HILA5: KEM and Public Key Encryption

From Ring-LWE and Error Correcting Codes

Markku-Juhani O. Saarinen
<mjos@mjos.fi>

P.O. Box 1339, CB1 0BZ, Cambridge, UK
Tel. US +1 (202) 559 0658

First NIST PQC Standardization Workshop

Fort Lauderdale, April 12, 2018

Key Encapsulation Mechanism (KEM) and Public Key Encryption

Following the NIST call [NI16] and Peikert [Pe14], our scheme is formalized as an IND-CPA Key Encapsulation Mechanism (KEM), consisting of three algorithms:

$(\mathsf{PK}, \mathsf{SK})$	$\leftarrow$	$\mathsf{KeyGen}()$.	Generate a public key PK and a secret key SK.
$(\mathsf{CT}, \mathsf{K})$	$\leftarrow$	$\mathsf{Encaps}(\mathsf{PK})$.	Encapsulate a (random) key K in ciphertext CT.
K	$\leftarrow$	$\mathsf{Decaps}(\mathsf{SK}, \mathsf{CT})$.	Decapsulate shared key K from CT with SK.

In this model, reconciliation data is a part of ciphertext produced by Encaps(). The three KEM algorithms constitute a naturalR single-roundtrip key exchange:

Alice		**Bob**
$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KeyGen}()$	$\xrightarrow{\mathsf{PK}}$	
	$\xleftarrow{\mathsf{CT}}$	$(\mathsf{CT}, \mathsf{K}) \leftarrow \mathsf{Encaps}(\mathsf{PK})$
$\mathsf{K} \leftarrow \mathsf{Decaps}(\mathsf{SK}, \mathsf{CT})$		

Thanks to its **low failure** rate ($< 2^{-128}$ due to **novel reconciliation methods** and **error correction**) HILA5 can also be used for **public key encryption** via (AEAD) Key Wrap.

Based on Ring-LWE (Learning with Errors in a Ring)

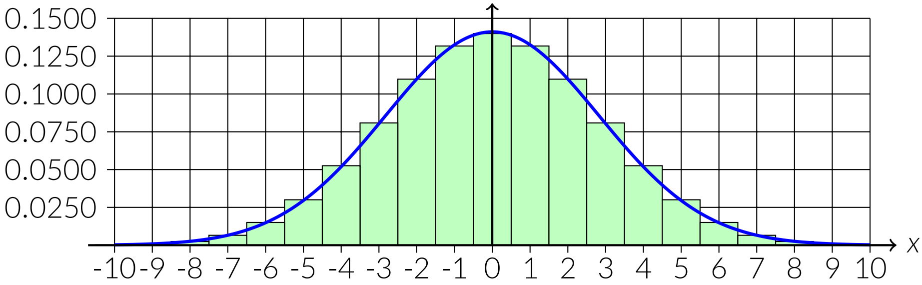
Let $\mathcal{R}$ be a ring with elements $\mathbf{v} \in \mathbb{Z}_q^n$. We use **fast NTT arithmetic** in $\mathbb{Z}_q[x]/(x^n + 1)$.

> **Definition (Informal)**
> With all distributions and computations in ring $\mathcal{R}$, let $\mathbf{s}, \mathbf{e}$ be elements randomly chosen from some non-uniform distribution χ, and $\mathbf{g}$ be a uniformly random public value. Determining $\mathbf{s}$ from $(\mathbf{g}, \mathbf{g} * \mathbf{s} + \mathbf{e})$ in ring $\mathcal{R}$ is the (Normal Form Search) Ring Learning With Errors ($\mathrm{RLWE}_{\mathcal{R},\chi}$) problem.

Typically χ is chosen so that each coefficient is a Discrete Gaussian or from some other "Bell-Shaped" distribution that is relatively tightly concentrated around zero.

The hardness of the problem is a function of n, q, and χ. **HILA5** uses **very fast** and **well-studied** *"New Hope"* parameters: $n = 1024$, $q = 3 * 2^{12} + 1 = 12289$, $\chi = \Psi_{16}$.

