The Design of Lattice-based Key-Encapsulation Mechanisms

Mark Schultz

University of California San Diego

27 May 2021

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions
  - Symmetric: The same key is used to encrypt and decrypt.

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions
  - Symmetric: The same key is used to encrypt and decrypt.
  - Asymmetric: Different keys are used to encrypt and decrypt.

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions
  - Symmetric: The same key is used to encrypt and decrypt.
  - Asymmetric: Different keys are used to encrypt and decrypt.
    - Can make the encryption key public.

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions
  - Symmetric: The same key is used to encrypt and decrypt.
  - Asymmetric: Different keys are used to encrypt and decrypt.
    - Can make the encryption key public.
- Symmetric constructions are *much* faster.

What is Key Exchange?

- Cryptography is divided into symmetric and asymmetric constructions
  - Symmetric: The same key is used to encrypt and decrypt.
  - Asymmetric: Different keys are used to encrypt and decrypt.
    - Can make the encryption key public.
- Symmetric constructions are *much* faster.
- Exchange symmetric keys, and then use symmetric primitives.

The Syntax of Key Exchange

- Two main paradigms:

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.
- Encoding:

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.
- Encoding:
  - PKE: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, k)$

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.
- Encoding:
  - PKE: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, k)$
  - KEM: $(c, k) \leftarrow \mathsf{Encaps}(\mathsf{pk})$

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.
- Encoding:
  - PKE: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, k)$
  - KEM: $(c, k) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
- Decoding:

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathrm{pk}, \mathrm{sk})$.
- Encoding:
  - PKE: $c \leftarrow \mathrm{Enc}(\mathrm{pk}, k)$
  - KEM: $(c, k) \leftarrow \mathrm{Encaps}(\mathrm{pk})$
- Decoding:
  - PKE: $k \leftarrow \mathrm{Dec}(\mathrm{sk}, c)$

The Syntax of Key Exchange

- Two main paradigms:
  - from Public-Key Encryption (PKE)
  - from Key-Encapsulation Mechanisms (KEM)
- Both: Key Generation algorithm that outputs $(\mathsf{pk}, \mathsf{sk})$.
- Encoding:
  - PKE: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, k)$
  - KEM: $(c, k) \leftarrow \mathsf{Encaps}(\mathsf{pk})$
- Decoding:
  - PKE: $k \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
  - KEM: $k \leftarrow \mathsf{Decaps}(\mathsf{sk}, c)$

Properties of Key Exchange

- Correctness: The sender and reciever end up with the same key k.

Properties of Key Exchange

- Correctness: The sender and reciever end up with the same key k.
  - Lattice-based schemes not always correct, modeling this is non-trivial [HHK17].

Properties of Key Exchange

- Correctness: The sender and reciever end up with the same key k.
  - Lattice-based schemes not always correct, modeling this is non-trivial [HHK17].
- (Passive) Security: The transcript (pk, c) reveal no information about the key k

Properties of Key Exchange

- Correctness: The sender and reciever end up with the same key k.
  - Lattice-based schemes not always correct, modeling this is non-trivial [HHK17].
- (Passive) Security: The transcript (pk, c) reveal no information about the key k
  - Standard definitions (IND-CPA security) in cryptography

Arithmetic of Lattice-based Cryptography

- All constructions can be viewed as "Linear Algebra over R" [Boo+20].

Arithmetic of Lattice-based Cryptography

- All constructions can be viewed as "Linear Algebra over R" [Boo+20].
  - $R \cong \mathbb{Z}_q[x]/(x^{2^k}+1)$ is a "quotient of a polynomial ring"

Arithmetic of Lattice-based Cryptography

- All constructions can be viewed as "Linear Algebra over R" [Boo+20].
  - $R \cong \mathbb{Z}_q[x]/(x^{2^k}+1)$ is a "quotient of a polynomial ring"
- For $k = 0$ and q prime, $R \cong \mathbb{F}_q$.

