Lattice-Based Cryptography

A Tale for the Modern Age: concerning Hobbits and Lightsabers

Daniel Apon
National Institute of Standards and Technology
December 7, 2021

dapon.crypto@gmail.com

If you re-read these slides offline later, here is the proper musical accompaniment:

Star Wars: The Force Theme – by John Williams

https://www.youtube.com/watch?v=W1937VEYguI

Frodo from Mars, and Kyber from Tatooine

What is Frodo?

A Key Exchange Mechanism (KEM) based on (plain) Learning with Errors (LWE).

The Frodo team:

Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, Douglas Stebila

Why is Frodo interesting?

It's the most conservative / security-conscious lattice-based KEM.

What is Kyber?

A KEM based on Module (i.e. algebraically-structured) Learning with Errors (MLWE).

The Kyber team:

Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé

Why is Kyber interesting?

It's one of the fastest/smallest lattice-based KEMs believed to be secure in the real world.

Learning with Errors [Regev04] (and earlier…)

error distribution, e.g., Gaussian

What is LWE? (all arithmetic mod q)

Random secret $\boldsymbol{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$.

- hard (search): find $\boldsymbol{s}$ from samples $(\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + e)$ [where $\boldsymbol{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $e \leftarrow \chi(\mathbb{Z}_q)$];
- hard (decision): distinguish above (i.e., LWE) samples from random (i.e., $\mathcal{U}(\mathbb{Z}_q^{n+1})$) samples.

Some notes:

- without errors, easy with Gaussian elim.; with errors, seems hard;
- $\boldsymbol{s}$ can also be short, i.e., $\boldsymbol{s} \leftarrow \chi(\mathbb{Z}_q^n)$

Asymptotic reductions:

- random self-reducible

  (succeed on non-negl fraction of instances $\Rightarrow$ succeed on all instances);
- can reduce LWE search to LWE decision;
- can reduce worst-case lattice problems to LWE search;

(But not all of this applies for practically relevant parameters.)

Besides all this awesomeness, you can also build pretty much any cryptographic thing you want from LWE!

Learning with Errors [Regev05]

error distribution, e.g., short Gaussian

What is LWE? (all arithmetic mod q)

Random secret $\boldsymbol{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$.

- hard (search): find $\boldsymbol{s}$ from samples $(\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + e)$ [where $\boldsymbol{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $e \leftarrow \chi(\mathbb{Z}_q)$];
- hard (decision): distinguish above (i.e., LWE) samples from random (i.e., $\mathcal{U}(\mathbb{Z}_q^{n+1})$) samples.

Immediately yields Public Key Encryption (PKE):

Sample $sk = \boldsymbol{s}$; $pk = $ a bunch of samples $\left(\boldsymbol{a}_j, \boldsymbol{a}_j \cdot \boldsymbol{s} + e_j\right)$

- to encrypt 0, sum up a random subset of the samples;

  (random-ish $\approx$ *true* equation)
- to encrypt 1, sum up a random subset of the samples and add $q/2$ to the last entry

  (random-ish $\approx$ *false* equation)

Easy to tell which is which (and thus decrypt) with sk.

Learning with Errors [Regev05, LP11]

unstructured LWE (compare to, e.g., Ring/Module-LWE)

Rewrite in matrix form.

Collect up the samples into rows:

$$(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \qquad \text{[where } \boldsymbol{s} \leftarrow \mathcal{U}(\mathbb{Z}_q^n) \text{ and } \boldsymbol{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{\ell \times n}) \text{ and } \boldsymbol{e} \leftarrow \chi(\mathbb{Z}_q^{\ell})\text{]}$$

The same PKE from last slide is now:

$$\mathrm{Enc}_{(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})}(\mu; r) = \left(r^{\mathrm{T}}\boldsymbol{A}, r^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) + \mu \cdot q/2\right) \qquad \text{where } r \leftarrow \{0,1\}^{\ell} \subset \mathbb{Z}_q^{\ell} \text{ (as col. vector)}$$