Discrete Gaussian $D_{\sqrt{8}}$ and Binomial "bitcount" Distribution Ψ_{16}



Green bars are the probability mass of binomial distribtion $P(X = x) = 2^{-32}\binom{32}{x+16}$.
Blue line is the discrete Gaussian distribution D_σ with deviation parameter $\sigma = \sqrt{8}$.

$\rho_\sigma(x) \propto \exp(-\frac{x^2}{2\sigma^2})$. Very good approximation: $\rho_\sigma(x) \approx \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$.

Noisy Diffie-Hellman in a Ring

Alice		Bob
$a \xleftarrow{\$} \chi$	*private keys*	$b \xleftarrow{\$} \chi$
$e \xleftarrow{\$} \chi$	*noise*	$e' \xleftarrow{\$} \chi$
$A = g * a + e$	*public keys*	$B = g * b + e'$
	$\xleftarrow{B} \quad \xrightarrow{A}$	
$x = B * a$	*shared secret*	$y = A * b$

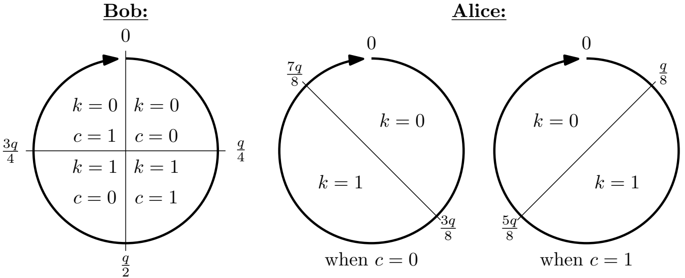
Here $\mathbf{g}$ is a uniform, public generator. By substituting variables in $\mathbf{A}$ and $\mathbf{B}$ we get

$$x = (g * b + e') * a = \underline{g * a * b} + e' * a$$
$$y = (g * a + e) * b = \underline{g * a * b} + e * b.$$

Because error terms are much smaller than the common term $\underline{g * a * b}$ we have $x \approx y$.

Reconciliation: Traditionally Needs Random Numbers

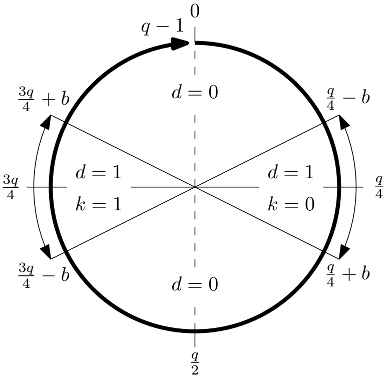
In **reconciliation**, we wish the holders of **x** and **y** (Alice and Bob, respectively) to arrive at exactly the same shared secret **k** with minimal communication **c**.

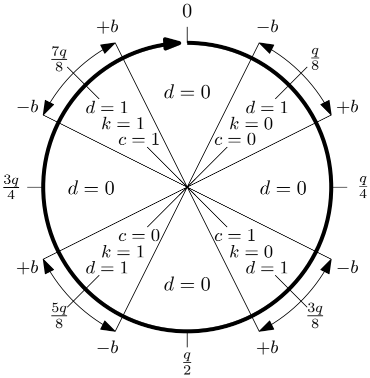


In Peikert's reconciliation [Pe14] Bob sends 1 "phase bit" c for each vector element.

Since q is odd and cannot be evenly divided in half, a **fresh random bit** is needed to "smoothen" the divide. **New Hope's** reconciliation of also needs random numbers.

HILA5's Novel "Safe Bits" Reconciliation and Error Correction

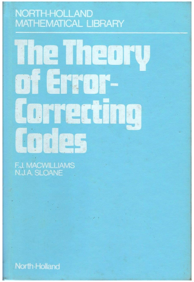




As we don't need $n = 1024$ bits, we can select "**Safe Bits**" away from the decision boundary in order to get unbiased secrets **without using additional randomness**.