Arithmetic of Lattice-based Cryptography

- All constructions can be viewed as "Linear Algebra over R" [Boo+20].
  - $R \cong \mathbb{Z}_q[x]/(x^{2^k}+1)$ is a "quotient of a polynomial ring"
- For $k = 0$ and q prime, $R \cong \mathbb{F}_q$.
- Will be rather lax with which results are proved for LWE vs RLWE

Arithmetic of Lattice-based Cryptography

- All constructions can be viewed as "Linear Algebra over R" [Boo+20].
  - $R \cong \mathbb{Z}_q[x]/(x^{2^k}+1)$ is a "quotient of a polynomial ring"
- For $k = 0$ and q prime, $R \cong \mathbb{F}_q$.
- Will be rather lax with which results are proved for LWE vs RLWE vs MLWE vs LWR vs RLWR vs MLWR

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm*

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm*
- Define a code with fundamental domain $\mathcal{E}$ as a pair of functions

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm*
- Define a code with fundamental domain $\mathcal{E}$ as a pair of functions

$$\text{encode} : A \to B, \qquad \text{decode} : B \to A$$

that satisfies the properties

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm*
- Define a code with fundamental domain $\mathcal{E}$ as a pair of functions

$$\text{encode} : A \to B, \qquad \text{decode} : B \to A$$

that satisfies the properties

  - Error-Correction: $\text{decode}(\text{encode}(m) + \mathcal{E}) = m$.

Codes — Removing and Introducing Noise

- Need some abstract notion of small elements.
  - In lattice-based cryptography, ℓ_p-norm*
- Define a code with fundamental domain $\mathcal{E}$ as a pair of functions

$$\text{encode} : A \to B, \qquad \text{decode} : B \to A$$

  that satisfies the properties
  - Error-Correction: $\text{decode}(\text{encode}(m) + \mathcal{E}) = m$.
  - Lossy Compression: $\text{encode}(\text{decode}(x)) \in x + \mathcal{E}$.

Noisy Lattice-based DH

- $\mathrm{A} \in R^{n \times n}$ a *matrix*,

Noisy Lattice-based DH

- $\mathsf{A} \in R^{n\times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathsf{A}\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t\mathsf{A} + \vec{e}_r)$,

Noisy Lattice-based DH

- $\mathsf{A} \in R^{n \times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathsf{A}\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t\mathsf{A} + \vec{e}_r)$, where $\vec{e}_s, \vec{e}_r \leftarrow \chi^n$ are small.

Noisy Lattice-based DH

- $\mathrm{A} \in R^{n\times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathrm{A}\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t\mathrm{A} + \vec{e}_r)$, where $\vec{e}_s, \vec{e}_r \leftarrow \chi^n$ are small.
  1. Exchange $\mathrm{A}\vec{s} + \vec{e}_s$ and $\vec{r}^t\mathrm{A} + \vec{e}_r$

Noisy Lattice-based DH

- $\mathrm{A} \in R^{n\times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathrm{A}\vec{s}+\vec{e_s}), (\vec{r}, \vec{r}^t\mathrm{A}+\vec{e_r})$, where $\vec{e_s}, \vec{e_r} \leftarrow \chi^n$ are small.
  1. Exchange $\mathrm{A}\vec{s}+\vec{e_s}$ and $\vec{r}^t\mathrm{A}+\vec{e_r}$
  2. Compute $(\vec{r}^t\mathrm{A}+\vec{e_r})\vec{s} \approx \vec{r}^t\mathrm{A}\vec{s} \approx \vec{r}(\mathrm{A}\vec{s}+\vec{e_s})$

Noisy Lattice-based DH

- $\mathsf{A} \in R^{n\times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathsf{A}\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t\mathsf{A} + \vec{e}_r)$, where $\vec{e}_s, \vec{e}_r \leftarrow \chi^n$ are small.
  1. Exchange $\mathsf{A}\vec{s} + \vec{e}_s$ and $\vec{r}^t\mathsf{A} + \vec{e}_r$
  2. Compute $(\vec{r}^t\mathsf{A} + \vec{e}_r)\vec{s} \approx \vec{r}^t\mathsf{A}\vec{s} \approx \vec{r}(\mathsf{A}\vec{s} + \vec{e}_s)$
  3. Difference $\langle \vec{e}_r, \vec{s}\rangle - \langle \vec{r}, \vec{e}_s\rangle$ should be small.