$+e'$ ← could add more error at encryption stage

Actual change: replace vectors $(\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{r})$ with matrices [LP11].

$$(\boldsymbol{A}, \boldsymbol{B} = \boldsymbol{A}\boldsymbol{S} + \boldsymbol{E}) \qquad \text{[where } \boldsymbol{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n}) \text{ and } \boldsymbol{S}, \boldsymbol{E} \leftarrow \chi(\mathbb{Z}_q^{n \times \bar{n}})\text{]}$$

Make a PKE like this now:

some encoding of ptxt m into a matrix

$$\mathrm{Enc}_{(\boldsymbol{A}, \boldsymbol{B})}(\mu; \boldsymbol{S}', \boldsymbol{E}', \boldsymbol{E}'') = (\boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}', \boldsymbol{S}'\boldsymbol{B} + \boldsymbol{E}'' + \boldsymbol{M}(\mu) \cdot q/2)$$

To decrypt $(\boldsymbol{C}_1, \boldsymbol{C}_2)$, compute $\boldsymbol{M} = \boldsymbol{C}_2 - \boldsymbol{C}_1\boldsymbol{S}$ and decode (e.g., output first t bits of each entry.)

Advantage: better parameters than original PKE.

From LP11 to FrodoPKE

- Efficient LWE PKE [LP11]. $(\boldsymbol{A}, \boldsymbol{B} = \boldsymbol{AS} + \boldsymbol{E})$ [where $\boldsymbol{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times n})$ and $\boldsymbol{S}, \boldsymbol{E} \leftarrow \chi(\mathbb{Z}_q^{n \times \bar{n}})$]

To encrypt μ : $\mathrm{Enc}_{(\boldsymbol{A},\boldsymbol{B})}(\mu; \boldsymbol{S}', \boldsymbol{E}', \boldsymbol{E}'') = (\boldsymbol{S}'\boldsymbol{A} + \boldsymbol{E}', \boldsymbol{S}'\boldsymbol{B} + \boldsymbol{E}'' + \boldsymbol{M}(\mu) \cdot q/2^B)$

where $\boldsymbol{S}', \boldsymbol{E}' \leftarrow \chi(\mathbb{Z}_q^{\bar{m} \times n})$ and $\boldsymbol{E}'' \leftarrow \chi(\mathbb{Z}_q^{\bar{m} \times \bar{n}})$

To decrypt $(\boldsymbol{C}_1, \boldsymbol{C}_2)$: compute $\boldsymbol{M} = \boldsymbol{C}_2 - \boldsymbol{C}_1\boldsymbol{S}$ and decode (e.g., output first t bits of each B-chunk of each entry.)

Algorithm 10 FrodoPKE.Enc.

Input: Message $\mu \in \mathcal{M}$ and public key $pk = (\mathsf{seed}_{\mathbf{A}}, \mathbf{B}) \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_\mathbf{A}}} \times \mathbb{Z}_q^{n \times \bar{n}}$.
Output: Ciphertext $c = (\mathbf{C}_1, \mathbf{C}_2) \in \mathbb{Z}_q^{\bar{m} \times n} \times \mathbb{Z}_q^{\bar{m} \times \bar{n}}$.

Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_\mathbf{A})$
Choose a uniformly random seed $\mathsf{seed}_{\mathbf{SE}} \leftarrow_\$ U(\{0,1\}^{\mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}}})$
Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\bar{m}n+\bar{m}\bar{n}-1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x96}\|\mathsf{seed}_{\mathbf{SE}}, (2\bar{m}n + \bar{m}\bar{n}) \cdot \mathsf{len}_\chi)$
Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\bar{m}n-1)}), \bar{m}, n, T_\chi)$
Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(\bar{m}n)}, \mathbf{r}^{(\bar{m}n+1)}, \ldots, \mathbf{r}^{(2\bar{m}n-1)}), \bar{m}, n, T_\chi)$
Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(2\bar{m}n)}, \mathbf{r}^{(2\bar{m}n+1)}, \ldots, \mathbf{r}^{(2\bar{m}n+\bar{m}\bar{n}-1)}), \bar{m}, \bar{n}, T_\chi)$
Compute $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$
return ciphertext $c \leftarrow (\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + \mathsf{Frodo.Encode}(\mu))$

