are $\deg < n$ polynomials with integer (mod q) coeffs.
 - ★ 'Errors' in R are polynomials with small (Gaussian) integer coefficients.
 - ★ Operations in R_q are very efficient using FFT-like algorithms.
- ▶ Public key is $(a, b \approx a \cdot s) \in R_q \times R_q$ for secret error $s \in R$.

- ▶ Encaps: choose error $r \in R$, let $u \approx r \cdot a$, $v \approx r \cdot b \in R_q$, output

$$c = (u, \langle v \rangle_2) \in R_q \times R_2, \quad \text{key } k = \lfloor v \rfloor_2 \in R_2.$$

- ▶ Decaps: recover $k = \lfloor v \rfloor_2$ from $\langle v \rangle_2$ and

$$w = u \cdot s \approx r \cdot a \cdot s \approx r \cdot b \approx v.$$

- ▶ Passively secure under ring-LWE [LPR'10].
- ▶ Generalizes (tightly!) to any cyclotomic ring using tools from [LPR'13].

Conclusions and Future Work

- ▶ We have taken one (small) step toward “making the Internet safe for lattice cryptography” (or vice versa?)

Conclusions and Future Work

- ▶ We have taken one (small) step toward “making the Internet safe for lattice cryptography” (or vice versa?)
- ▶ Much remains to be done: signature protocols, implementations, formal standards, . . .

Conclusions and Future Work

- ▶ We have taken one (small) step toward “making the Internet safe for lattice cryptography” (or vice versa?)
- ▶ Much remains to be done: signature protocols, implementations, formal standards, . . .
- ▶ Please help!

Conclusions and Future Work

- ▶ We have taken one (small) step toward “making the Internet safe for lattice cryptography” (or vice versa?)
- ▶ Much remains to be done: signature protocols, implementations, formal standards, . . .
- ▶ Please help!

Thanks!