Noisy Lattice-based DH

- $\mathrm{A} \in R^{n\times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, \mathrm{A}\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t\mathrm{A} + \vec{e}_r)$, where $\vec{e}_s, \vec{e}_r \leftarrow \chi^n$ are small.
  1. Exchange $\mathrm{A}\vec{s} + \vec{e}_s$ and $\vec{r}^t\mathrm{A} + \vec{e}_r$
  2. Compute $(\vec{r}^t\mathrm{A} + \vec{e}_r)\vec{s} \approx \vec{r}^t\mathrm{A}\vec{s} \approx \vec{r}(\mathrm{A}\vec{s} + \vec{e}_s)$
  3. Difference $\langle \vec{e}_r, \vec{s}\rangle - \langle \vec{r}, \vec{e}_s\rangle$ should be small.
- Must be *hard* to compute $\vec{r}^t\mathrm{A}\vec{s}$ from $(\vec{r}^t\mathrm{A} + \vec{e}_r, \mathrm{A}\vec{s} + \vec{e}_s)$.

Noisy Lattice-based DH

- $A \in R^{n \times n}$ a *matrix*,
- Two users with LWE Shares $(\vec{s}, A\vec{s} + \vec{e}_s), (\vec{r}, \vec{r}^t A + \vec{e}_r)$, where $\vec{e}_s, \vec{e}_r \leftarrow \chi^n$ are small.
  1. Exchange $A\vec{s} + \vec{e}_s$ and $\vec{r}^t A + \vec{e}_r$
  2. Compute $(\vec{r}^t A + \vec{e}_r)\vec{s} \approx \vec{r}^t A\vec{s} \approx \vec{r}(A\vec{s} + \vec{e}_s)$
  3. Difference $\langle \vec{e}_r, \vec{s} \rangle - \langle \vec{r}, \vec{e}_s \rangle$ should be small.
- Must be *hard* to compute $\vec{r}^t A\vec{s}$ from $(\vec{r}^t A + \vec{e}_r, A\vec{s} + \vec{e}_s)$.
- Secure, but only *noisy* key agreement.

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathsf{Enc}(\cdot,\cdot)$ and $\mathsf{Encaps}(\cdot,\cdot)$ are identical.

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathrm{Enc}(\cdot,\cdot)$ and $\mathrm{Encaps}(\cdot,\cdot)$ are identical.
- Both parametrized by the choice of code $(\mathrm{encode}, \mathrm{decode})$.

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathsf{Enc}(\cdot,\cdot)$ and $\mathsf{Encaps}(\cdot,\cdot)$ are identical.
- Both parametrized by the choice of code $(\mathsf{encode}, \mathsf{decode})$.

KeyGen(1^n)
$\mathsf{A} \leftarrow R^{n\times n}$
$\mathsf{sk} \leftarrow R^n$
$\vec{e}_{\mathsf{sk}} \leftarrow \chi^n$
$\mathsf{pk} \leftarrow (\mathsf{A}, \mathsf{A}\mathsf{sk} + \vec{e}_{\mathsf{sk}})$
return $(\mathsf{pk}, \mathsf{sk})$

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathrm{Enc}(\cdot,\cdot)$ and $\mathrm{Encaps}(\cdot,\cdot)$ are identical.
- Both parametrized by the choice of code $(\mathrm{encode}, \mathrm{decode})$.

KeyGen(1^n)	**E**(pk)
	$(\mathsf{A}, \vec{b}) = \mathsf{pk}$
$\mathsf{A} \leftarrow R^{n\times n}$	$\vec{r} \leftarrow R^n$
$\mathsf{sk} \leftarrow R^n$	$\vec{e}_r \leftarrow \chi^n$
$\vec{e}_{\mathsf{sk}} \leftarrow \chi^n$	$\vec{u} \leftarrow \vec{r}\mathsf{A} + \vec{e}_r$
$\mathsf{pk} \leftarrow (\mathsf{A}, \mathsf{A}\mathsf{sk} + \vec{e}_{\mathsf{sk}})$	$e \leftarrow \chi$
return $(\mathsf{pk}, \mathsf{sk})$	$v \leftarrow \vec{r}^t\vec{b} + e$
	return $(\vec{u}, v)$

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathrm{Enc}(\cdot,\cdot)$ and $\mathrm{Encaps}(\cdot,\cdot)$ are identical.
- Both parametrized by the choice of code (encode, decode).