LP11

	n	q	$\bar{m} \times \bar{n}$
Frodo-640	640	2^{15}	8×8
Frodo-976	976	2^{16}	8×8
Frodo-1344	1344	2^{16}	8×8

From FrodoPKE to FrodoKEM

FrodoPKE only satisfies IND-CPA security.

Totally broken even in CCA1: submit $(\mathbf{0}, -\boldsymbol{I})$ to Dec oracle, get bits of $\boldsymbol{S}$ as response.

Fujisaki-Okamoto transform [FO11].

OW-CPA PKE + one-time SKE + hashes $G, H \implies$ IND-CCA2 PKE (in the ROM):

Key generation: just use $\mathrm{PKE.KeyGen}$.

To encrypt m:

- $r \leftarrow \{0,1\}^*$;
- $c = \mathrm{Enc}^{\mathrm{SKE}}_{G(r)}(m)$
- Output $\left(c, Enc^{PKE}_{pk}(r; H(r,c))\right)$

$\underline{\mathsf{Gen}^{\not\perp}}$

$(pk', sk') \leftarrow \mathsf{Gen}_1$

$s \xleftarrow{\$} \mathcal{M}$

$sk := (sk', s)$

$\mathbf{return}\ (pk', sk)$

$\underline{\mathsf{Encaps}(pk)}$

$m \xleftarrow{\$} \mathcal{M}$

$c \leftarrow \mathsf{Enc}_1(pk, m)$

$K := \mathsf{H}(m, c)$

$\mathbf{return}\ (K, c)$

$\underline{\mathsf{Decaps}^{\not\perp}(sk, c)}$

$\text{Parse } sk = (sk', s)$

$m' := \mathsf{Dec}_1(sk', c)$

$\mathbf{if}\ m' \neq \perp$

$\quad \mathbf{return}\ K := \mathsf{H}(m', c)$

$\mathbf{else\ return}\ K := \mathsf{H}(s, c)$

Lots of work on variants of FO since then; most relevant are maybe [TU16, HHK17].

Frodo will use a variant of a transform from [HHK17] (the same one as Kyber [BDK+17].)

From FrodoPKE to FrodoKEM

- FrodoPKE only satisfies IND-CPA security.

Totally broken even in CCA1: submit $(\mathbf{0}, -\boldsymbol{I})$ to Dec oracle, get bits of $\boldsymbol{S}$ as response.

HHK transform [HHK17]: "FO with implicit rejection."

OW-CPA PKE + hash $H \implies$ IND-CCA2 **KEM** (in the ROM):

$\underline{\mathsf{Gen}^{\not\perp}}$

$(pk, sk) \leftarrow \mathsf{Gen}$

$s \xleftarrow{\$} \mathcal{M}$

$sk' := (sk, s)$

$\mathbf{return}\ (pk, sk')$

for implicit rejection

$\underline{\mathsf{Encaps}(pk)}$

$m \xleftarrow{\$} \mathcal{M}$

$c := \mathsf{Enc}(pk, m; G(m))$

$K := \mathsf{H}(m, c)$

$\mathbf{return}\ (K, c)$

$\underline{\mathsf{Decaps}^{\not\perp}(sk' = (sk, s), c)}$

$m' := \mathsf{Dec}(sk, c)$

$\mathbf{if}\ c \neq \mathsf{Enc}(pk, m'; G(m'))\ \mathbf{or}\ m' = \perp$

$\quad \mathbf{return}\ K := \mathsf{H}(s, c)$

$\mathbf{else\ return}\ K := \mathsf{H}(m', c)$

implicit rejection

shared secret

From FrodoPKE to FrodoKEM

The FrodoKEM transform (Variant of [HHK17])