We designed **error correction codes** to push the failure probability well under 2^{-128}.

Error Correction Code XE5



$\leftarrow$ Hey students! Pay attention in the coding theory classes!

I designed a linear block code, XE5, specifically for HILA5.

Security Requirement: Fast, constant-time implementatable.

After various considerations (**SafeBits**), ended up with a block size of 496 bits (256 message bits + 240 redundancy bits.)

Always corrects 5 random bit flips, more with high probability.

I first described similar constant-time error correction techniques (for TRUNC8) in:

M.-J. O. Saarinen. **"Ring-LWE ciphertext compression and error correction: Tools for lightweight post-quantum cryptography"**. Proc. 3rd ACM International Workshop on IoT Privacy, Trust, and Security, IoTPTS '17, pp. 15-22. ACM, April 2017.

`https://eprint.iacr.org/2016/1058` (Original uploaded November 15, 2016)

Pindakaas: HILA5 is IND-CPA, not IND-CCA

[BBLP17] D. J. Bernstein, L. G. Bruinderink, T. Lange, and L. Panny: **"HILA5 pindakaas: On the CCA security of lattice-based encryption with error correction."** IACR ePrint 2017/1214. `https://eprint.iacr.org/2017/1214`.

There is a single point on p. 17 of the HILA5 specification which erroneously claims IND-CCA security. With (too) much speculation this was shown not to be correct in [BBLP17]. The original SAC 2017 academic paper never even mentions IND-CCA.

Furthermore even [BBLP17] itself clearly states that:

> *"We emphasize that our attack does not break the IND-CPA security of HILA5. If HILA5 were clearly labeled as aiming merely for IND-CPA security then our attack would merely be a cautionary notee, showing the importance of not reusing keys."*

Creating an IND-CCA variant via **Fujisaki–Okamoto transform** is straightforward.
I will propose such variant, probably not very dissimilar to "HILA5FO" from [BBLP17].

What Distinguishes HILA5 from the Rest ?

+ **It's Very Fast and can do KEM <u>and</u> Public Key Encryption.** Only about 5% slower than fastest New Hope (CPA) implementation (Matching Ring-LWE parameters.) I'll have to get better NTT code for the new version, **my current NTT code sucks!**
+ **Less randomness required.** Reconciliation method produces unbiased secrets without randomized smoothing; much less randomness is therefore required.
+ **HILA5 decryption doesn't fail.** HILA5 has a failure rate well under 2^{-128}. Non-negligible decryption failure rate is needed in **public key encryption**.
+ **Non-malleable.** Computation of the final shared secret in HILA5 KEM uses the full public key and ciphertext messages, thereby reinforcing non-malleability and making a class of adaptive attacks infeasible.
+ **Shorter messages.** Ciphertext messages are slightly smaller than New Hope's.
+ **Patent free.** As the sender can "choose the message" (as in NEWHOPE-SIMPLE), Ding's Ring-LWE key exchange patents less likely to be applicable.

HILA5 Spec Sheet: Questions ?

Algorithm Purpose:	Key Encapsulation and Public Key Encryption.
Underlying problem:	Ring-LWE (New Hope: $n = 1024$, $q = 12289$, Ψ_{16})
Public key size:	1824 Bytes (+32 Byte private key hash.)
Private key size:	1792 Bytes (640 Bytes compressed.)
Ciphertext size:	2012 Byte expansion (KEM) + payload + MAC.
Failure rate:	$< 2^{-128}$, consistent with security level.
Classical security:	2^{256} (Category 5 – Equivalent to AES-256).
Quantum security:	2^{128} (Category 5 – Equivalent to AES-256).

Paper: M.-J. O. Saarinen: "HILA5: On Reliability, Reconciliation, and Error Correction for Ring-LWE Encryption." Selected Areas in Cryptography – SAC 2017, LNCS 10719, Springer, pp. 192-212, 2018. `https://eprint.iacr.org/2017/424`

Always get the latest code and specs at: `https://github.com/mjosaarinen/hila5`