	$\mathbf{E}(\mathsf{pk})$	
$\mathbf{KeyGen}(1^n)$	$(\mathsf{A}, \vec{b}) = \mathsf{pk}$	$\mathbf{D}(\mathsf{sk}, (\vec{u}, h))$
$\mathsf{A} \leftarrow R^{n\times n}$	$\vec{r} \leftarrow R^n$	$\mathrm{decode}(\vec{u}^t \vec{\mathsf{sk}} - h)$
$\mathsf{sk} \leftarrow R^n$	$\vec{e}_r \leftarrow \chi^n$	
$\vec{e}_{\mathsf{sk}} \leftarrow \chi^n$	$\vec{u} \leftarrow \vec{r}\mathsf{A} + \vec{e}_r$	
$\mathsf{pk} \leftarrow (\mathsf{A}, \mathsf{A}\mathsf{sk} + \vec{e}_{\mathsf{sk}})$	$e \leftarrow \chi$	
return $(\mathsf{pk}, \mathsf{sk})$	$v \leftarrow \vec{r}^t \vec{b} + e$	
	return $(\vec{u}, v)$	

Key Exchange Construction

- Two main paradigms [LPR10; JZ17], both have the same core.
  - In this presentation, everything but $\mathrm{Enc}(\cdot,\cdot)$ and $\mathrm{Encaps}(\cdot,\cdot)$ are identical.
- Both parametrized by the choice of code $(\mathrm{encode}, \mathrm{decode})$.

KeyGen(1^n)
$\mathsf{A} \leftarrow R^{n\times n}$
$\mathrm{sk} \leftarrow R^n$
$\vec{e}_{\mathrm{sk}} \leftarrow \chi^n$
$\mathrm{pk} \leftarrow (\mathsf{A}, \mathsf{A}\mathrm{sk} + \vec{e}_{\mathrm{sk}})$
return $(\mathrm{pk}, \mathrm{sk})$

E(pk)
$(\mathsf{A}, \vec{b}) = \mathrm{pk}$
$\vec{r} \leftarrow R^n$
$\vec{e}_r \leftarrow \chi^n$
$\vec{u} \leftarrow \vec{r}\mathsf{A} + \vec{e}_r$
$e \leftarrow \chi$
$v \leftarrow \vec{r}^t\vec{b} + e$
return $(\vec{u}, v)$

D$(\mathrm{sk}, (\vec{u}, h))$
$\mathrm{decode}(\vec{u}^t\vec{\mathrm{sk}} - h)$

- h is a hint, that differs in the PKE and KEM constructions.

PKE Construction [LPR10]

- PKE: The hint h is constructed as:

PKE Construction [LPR10]

- PKE: The hint h is constructed as:
  $\mathrm{Enc}(\mathrm{pk}, k)$
  $(\vec{u}, v) \leftarrow \mathbf{E}(\mathrm{pk})$
  return $(\vec{u}, v - \mathrm{encode}(k))$

PKE Construction [LPR10]

- PKE: The hint h is constructed as:
  $\mathrm{Enc}(\mathrm{pk}, k)$
  $(\vec{u}, v) \leftarrow \mathbf{E}(\mathrm{pk})$
  return $(\vec{u}, v - \mathrm{encode}(k))$
- Correctness: $\vec{u}^t \mathrm{sk} - h \approx \vec{r}^t A\vec{s} - v + \mathrm{encode}(k) \approx \mathrm{encode}(k)$.

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.
  - It encodes a random key under a code.

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.
  - It encodes a random key under a code.
  - Can we instead extract a shared key from the key we *approximately* agree on?

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.
  - It encodes a random key under a code.
  - Can we instead extract a shared key from the key we *approximately* agree on?
- For reconciliation, the hint h is constructed as:

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.
  - It encodes a random key under a code.
  - Can we instead extract a shared key from the key we *approximately* agree on?
- For reconciliation, the hint h is constructed as:

  Encaps(pk)
  $(\vec{u}, v) \leftarrow \mathbf{E}(\mathrm{pk})$
  $h \leftarrow v - \mathrm{encode}(\mathrm{decode}(v))$
  $k \leftarrow \mathrm{decode}(v)$
  return $(\vec{u}, h)$

Reconciliation Construction, [Din12; Pei14; Alk+16; JZ17]

- PKE was wasteful.
  - It encodes a random key under a code.
  - Can we instead extract a shared key from the key we *approximately* agree on?
- For reconciliation, the hint h is constructed as:
  Encaps(pk)
  $(\vec{u}, v) \leftarrow \mathbf{E}(\mathrm{pk})$
  $h \leftarrow v - \mathrm{encode}(\mathrm{decode}(v))$
  $k \leftarrow \mathrm{decode}(v)$
  return $(\vec{u}, h)$
- The hint h can be seen as "centering" $u^t\mathrm{sk} \approx \vec{r}^t A\vec{s}$ in the fundamental domain of the code.