$\mathsf{KEM}^{\not\perp'}.\mathrm{KeyGen}()$:

$(pk, sk) \leftarrow_{\$} \mathsf{PKE}.\mathrm{KeyGen}()$
$\mathbf{s} \leftarrow_{\$} \{0,1\}^{\mathsf{len_s}}$
$\mathbf{pkh} \leftarrow G_1(pk)$
$sk' \leftarrow (sk, \mathbf{s}, pk, \mathbf{pkh})$
$\mathbf{return}\ \ (pk, sk')$

$\mathsf{KEM}^{\not\perp'}.\mathrm{Encaps}(pk)$:

$\mu \leftarrow_{\$} \mathcal{M}$
$(\mathbf{r}, \mathbf{k}) \leftarrow G_2(G_1(pk) \| \mu)$
$c \leftarrow \mathsf{PKE}.\mathrm{Enc}(\mu, pk; \mathbf{r})$
$\mathbf{ss} \leftarrow F(c \| \mathbf{k})$
$\mathbf{return}\ \ (c, \mathbf{ss})$

$\mathsf{KEM}^{\not\perp'}.\mathrm{Decaps}(c, (sk, \mathbf{s}, pk, \mathbf{pkh}))$:

$\mu' \leftarrow \mathsf{PKE}.\mathrm{Dec}(c, sk)$
$(\mathbf{r}', \mathbf{k}') \leftarrow G_2(\mathbf{pkh} \| \mu')$
$\mathbf{ss}_0' \leftarrow F(c \| \mathbf{k}')$
$\mathbf{ss}_1' \leftarrow F(c \| \mathbf{s})$
(in constant time) $\mathbf{ss}' \leftarrow \mathbf{ss}_0'$ if $c = \mathsf{PKE}.\mathrm{Enc}(\mu', pk; \mathbf{r}')$ else $\mathbf{ss}' \leftarrow \mathbf{ss}_1'$
$\mathbf{return}\ \ \mathbf{ss}'$

HHK17

$\mathsf{Gen}^{\not\perp}$

$(pk, sk) \leftarrow \mathsf{Gen}$
$s \xleftarrow{\$} \mathcal{M}$
$sk' := (sk, s)$
$\mathbf{return}\ (pk, sk')$

$\mathsf{Encaps}(pk)$

$m \xleftarrow{\$} \mathcal{M}$
$c := \mathsf{Enc}(pk, m; G(m))$
$K := \mathsf{H}(m, c)$
$\mathbf{return}\ (K, c)$

$\mathsf{Decaps}^{\not\perp}(sk' = (sk, s), c)$

$m' := \mathsf{Dec}(sk, c)$
$\mathbf{if}\ c \neq \mathsf{Enc}(pk, m'; G(m'))\ \mathbf{or}\ m' = \perp$
$\quad \mathbf{return}\ K := \mathsf{H}(s, c)$
$\mathbf{else\ return}\ K := \mathsf{H}(m', c)$

The actual FrodoKEM (Encaps)

Algorithm 13 FrodoKEM.Encaps.

Input: Public key $pk = \mathsf{seed}_{\mathbf{A}} \| \mathbf{b} \in \{0,1\}^{\mathsf{len}_{\mathsf{seed}_{\mathbf{A}}} + D \cdot n \cdot \overline{n}}$.

Output: Ciphertext $\mathbf{c}_1 \| \mathbf{c}_2 \in \{0,1\}^{(\overline{m} \cdot n + \overline{m} \cdot \overline{n})D}$ and shared secret $\mathbf{ss} \in \{0,1\}^{\mathsf{len}_{\mathbf{ss}}}$.