Key Differences

- Very similar constructions, up to the computations of the hints.

$$v \mapsto v - \mathrm{encode}(k), \qquad v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

Key Differences

- Very similar constructions, up to the computations of the hints.

$$v \mapsto v - \mathrm{encode}(k), \qquad v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- Two large differences:
  - Can only "choose" which key to use in the PKE scheme.

Key Differences

- Very similar constructions, up to the computations of the hints.

$$v \mapsto v - \mathrm{encode}(k), \qquad v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- Two large differences:
  - Can only "choose" which key to use in the PKE scheme.
  - The KEM scheme has its hint contained in the code's fundamental domain, which can be quite small.

Key Differences

- Very similar constructions, up to the computations of the hints.

$$v \mapsto v - \text{encode}(k), \qquad v \mapsto v - \text{encode}(\text{decode}(v))$$

- Two large differences:
  - Can only "choose" which key to use in the PKE scheme.
  - The KEM scheme has its hint contained in the code's fundamental domain, which can be quite small.
    - Under ideal circumstances $\{0,1\}^{2^k} \subseteq R \cong \mathbb{Z}_q^{2^k}$, e.g. a reduction in the size of the hint by $\log_2 q \approx 10$ factor.

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security
  - Active — adversary can submit ciphertexts for decryption, and observe behavior

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security
  - Active — adversary can submit ciphertexts for decryption, and observe behavior
- Randomness used in encryption is computed from the message ($r = G(k)$ for a PRG G)

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security
  - Active — adversary can submit ciphertexts for decryption, and observe behavior
- Randomness used in encryption is computed from the message ($r = G(k)$ for a PRG G)
- Later "checks" this upon decryption by reencrypting.

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security
  - Active — adversary can submit ciphertexts for decryption, and observe behavior
- Randomness used in encryption is computed from the message ($r = G(k)$ for a PRG G)
- Later "checks" this upon decryption by reencrypting.
- Some issues with incorrect schemes, but can be resolved.

FO Transform, [FO99; HHK17]

- Transforms from passive security to active security
  - Active — adversary can submit ciphertexts for decryption, and observe behavior
- Randomness used in encryption is computed from the message ($r = G(k)$ for a PRG G)
- Later "checks" this upon decryption by reencrypting.
- Some issues with incorrect schemes, but can be resolved.
- Unknown how to apply to KEMs, and therefore reconciliation-based schemes.

Compression of A

- Can send a short n-bit seed rather than an $2^k n^2 \log_2 q$-bit matrix.

Compression of A

- Can send a short n-bit seed rather than an $2^k n^2 \log_2 q$-bit matrix.
  - Some theoretical applications, needed for tight rate bounds.

Compression of $A\vec{s} + \vec{e_s}$?

- Can't just send a short seed.

Compression of $A\vec{s} + \vec{e_s}$?

- Can't just send a short seed.
- Practically [Ava+; DAn+18], people round off the lowest coefficients, which are mostly noise.

Compression of $A\vec{s} + \vec{e_s}$?

- Can't just send a short seed.
- Practically [Ava+; DAn+18], people round off the lowest coefficients, which are mostly noise.
  - Can be viewed as lossy compression via a particular code.

Compression of $A\vec{s} + \vec{e_s}$?

- Can't just send a short seed.
- Practically [Ava+; DAn+18], people round off the lowest coefficients, which are mostly noise.
  - Can be viewed as lossy compression via a particular code.
  - Can use other codes [Bra+19].

Compression of $A\vec{s} + \vec{e}_s$?

- Can't just send a short seed.
- Practically [Ava+; DAn+18], people round off the lowest coefficients, which are mostly noise.
  - Can be viewed as lossy compression via a particular code.
  - Can use other codes [Bra+19].
    - Using other codes is provably better.

Compression of $A\vec{s} + \vec{e_s}$?

- Can't just send a short seed.
- Practically [Ava+; DAn+18], people round off the lowest coefficients, which are mostly noise.
  - Can be viewed as lossy compression via a particular code.
  - Can use other codes [Bra+19].
    - Using other codes is provably better.
  - [Ava+] even argues the additional rounding error has security benefits.

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k} + 1)$, optimal multiplication requires q to have special structure.

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k} + 1)$, optimal multiplication requires q to have special structure.
    - View $R \subseteq [-q/2, q/2)^{2^k} \subseteq \mathbb{Z}[x]$

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k}+1)$, optimal multiplication requires q to have special structure.
    - View $R \subseteq [-q/2, q/2)^{2^k} \subseteq \mathbb{Z}[x]$
    - Bound the size of the coefficients of the product of $f(x), g(x)$ from this set.

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k}+1)$, optimal multiplication requires q to have special structure.
  - View $R \subseteq [-q/2, q/2)^{2^k} \subseteq \mathbb{Z}[x]$
  - Bound the size of the coefficients of the product of $f(x), g(x)$ from this set.
  - Find a q' with special structure that exceeds this bound, multiply in $\mathbb{Z}_{q'}[x]/(x^{2^k}+1)$.

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k}+1)$, optimal multiplication requires q to have special structure.
  - View $R \subseteq [-q/2, q/2)^{2^k} \subseteq \mathbb{Z}[x]$
  - Bound the size of the coefficients of the product of $f(x), g(x)$ from this set.
  - Find a q' with special structure that exceeds this bound, multiply in $\mathbb{Z}_{q'}[x]/(x^{2^k}+1)$.
- Works surprisingly well, $\approx 20\%$ speedup in Saber.

Speeding up NTTs [Chu+20]

- For $R = \mathbb{Z}_q[x]/(x^{2^k}+1)$, optimal multiplication requires q to have special structure.
  - View $R \subseteq [-q/2, q/2)^{2^k} \subseteq \mathbb{Z}[x]$
  - Bound the size of the coefficients of the product of $f(x), g(x)$ from this set.
  - Find a q' with special structure that exceeds this bound, multiply in $\mathbb{Z}_{q'}[x]/(x^{2^k}+1)$.
- Works surprisingly well, $\approx 20\%$ speedup in Saber.
- Uses rather naive bounds — can they be sharpened at all?

Wyner-Ziv Reconciliation [SLL20]

- Recall that the hint in the Reconciliation-based KEM was constructed as

$$v \mapsto v - \text{encode}(\text{decode}(v))$$

Wyner-Ziv Reconciliation [SLL20]

- Recall that the hint in the Reconciliation-based KEM was constructed as

$$v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- This construction first pre-processes v by compressing it with another code.

Wyner-Ziv Reconciliation [SLL20]

- Recall that the hint in the Reconciliation-based KEM was constructed as

$$v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- This construction first pre-processes v by compressing it with another code.
  - (There are some other minor differences)

Wyner-Ziv Reconciliation [SLL20]

- Recall that the hint in the Reconciliation-based KEM was constructed as

$$v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- This construction first pre-processes v by compressing it with another code.
  - (There are some other minor differences)
- For PKE you can show that two codes are provably better than one.

Wyner-Ziv Reconciliation [SLL20]

- Recall that the hint in the Reconciliation-based KEM was constructed as

$$v \mapsto v - \mathrm{encode}(\mathrm{decode}(v))$$

- This construction first pre-processes v by compressing it with another code.
  - (There are some other minor differences)
- For PKE you can show that two codes are provably better than one.
  - Show in the KEM setting as well?

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a “hintless” KEM.
  - Could be used to remove the interaction in LWE-based key exchange.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
  - Could be used to remove the interaction in LWE-based key exchange.
    - (One would have to publicly post A)

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
  - Could be used to remove the interaction in LWE-based key exchange.
    - (One would have to publicly post A)
- Reduces problem to post-processing local view of $\vec{r}^t A \vec{r}$.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
    - Could be used to remove the interaction in LWE-based key exchange.
        - (One would have to publicly post A)
- Reduces problem to post-processing local view of $\vec{r}^t A\vec{r}$.
    - And other variables that are locally known.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
  - Could be used to remove the interaction in LWE-based key exchange.
    - (One would have to publicly post A)
- Reduces problem to post-processing local view of $\vec{r}^t A \vec{r}$.
  - And other variables that are locally known.
  - Gives strong bounds against some restricted classes of functions.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
  - Could be used to remove the interaction in LWE-based key exchange.
    - (One would have to publicly post A)
- Reduces problem to post-processing local view of $\vec{r}^t A \vec{r}$.
  - And other variables that are locally known.
  - Gives strong bounds against some restricted classes of functions.
  - Shows that *any* function would imply a weak-PRF.

(No) NIKE from LWE [Guo+20]

- Investigates the possibility of a "hintless" KEM.
  - Could be used to remove the interaction in LWE-based key exchange.
    - (One would have to publicly post A)
- Reduces problem to post-processing local view of $\vec{r}^t A \vec{r}$.
  - And other variables that are locally known.
  - Gives strong bounds against some restricted classes of functions.
  - Shows that *any* function would imply a weak-PRF.
    - Only recently constructed (for parameters of interest) [Kim20].

Binary Error-Correction

- If the code (encode, decode) is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.

Binary Error-Correction

- If the code $(\text{encode}, \text{decode})$ is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.
  - Pre-process k with a Binary error-correcting code?

Binary Error-Correction

- If the code (encode, decode) is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.
  - Pre-process k with a Binary error-correcting code?
- Works if bit-flips are independent.

Binary Error-Correction

- If the code $(\text{encode}, \text{decode})$ is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.
  - Pre-process k with a Binary error-correcting code?
- Works if bit-flips are independent.
  - Shown asymptotically [JZ17].

Binary Error-Correction

- If the code $(\mathsf{encode}, \mathsf{decode})$ is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.
  - Pre-process k with a Binary error-correcting code?
- Works if bit-flips are independent.
  - Shown asymptotically [JZ17].
- Independence heuristic has issues [DVV19].

Binary Error-Correction

- If the code (encode, decode) is applied "coordinate-wise", failed decoding leads to a bit-flip in the key k.
  - Pre-process k with a Binary error-correcting code?
- Works if bit-flips are independent.
  - Shown asymptotically [JZ17].
- Independence heuristic has issues [DVV19].
  - Up to 2^{48} difference in failure probabillity estimations.

Non-trivial Codes

- Most codes used are fairly simple:

Non-trivial Codes

- Most codes used are fairly simple:

$$\mathrm{encode}(m) = (q/2)m \bmod q, \qquad \mathrm{decode}(x) = \lfloor (2/q)x \rceil \bmod 2$$

Non-trivial Codes

- Most codes used are fairly simple:

$$\mathrm{encode}(m) = (q/2)m \bmod q, \qquad \mathrm{decode}(x) = \lfloor (2/q)x \rceil \bmod 2$$

- Known in coding theory that high-dimensional codes are preferrable.

Non-trivial Codes

- Most codes used are fairly simple:

$$\text{encode}(m) = (q/2)m \bmod q, \qquad \text{decode}(x) = \lfloor (2/q)x \rceil \bmod 2$$

- Known in coding theory that high-dimensional codes are preferrable.
  - Some benefits in crypto [Pop16; JZ20].
  - Second code of [Bra+19] is high-dimensional.

Reconciliation for Encryption?

- (One view of) reconciliation is that it compresses $h \in R \cong (\mathbb{Z}_q^{2^k})$ to $\{0,1\}^{2^k}$.

Reconciliation for Encryption?

- (One view of) reconciliation is that it compresses $h \in R \cong (\mathbb{Z}_q^{2^k})$ to $\{0,1\}^{2^k}$.
- The PKE of [Bra+19] compresses $h \in R \to \mathbb{Z}_q \times \{0,1\}^{2^k}$.

Reconciliation for Encryption?

- (One view of) reconciliation is that it compresses $h \in R \cong (\mathbb{Z}_q^{2^k})$ to $\{0,1\}^{2^k}$.
- The PKE of [Bra+19] compresses $h \in R \to \mathbb{Z}_q \times \{0,1\}^{2^k}$.
  - PKE so FO-transform still works.

Reconciliation for Encryption?

- (One view of) reconciliation is that it compresses $h \in R \cong (\mathbb{Z}_q^{2^k})$ to $\{0,1\}^{2^k}$.
- The PKE of [Bra+19] compresses $h \in R \to \mathbb{Z}_q \times \{0,1\}^{2^k}$.
  - PKE so FO-transform still works.
  - How similar are PKE and KEM constructions overall?

- Many exciting developments in lattice-based KEMs.

- Many exciting developments in lattice-based KEMs.
  - Especially with the use of non-trivial codes.

- Many exciting developments in lattice-based KEMs.
  - Especially with the use of non-trivial codes.
- Potential for the unification and clarification of different techniques.

- Many exciting developments in lattice-based KEMs.
  - Especially with the use of non-trivial codes.
- Potential for the unification and clarification of different techniques.
  - And different sub-fields [Boo+20].