FrodoPKE

FO xform variant

1: Choose a uniformly random key $\mu \leftarrow_{\$} U(\{0,1\}^{\mathsf{len}_\mu})$

2: Compute $\mathbf{pkh} \leftarrow \mathrm{SHAKE}(pk, \mathsf{len}_{\mathbf{pkh}})$

3: Generate pseudorandom values $\mathsf{seed}_{\mathbf{SE}} \| \mathbf{k} \leftarrow \mathrm{SHAKE}(\mathbf{pkh} \| \mu, \mathsf{len}_{\mathsf{seed}_{\mathbf{SE}}} + \mathsf{len}_{\mathbf{k}})$

4: Generate pseudorandom bit string $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(2\overline{m}n + \overline{m}\overline{n} - 1)}) \leftarrow \mathrm{SHAKE}(\mathtt{0x96} \| \mathsf{seed}_{\mathbf{SE}}, (2\overline{m}n + \overline{m}\overline{n}) \cdot \mathsf{len}_\chi)$

5: Sample error matrix $\mathbf{S}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \ldots, \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_\chi)$

6: Sample error matrix $\mathbf{E}' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_\chi)$

7: Generate $\mathbf{A} \leftarrow \mathsf{Frodo.Gen}(\mathsf{seed}_{\mathbf{A}})$

8: Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$

9: Compute $\mathbf{c}_1 \leftarrow \mathsf{Frodo.Pack}(\mathbf{B}')$

10: Sample error matrix $\mathbf{E}'' \leftarrow \mathsf{Frodo.SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, \ldots, \mathbf{r}^{(2\overline{m}n+\overline{m}\overline{n}-1)}), \overline{m}, \overline{n}, T_\chi)$

11: Compute $\mathbf{B} \leftarrow \mathsf{Frodo.Unpack}(\mathbf{b}, n, \overline{n})$

12: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$

13: Compute $\mathbf{C} \leftarrow \mathbf{V} + \mathsf{Frodo.Encode}(\mu)$

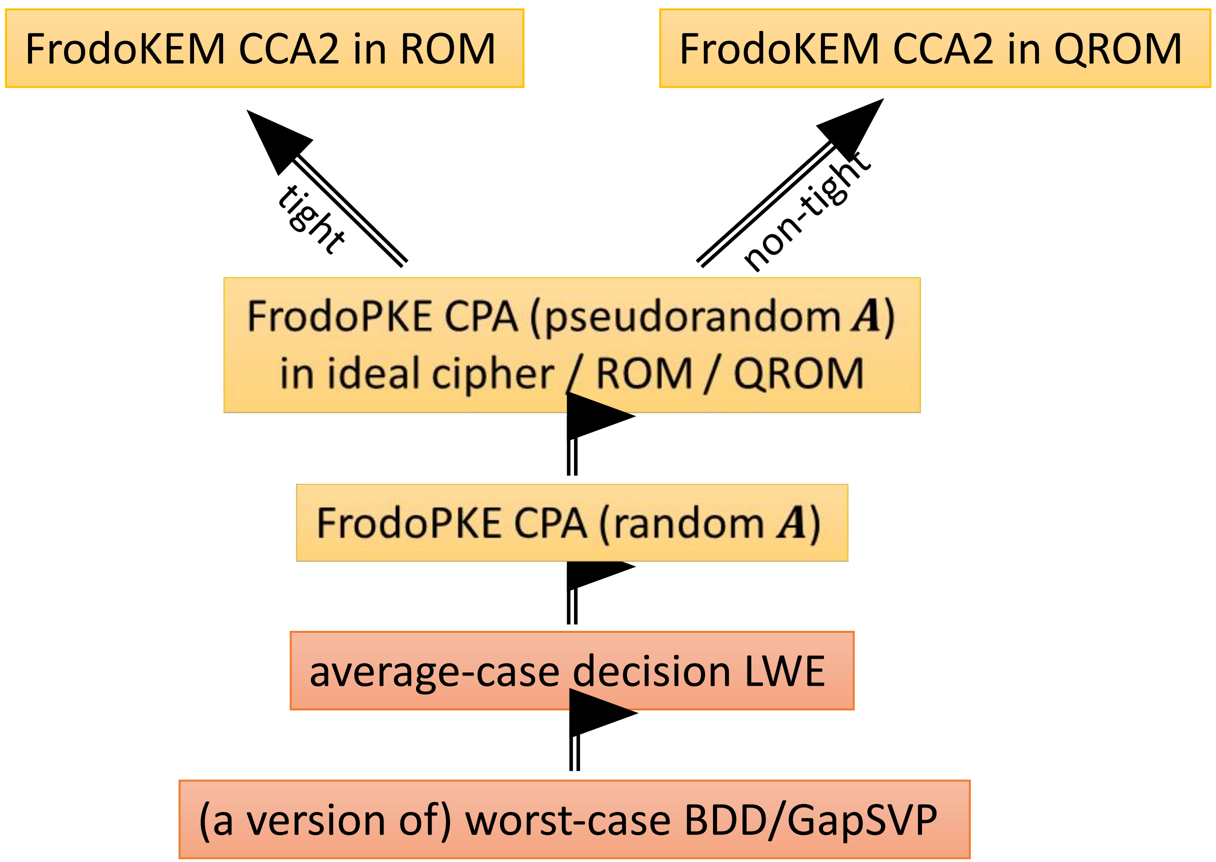
14: Compute $\mathbf{c}_2 \leftarrow \mathsf{Frodo.Pack}(\mathbf{C})$