- Many exciting developments in lattice-based KEMs.
    - Especially with the use of non-trivial codes.
- Potential for the unification and clarification of different techniques.
    - And different sub-fields [Boo+20].
- Thanks!!

Erdem Alkim et al. "Post-quantum Key Exchange - A New Hope". In: *USENIX Security 2016: 25th USENIX Security Symposium*. Ed. by Thorsten Holz and Stefan Savage. Austin, TX, USA: USENIX Association, Aug. 2016, pp. 327–343.

Roberto Avanzi et al. "Algorithm Specifications And Supporting Documentation". en. In: (), p. 43.

Carl Bootland et al. "A framework for cryptographic problems from linear algebra". eng. In: *JOURNAL OF MATHEMATICAL CRYPTOLOGY* 14.1 (2020), pp. 202–217. ISSN: 1862-2976. URL: http://dx.doi.org/10.1515/jmc-2019-0032.

Zvika Brakerski et al. "Leveraging Linear Decryption: Rate-1 Fully-Homomorphic Encryption and Time-Lock

Puzzles". In: *TCC 2019: 17th Theory of Cryptography Conference, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. Lecture Notes in Computer Science. Nuremberg, Germany: Springer, Heidelberg, Germany, Dec. 2019, pp. 407–437. DOI: 10.1007/978-3-030-36033-7_16.

Chi-Ming Marvin Chung et al. *NTT Multiplication for NTT-unfriendly Rings*. Cryptology ePrint Archive, Report 2020/1397. https://eprint.iacr.org/2020/1397. 2020.

Jan-Pieter D'Anvers et al. "Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM". In: *AFRICACRYPT 18: 10th International Conference on Cryptology in Africa*. Ed. by Antoine Joux, Abderrahmane Nitaj, and

Tajjeeddine Rachidi. Vol. 10831. Lecture Notes in Computer Science. Marrakesh, Morocco: Springer, Heidelberg, Germany, May 2018, pp. 282–305. DOI: 10.1007/978-3-319-89339-6_16.

Jintai Ding. *New cryptographic constructions using generalized learning with errors problem*. Cryptology ePrint Archive, Report 2012/387. https://eprint.iacr.org/2012/387. 2012.

Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. "The Impact of Error Dependencies on Ring/Mod-LWE/LWR Based Schemes". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Chongqing, China:

Springer, Heidelberg, Germany, May 2019, pp. 103–115. DOI: 10.1007/978-3-030-25510-7_6.

Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology – CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34.

Siyao Guo et al. *Limits on the Efficiency of (Ring) LWE based Non-Interactive Key Exchange*. Cryptology ePrint Archive, Report 2020/1555. https://eprint.iacr.org/2020/1555. 2020.

Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A Modular Analysis of the Fujisaki-Okamoto Transformation". In: *TCC 2017: 15th Theory of Cryptography Conference, Part I*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10677. Lecture Notes in Computer Science. Baltimore, MD, USA: Springer, Heidelberg, Germany, Nov. 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12.

Zhengzhong Jin and Yunlei Zhao. *Optimal Key Consensus in Presence of Noise*. Cryptology ePrint Archive, Report 2017/1058. https://eprint.iacr.org/2017/1058. 2017.

Zhengzhong Jin and Yunlei Zhao. *AKCN-E8: Compact and Flexible KEM from Ideal Lattice*. Cryptology ePrint Archive, Report 2020/056. https://eprint.iacr.org/2020/056. 2020.

Sam Kim. "Key-Homomorphic Pseudorandom Functions from LWE with Small Modulus". In: *Advances in Cryptology – EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Heidelberg, Germany, May 2020, pp. 576–607. DOI: 10.1007/978-3-030-45724-2_20.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera:

Springer, Heidelberg, Germany, May 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1.

Chris Peikert. "Lattice Cryptography for the Internet". In: *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*. Ed. by Michele Mosca. Waterloo, Ontario, Canada: Springer, Heidelberg, Germany, Oct. 2014, pp. 197–219. DOI: 10.1007/978-3-319-11659-4_12.

Alex van Poppelen. *Cryptographic decoding of the Leech lattice*. Cryptology ePrint Archive, Report 2016/1050. https://eprint.iacr.org/2016/1050. 2016.

Charbel Saliba, Laura Luzzi, and Cong Ling. *Wyner-Ziv reconciliation for key exchange based on*

Ring-LWE. Cryptology ePrint Archive, Report 2020/076. https://eprint.iacr.org/2020/076. 2020.