15: Compute $\mathbf{ss} \leftarrow \mathrm{SHAKE}(\mathbf{c}_1 \| \mathbf{c}_2 \| \mathbf{k}, \mathsf{len}_{\mathbf{ss}})$

16: **return** ciphertext $\mathbf{c}_1 \| \mathbf{c}_2$ and shared secret $\mathbf{ss}$

Frodo’s Security (asymptotic)

•



($A \Rightarrow B$ means if A is hard/secure, then so is B.)

Frodo's Security (concrete)

Parameters based on actual cryptanalytic attacks.

- the best attacks are primal and dual BKZ [CN11];
- roughly, these are two different ways of converting an LWE instance into an SVP instance
- In either case, SVP is then solved with BKZ with block size b
- This involves poly-many calls to SVP in dimension b;
- Core-SVP hardness : set params under assumption only one SVP call enough to break;

Table 10: **Primal and dual attacks on a single instance of an LWE problem.** Attack costs are given as the base-2 logarithm.

based on "known unknowns" discussion in Kyber round 3 spec

Scheme	Attack Mode	Classical	Quantum	Plausible
Frodo-640	Primal	150.8	137.6	109.6
	Dual	149.6	136.5	108.7
Frodo-976	Primal	216.0	196.7	156.0
	Dual	214.5	195.4	154.9
Frodo-1344	Primal	281.6	256.3	202.6
	Dual	279.8	254.7	201.4

Parameters, details

- L1, L3, L5 Parameter sets (i.e. 128-bit-, 192-bit-, 256-bit-secure cryptosystems):

Table 1: **Parameters at a glance**

	n	q	σ	**support of** χ	B	$\bar{m} \times \bar{n}$	c **size (bytes)**
Frodo-640	640	2^{15}	2.8	$[-12 \dots 12]$	2	8×8	9,720
Frodo-976	976	2^{16}	2.3	$[-10 \dots 10]$	3	8×8	15,744
Frodo-1344	1344	2^{16}	1.4	$[-6 \dots 6]$	4	8×8	21,632

error distribution and its std dev

- $q = 2^D$, a power-of-two integer modulus with exponent $D \leq 16$;
- $n, \overline{m}, \overline{n}$, integer matrix dimensions with $n \equiv 0 \pmod{16}$;
- $B \leq D$, the number of bits encoded in each matrix entry;

of the matrix $M(\mu)$

Recall:

- in key generation: $\boldsymbol{A}$ is $n \times n$ and $\boldsymbol{S}, \boldsymbol{E}$ are $n \times \bar{n}$
- in encryption: $\boldsymbol{S}', \boldsymbol{E}'$ are $\bar{m} \times n$ and $\boldsymbol{E}''$ is $\bar{m} \times \bar{n}$
- entries are all in $\mathbb{Z}_q$.

- error distribution is not quite Gaussian;
- it's simpler and easier to sample from…
- … and a Rényi divergence argument shows it's (basically) just as good.

Two variants, depends on RO instantiation: SHAKE and AES (latter for platforms with AES hardware acceleration)

Module Learning with Errors [Brakerski-Gentry-Vaikuntanathan2011]

What is MLWE? (all arithmetic mod q)

Pick a polynomial ring R_q.

A typical choice is $R_q = \mathbb{Z}_q[x]/(x^d + 1)$, for "ring dimension" d that's some power of 2 (e.g., 1024, 2048).

Elements of R_q are polynomials $\mathbf{a}$ that look like $\mathbf{a} = a_0 + a_1 x + \cdots + a_{d-1}x^{d-1}$.

The arithmetic will be matrix-wise, with entries that are polynomials in the ring.

error distribution, e.g., short Gaussian for every coefficient of the polynomials

Random secret $\boldsymbol{s} \leftarrow \mathcal{U}(R_q^k)$ for a small "module rank" k (e.g., 2, 3, 4).

- hard (search): find $\boldsymbol{s}$ from samples $(\boldsymbol{A}, \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e})$ [where $\boldsymbol{A} \leftarrow \mathcal{U}(R_q^{m \times k})$ and $\boldsymbol{e} \leftarrow \chi(R_q^m)$];
- hard (decision): distinguish above (i.e., MLWE) samples from random (i.e., $\mathcal{U}(R_q^m)$) samples.

Kyber in a nutshell – Do Frodo, but with MWLE instead of LWE!

What is Kyber?

It's constructed just like FrodoKEM (modulo some details and some implementation tricks)

Why the extra complication of algebraic structure?

The public keys drop in size by around a factor 10x to 100x, but we don't think any security is lost.

Some parameters can be chosen smaller in practice (so ciphertexts get smaller too).

Also, Kyber runs about 10x to 100x faster than Frodo (cycle counts of KeyGen, Encaps, Decaps).

For example, degree-d polynomial multiplication, using the Number Theoretic Transform (NTT), is $\approx \mathrm{d}\log(\mathrm{d})$ time.

Conclusion

Thanks for your attention!

NIST will be selecting the first post-quantum standards for KEMs (and digital signatures) around the end of December or sometime in early January.

What an exciting time!

Questions?

References

[Regev04] : https://dl.acm.org/doi/pdf/10.1145/1568318.1568324

[LP11] : https://eprint.iacr.org/2010/613

[FO11] : https://link.springer.com/content/pdf/10.1007/s00145-011-9114-1.pdf

[BGV11] : https://eprint.iacr.org/2011/277/

[TU16] : https://eprint.iacr.org/2015/1210

[HHK17] : https://eprint.iacr.org/2017/604

[BDK+17] : https://eprint.iacr.org/2017/634

[CN11] : https://www.iacr.org/archive/asiacrypt2011/70730001/70730001.pdf

[RRB+18] : https://eprint.iacr.org/2018/211

[JZC+18] : https://eprint.iacr.org/2017/1096

[GJN20] : https://eprint.iacr.org/2020/743

[HMOR21] : https://eprint.iacr.org/2021/155

[BOR+21] : https://eprint.iacr.org/2021/711

References

[KJK+21] : https://link.springer.com/chapter/10.1007%2F978-3-030-89432-0_17

[GMP21] : https://eprint.iacr.org/2021/708

[Xagawa21a] : https://eprint.iacr.org/2021/1323

[Xagawa21b] : https://eprint.iacr.org/2021/741

[XIU+21] : https://eprint.iacr.org/2021/840

[UXT+21] : https://eprint.iacr.org/2021/849

Frodo specification : https://frodokem.org/files/FrodoKEM-specification-20210604.pdf

Kyber specification : https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf