SMAUG, the Module lattice based key exchange algorithm

Jung Hee Cheon[1, 2], Hyeongmin Choe[1], **Dongyeon Hong**[2], Jeongdae Hong[3], Hyoeun Seong[2], Junbum Shin[2], MinJune Yi[1, 2]

[1]Seoul National University, [2]CryptoLab Inc., [3]Ministry of National Defense

February 22, 2023

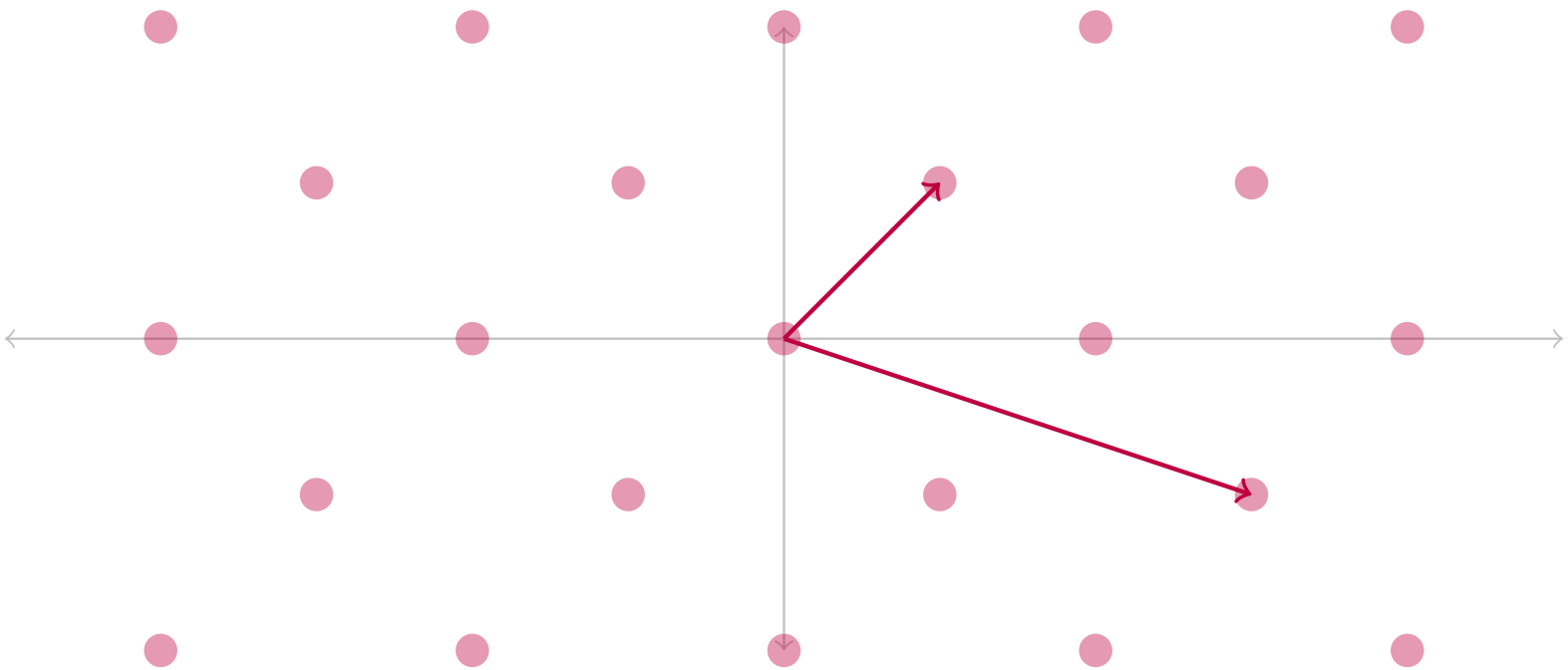
SMAUG
HEAAN
CRYPTO LAB

Preliminaries

Definition

A lattice $\mathcal{L}$ is a set of integer linear combinations of independent vectors $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$ of $\mathbb{R}^n$,

$$\mathcal{L} = \left\{ \sum_i a_i \cdot \boldsymbol{v}_i | a_i \in \mathbb{Z} \right\}$$



Preliminaries

Definition

LWE (Learning with Errors): given a pair $(\boldsymbol{A}, \boldsymbol{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ where $\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, $\boldsymbol{s} \leftarrow \chi$, and $\boldsymbol{e} \leftarrow \mathsf{DG}_\sigma$, $(\boldsymbol{A}, \boldsymbol{b})$ is indistinguishable with a uniformly random sample from $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

Definition

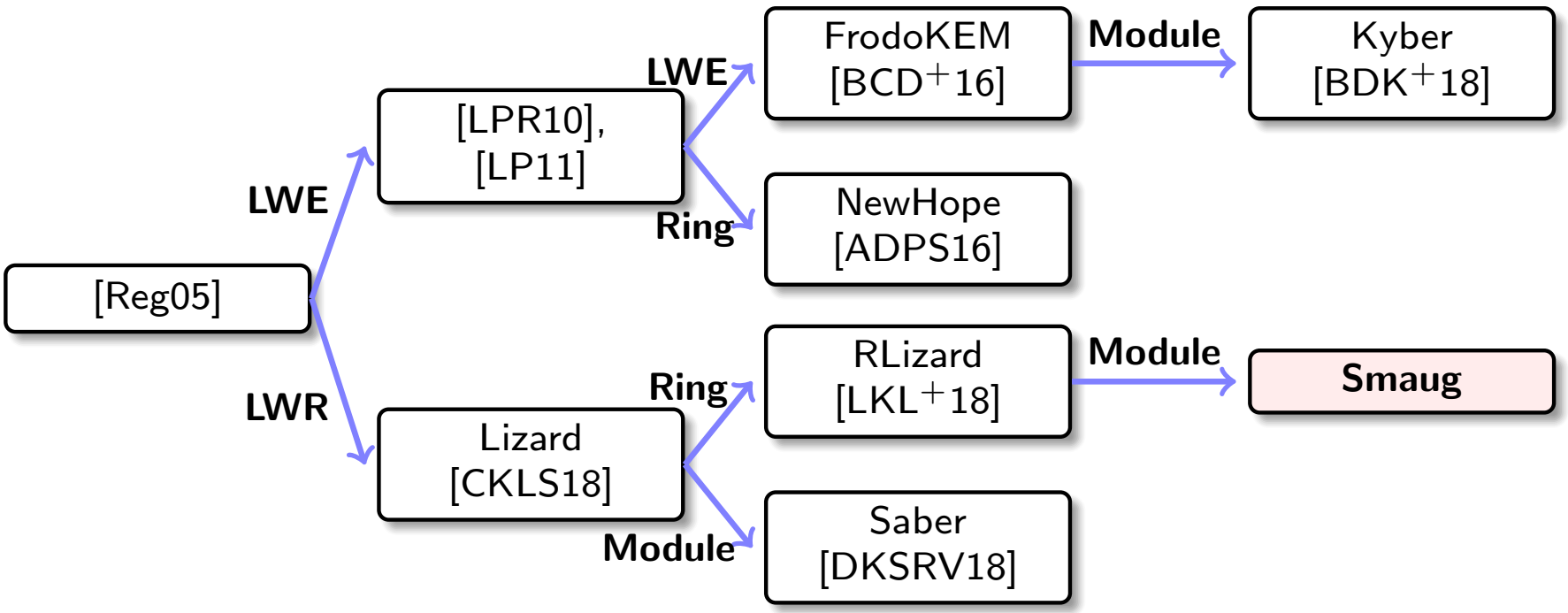
Module variant of LWE: given a pair $(\boldsymbol{A}, \boldsymbol{b}) \in \mathcal{R}_q^{m \times n} \times \mathcal{R}_q^m$ where $\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, $\boldsymbol{s} \leftarrow \chi''$ and $\boldsymbol{e} \leftarrow \mathsf{DG}_{\sigma,\mathcal{R}}^m$, $(\boldsymbol{A}, \boldsymbol{b})$ is indistinguishable with a uniformly random sample from $\mathcal{R}_q^{m \times n} \times \mathcal{R}_q^m$.

Definition

LWR (Learning with Rounding): given a pair $(\boldsymbol{A}, \boldsymbol{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ where $\boldsymbol{b} = \lfloor p/q \, (\boldsymbol{A} \cdot \boldsymbol{s}) \rceil$ and $\boldsymbol{s} \leftarrow \chi$, $(\boldsymbol{A}, \boldsymbol{b})$ is indistinguishable with a uniformly random sample from $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

History of Lattice PKE/KEM

History of PKE and KEM based on LWE and LWR



Design goal

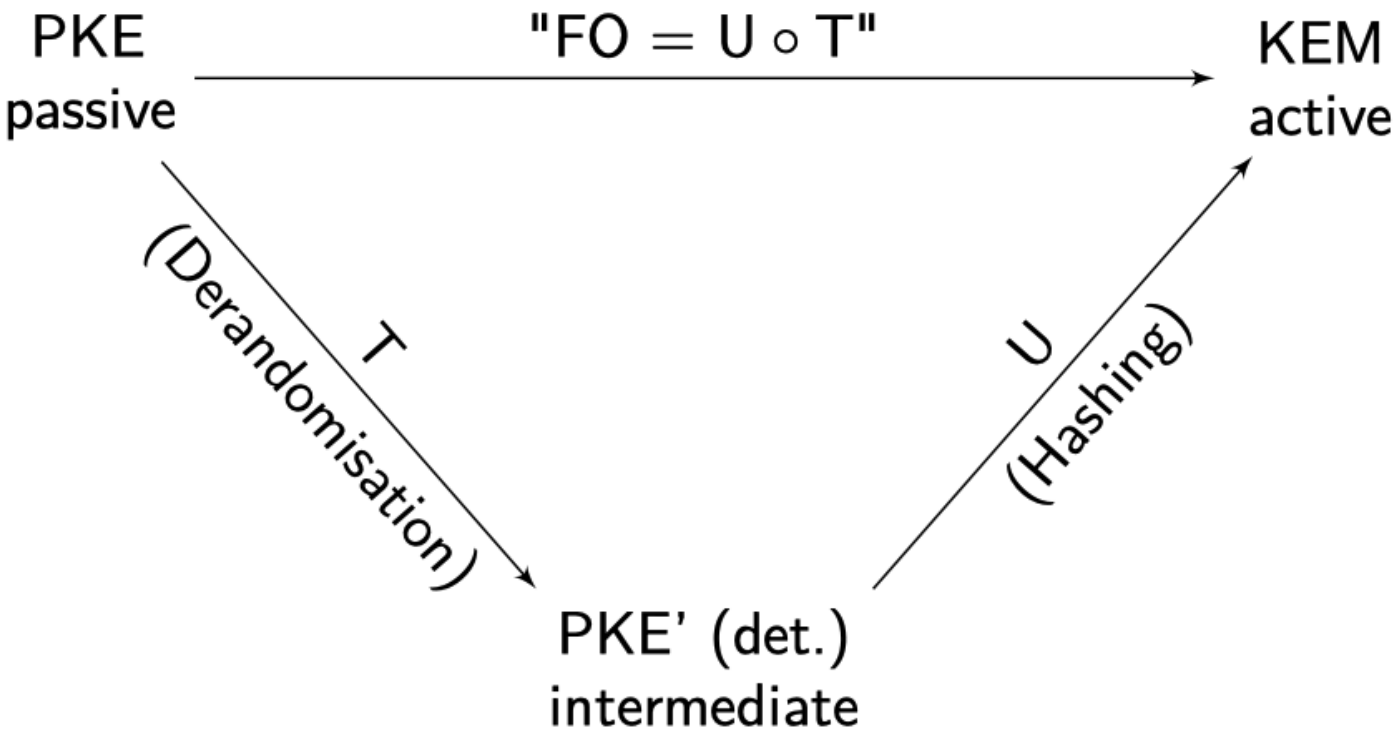
Goal: Suitable to lightweight devices

- Better performance: faster than previous works
- Small memory: smaller size in software
- Low communication cost: smaller key and ciphertext size
- Resistant to side-channel attack: constant-time implementation

Scheme - IND-CPA PKE to IND-CCA KEM

Basic idea: FO (Fujisaki-Okamoto) transformation [FO99]

- from IND-CPA PKE to IND-CCA KEM scheme



Scheme - FO transformation

Two steps of FO transformation

- Derandomization T
  - Make an IND-CPA PKE scheme deterministic

$$\mathsf{Encrypt}_1(pk, \mu) = \mathsf{Encrypt}(pk, \mu; H(\mu))$$

where $H(\mu)$ is as the random coin [HHK17].

- Hashing U
  - $\mathsf{U}^{\perp}$: turn an IND-CPA PKE into an IND-CCA KEM

Encapsulation

1. Choose a uniformly random message μ
2. $c = \mathsf{Encrypt}_1(pk, \mu)$
3. $K = \mathsf{KDF}(\mu, c)$

(a)

Decapsulation

1. $\mu' = \mathsf{Decrypt}_1(sk, c)$
2. if $\mu' = \perp$ return $\perp$
3. else return $K = \mathsf{KDF}(\mu', c)$

(b)

Scheme - FO transformation

Two steps of FO transformation

- Hashing U
  - Many variants of $U^{\perp}$
    - Implicit rejection, Re-encryption, Key confirmation, and Replace input $H(\mu, ctxt)$ of key derivation with $H(\mu)$

Encapsulation

1. Choose a uniformly random message μ
2. $c = \mathsf{Encrypt}_1(pk, \mu)$
3. $K = \mathsf{KDF}(\mu, c)$

(c)

Decapsulation

1. $\mu' = \mathsf{Decrypt}_1(sk, c)$
2. if $\mu' = \perp$ return $\mathsf{KDF}(t, c)$ (a.k.a implicit rejection)
3. else return $K = \mathsf{KDF}(\mu', c)$

where t is a random value in a secret key

(d)

Scheme - FO transformation

Two steps of FO transformation

- Hashing U
  - Many variants of $U^{\perp}$
    - Implicit rejection, Re-encryption, Key confirmation, and Replace input $H(\mu, ctxt)$ of key derivation with $H(\mu)$

Encapsulation

1. Choose a uniformly random message μ
2. $c = \mathsf{Encrypt}_1(pk, \mu)$
3. $K = \mathsf{KDF}(\mu, c)$

(e)

Decapsulation

1. $\mu' = \mathsf{Decrypt}_1(sk, c)$
2. if $c \neq \mathsf{Encrypt}_1(pk, \mu')$ return $\mathsf{KDF}(t, c)$ (a.k.a re-encryption)
3. else return $K = \mathsf{KDF}(\mu', c)$

(f)

Scheme - FO transformation

Two steps of FO transformation

- Hashing U
  - Many variants of $\mathsf{U}^{\perp}$
    - Implicit rejection, Re-encryption, Key confirmation, and Replace input $H(\mu, ctxt)$ of key derivation with $H(\mu)$

Encapsulation

1. Choose a uniformly random message μ
2. $c_1 = \mathsf{Encrypt}_1(pk, \mu)$
3. $K = \mathsf{KDF}(\mu, c)$
4. $c = (c_1, d)$ where $d = H(\mu)$ is a key confirmation value

(g)

Decapsulation

1. $\mu' = \mathsf{Decrypt}_1(sk, c)$
2. if $c \neq \mathsf{Encrypt}_1(pk, \mu')$ or $d \neq H(\mu')$ return $\mathsf{KDF}(t, c)$
3. else return $K = \mathsf{KDF}(\mu', c)$

(h)

Scheme - FO transformation

Two steps of FO transformation

- Hashing U
  - Many variants of $U^{\perp}$
    - Implicit rejection, Re-encryption, Key confirmation, and Replace input $H(\mu, ctxt)$ of key derivation with $H(\mu)$

Encapsulation

1. Choose a uniformly random message μ
2. $c_1 = \mathsf{Encrypt}_1(pk, \mu)$
3. $K = \mathsf{KDF}(\mu)$
4. $c = (c_1, d)$ where $d = H(\mu)$ is a key confirmation value

(i)

Decapsulation

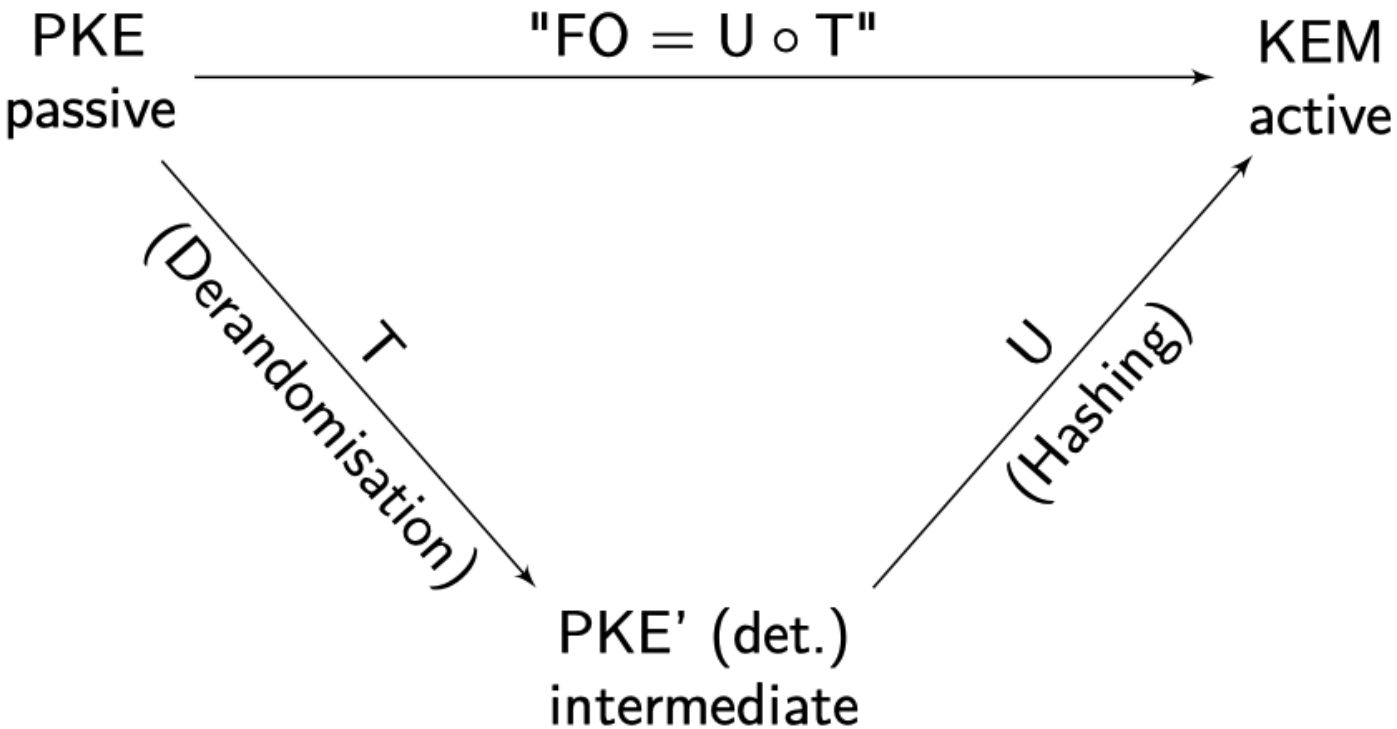
1. $\mu' = \mathsf{Decrypt}_1(sk, c)$
2. if $c \neq \mathsf{Encrypt}_1(pk, \mu')$ or $d \neq H(\mu')$ return $\mathsf{KDF}(t)$
3. else return $K = \mathsf{KDF}(\mu')$

(j)

Scheme - FO transformation

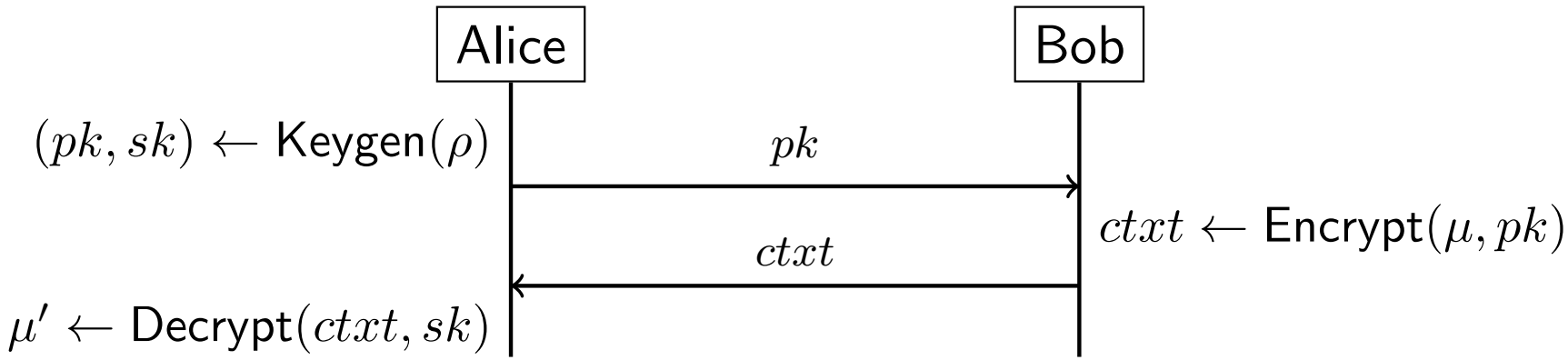
Security reduction of FO transformation

- Security proof on ROM (Random Oracle Model) and QROM (Quantum ROM)



Scheme - Smaug IND-CPA PKE

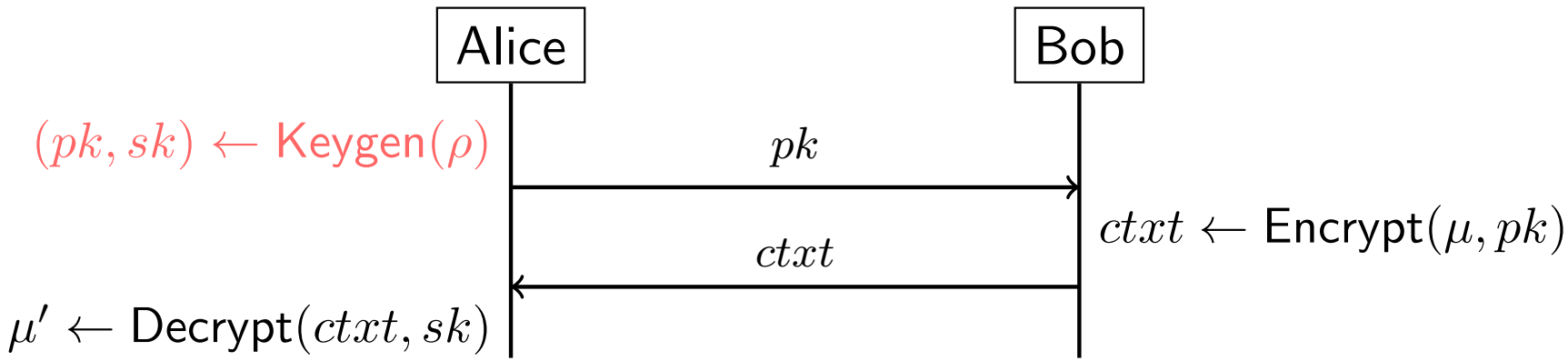
- Communication



- Notation
  - $\mathcal{R} = \mathbb{Z}[x]/(x^N + 1) = \{a_0 + a_1 x^1 \cdots + a_{N-1} x^{N-1} \mid a_i \in \mathbb{Z}\}$
  - $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \{b_0 + b_1 x^1 \cdots + b_{N-1} x^{N-1} \mid b_i \in \mathbb{Z}_q\}$
  - $\lfloor \boldsymbol{a} \rceil$: Round each coefficients of polynomial $a_i(x)$ in a vector $\boldsymbol{a}$
  - HWT$_h$: Sample a ternary polynomial having h non-zero coefficients (a.k.a hamming weight sampling)
  - DG$_\sigma$: Discrete Gaussian sampling with a standard deviation σ

Scheme - Smaug IND-CPA PKE

- Communication



- Keygen

$$\mathbf{A} \leftarrow \mathcal{R}_q^{k \times k}$$

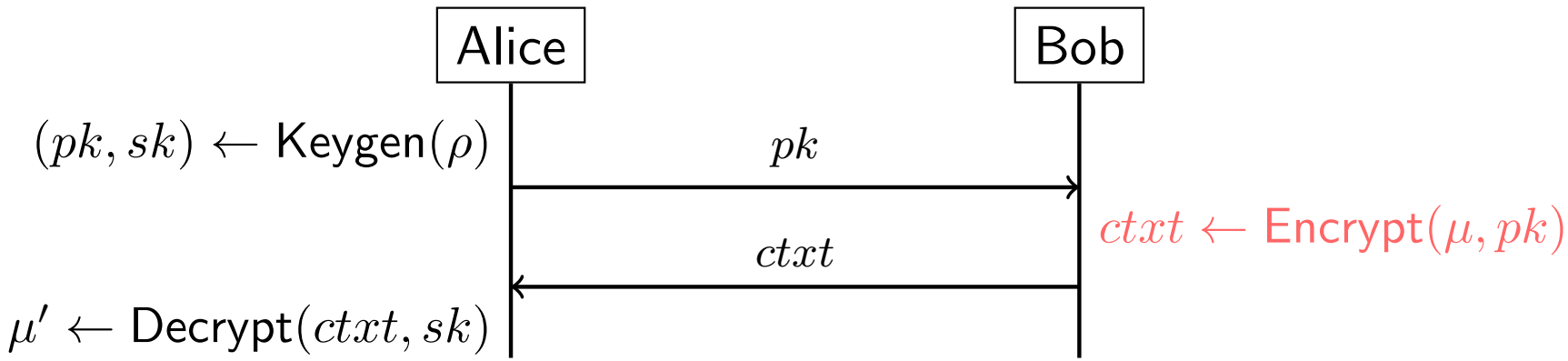
$$\mathbf{s} \leftarrow \mathsf{HWT}_{h_s} \in \mathcal{R}_\eta^k \text{ and } \mathbf{e} \leftarrow \mathsf{DG}_\sigma \in \mathcal{R}_q^k$$

$$\mathbf{b} = -\mathbf{A}^T \cdot \mathbf{s} + \mathbf{e} \in \mathcal{R}_q^k$$

$$pk = (\mathbf{A}, \mathbf{b}),\ sk = \mathbf{s}$$

Scheme - Smaug IND-CPA PKE

- Communication

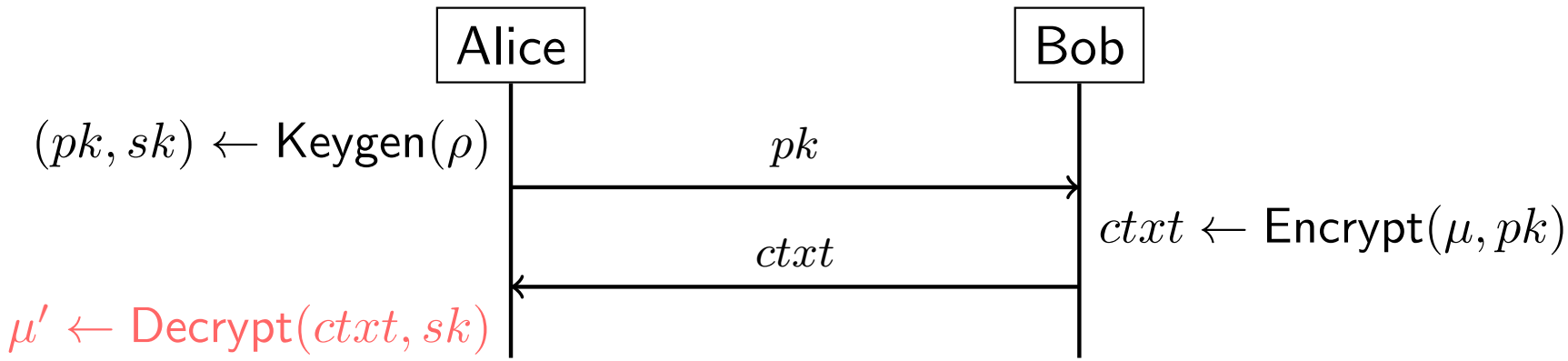


- Encrypt

$$\mathbf{r} \leftarrow \mathsf{HWT}_{h_r} \in \mathcal{R}_q^k$$
$$\mathbf{c}_1 = \lfloor p/q \cdot (\mathbf{A} \cdot \mathbf{r}) \rceil$$
$$c_2 = \lfloor p/q \cdot (\mathbf{b}^T \cdot \mathbf{r} + q/t \cdot \mu) \rceil$$
$$ctxt = (\mathbf{c}_1, c_2) \in \mathcal{R}_p^k \times \mathcal{R}_p$$

Scheme - Smaug IND-CPA PKE

- Communication



- Decrypt

$$\mu' = \lfloor t/p \cdot (c_2 + \mathbf{c}_1^T \cdot \mathbf{s}) \rceil$$

Scheme - Smaug IND-CCA KEM

Apply variant of FO (Fujisaki-Okamoto) transformation [FO99].

- Security reduction in ROM and QROM [HHK17, SXY18].
- Hash of public key in Encrypt
  - Random coin and prevention of multi-target attacks

$$
\begin{aligned}
&1:\ \mu \leftarrow \{0,1\}^{256}\\
&2:\ ctxt := \mathsf{CPAPKE.Encrypt}\big(pk, \mu; \mathrm{G}(\mu, \mathrm{H}(pk))\big)\\
&3:\ K := \mathrm{KDF}\big(\mu, \mathrm{H}(ctxt)\big)
\end{aligned}
$$

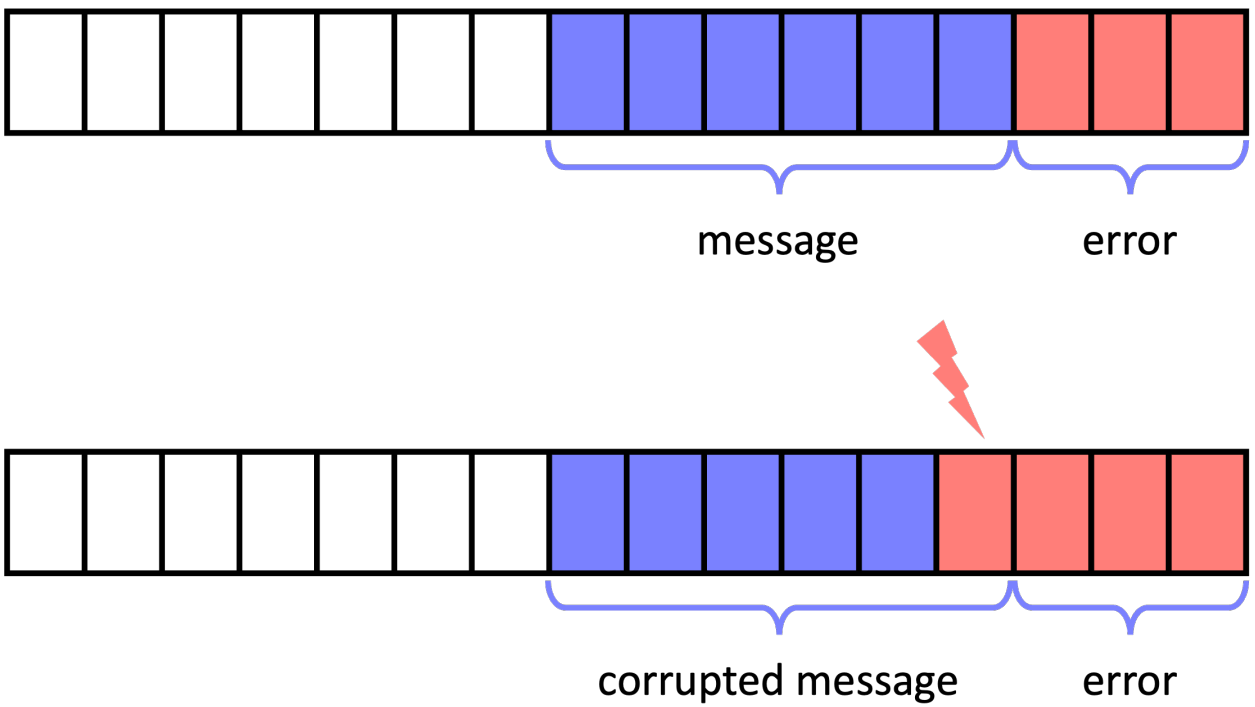
- Re-encryption and Hash of ciphertext in key derivation

$$
\begin{aligned}
&1:\ \mu' := \mathsf{CPAPKE.Decrypt}(sk.\mathbf{s}, ctxt)\\
&2:\ ctxt' := \mathsf{CPAPKE.Encrypt}\big(pk, \mu'; \mathrm{G}(\mu', \mathrm{H}(pk))\big)\\
&3:\ \textbf{if } ctxt = ctxt' \textbf{ then}\\
&4:\quad \textbf{Return } K' = \mathrm{KDF}\big(\mu, \mathrm{H}(ctxt)\big)\\
&5:\ \textbf{else}\\
&6:\quad \textbf{Return } K' = \mathrm{KDF}\big(sk.d, \mathrm{H}(ctxt)\big)\\
&7:\ \textbf{end if}
\end{aligned}
$$

- No additional hash in ciphertext like RLizard
  - Reduce size of ciphertext

Correctness - error bounds

Error bounds



Correctness - error bounds

Correctness equation

- A ciphertext $\mathbf{c}_1$ and c_2 can be expressed with errors:

$$\mathbf{c}_1 = \lfloor p/q \cdot (\mathbf{A} \cdot \mathbf{r}) \rceil = p/q \cdot (\mathbf{A} \cdot \mathbf{r} + \mathbf{e}_1)$$
$$c_2 = \lfloor p/q \cdot \left(\mathbf{b}^T \cdot \mathbf{r} + q/t \cdot \mu\right) \rceil = p/q \cdot (\mathbf{b}^T \cdot \mathbf{r} + e_2) + p/t \cdot \mu$$

- Then, in decryption

$$\left\lfloor \frac{t}{p} \cdot (c_2 + \mathbf{c}_1^T \cdot \mathbf{s}) \right\rceil = \left\lfloor \frac{t}{q} \cdot \mathbf{r}^T \cdot (\mathbf{A}^T\mathbf{s} + \mathbf{b}) + \mu + \frac{t}{q} \cdot (\mathbf{s}^T\mathbf{e}_1 + e_2) \right\rceil$$
$$= \mu + \underline{\left\lfloor \frac{t}{q} \cdot \left(\mathbf{r}^T\mathbf{e} + \mathbf{s}^T\mathbf{e}_1 + e_2\right) \right\rceil}.$$

Correctness - error bounds

Decryption failure probability

- To recover the message μ correctly, the ∞-norm of error must be smaller than $q/2t$.
- We define a decryption failure probability (DFP) δ:

$$\delta = \Pr\left[\left\|\mathbf{r}^T\mathbf{e} + \mathbf{s}^T\mathbf{e}_1 + e_2\right\|_\infty > q/2t\right].$$

- DFP estimation
  - How to test and result

```
# Env setting
$ docker pull sagemath/sagemath
$ docker run -itd -name lattice-estimator sagemath/sagemath
$ docker copy ./Smaug_security_fail_prob_estimator lattice-estimator:/home/sage
$ docker exec -it lattice-estimator /bin/bash

# In docker container
$ sage

sage: pip install pandas
sage: import Smaug_estimator_final

# If you want to run only DFP, comment lines 16-76 and run
```

(k)

```
sage: import Smaug_estimator_final
Decryption failure prob: 2^-132.7
====================
Decryption failure prob: 2^-136.1
====================
Decryption failure prob: 2^-174.5
====================
```

(l)

Security

Security estimation and DFP of SMAUG parameter sets.

- Core-SVP and DFP

Parameters sets Target security	SMAUG128 I	SMAUG192 III	SMAUG256 V
Classical core-SVP Quantum core-SVP	120.0 105.6	181.7 160.9	264.5 245.2
Dec. fail. prob.*	-132.7	-136.1	-174.5

*Decryption failure probability δ is in $\log_2$ scale.

- Beyond the core-SVP

Parameters sets Target security	SMAUG128 I	SMAUG192 III	SMAUG256 V
Arora-Ge	741.3(598.0)	964.4(636.5)	-
BKW	144.7(133.7)	202.0(190.7)	274.6(256.9)
Meet-LWE	164.3(143.7)	213.8(192.4)	283.2(254.7)

Security

The Core-SVP

- Most widely used as a measurement criterion for security level.
- Known attacks for core-SVP
  - Primal attack
  - Dual attack
- Based on [BDGL16] method, it takes
  - for classical: $2^{0.292\beta+o(\beta)}$ time and
  - for quantum: $2^{0.257\beta+o(\beta)}$ time where β is the block size of β-BKZ reduction algorithm.

Security

Beyond the core-SVP security

- Arora-Ge [AG11, ACF+14] and BKW (Blum, Kalai, and Wasserman) [BKW03]
  - Algebraic and combinatorial attack
  - Test results

```
ADPS16_classical
== Smaug128: LWE ==
Algorithm functools.partial(<function dual_hybrid at 0x7fff9a5ceb00>, red_cost_model=<reduction.ADPS16 object at 0x7fffa13e7100>, mitm_optimization=True)
) failed with β = 79 > d = 76
arora-gb            :: rop: ≈2^829.4, m: ≈2^360.2, dreg: 106, t: 4, mem: ≈2^475.0, tag: arora-gb, ϑ: ≈2^354.4, ζ: 373, |S|: ≈2^246.9, prop: ≈2^-105.3
bkw                 :: rop: ≈2^144.7, m: ≈2^132.7, mem: ≈2^133.7, b: 13, t1: 0, t2: 13, ℓ: 12, #cod: 420, #top: 1, #test: 91, tag: coded-bkw
usvp                :: rop: ≈2^120.0, red: ≈2^120.0, δ: 1.003908, β: 411, d: 869, tag: usvp
```

- Meet-LWE
  - Meet-in-the-Middle attack improved by [May21].
  - For a key space size $\mathcal{S}$, run in time $\mathcal{S}^{0.3}$ asymptotically.
  - However, there is a constant increasing the estimation result.

Characteristics

Main characteristics of Smaug

- Take both advantages of LWE and LWR
  - LWE: Conservative security guarantee for a key pair
  - LWR: Efficient encryption and decryption
- Sparse polynomial
  - Smaller secret key, and fast and simple polynomial multiplication
  - Security of LWE with sparse secret is guaranteed [CHK+16].

Differences with Kyber

- Hard problems
  - Smaug uses LWR to encrypt and decrypt for high efficiency.
- Advantage and security analysis of a sparse polynomial
  - Faster polynomial multiplication

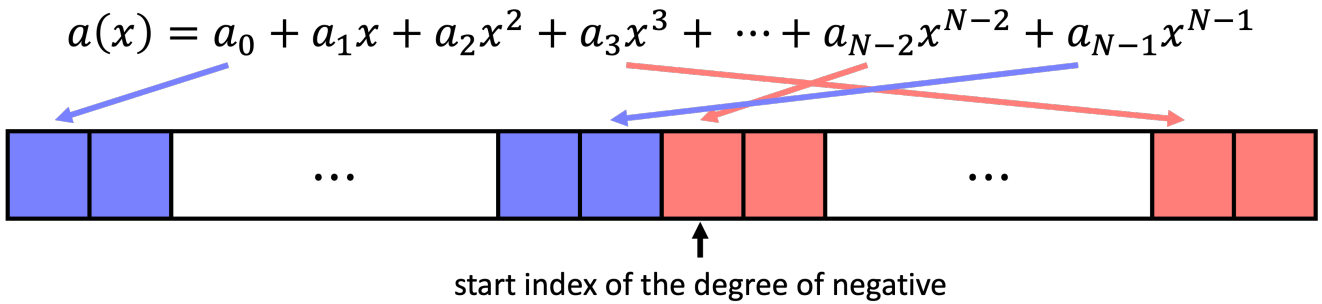
Security level	I	III	V
Smaug	15,048	24,048	44,568
Kyber	32,832	35,424	41,868

  - Security analysis of hybrid attacks are different with Kyber.

Implementation

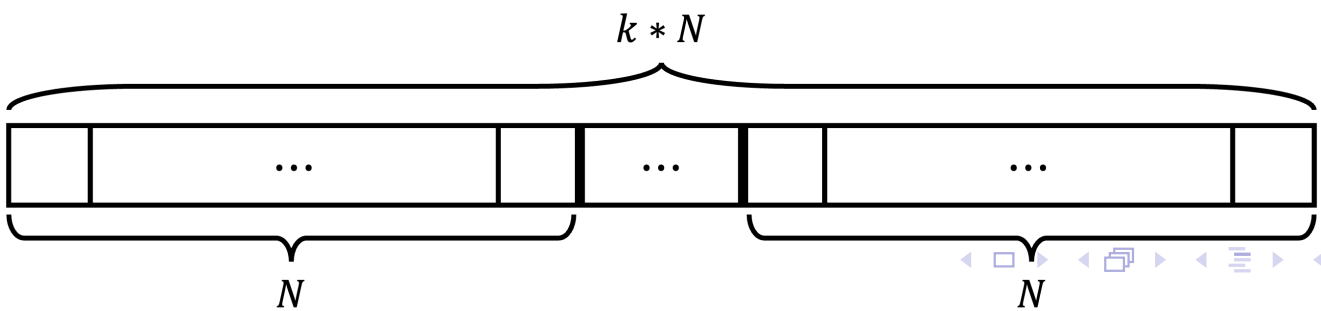
Degree index polynomial form

- Given a sparse polynomial $a(x)$ of degree N with hamming weight h, store the degrees of non-zero coefficient into the array like below:



Hamming weight sampling

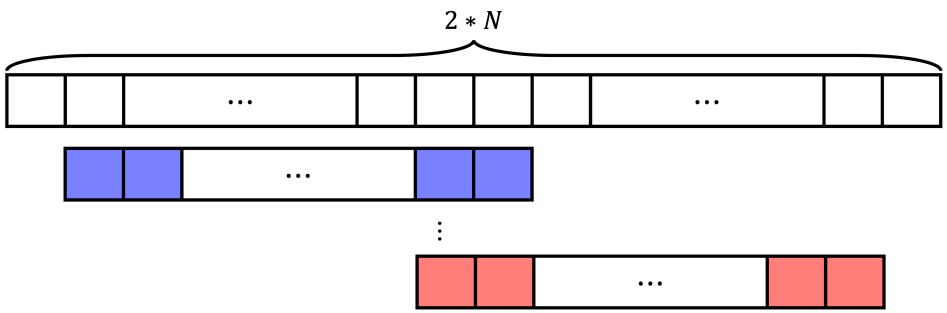
- Sample a sparse polynomial with hamming weight h from a set of polynomials of degree $k * N$.
- Divide into k polynomials of degree N.



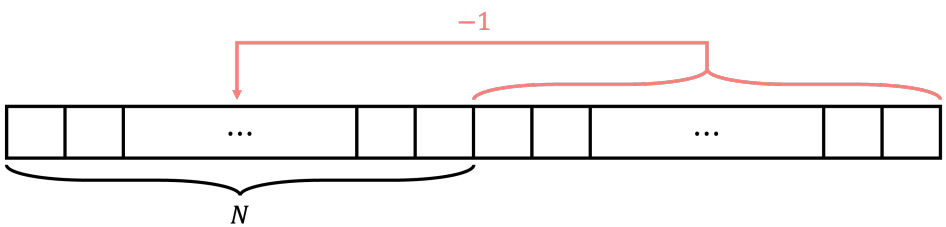
Implementation

Polynomial multiplication

- Polynomial multiplication only with add and subtraction in $2 * N$ space
  - Like schoolbook multiplication, add or subtract the polynomial after increasing the degree:



  - Quotient the polyonmial



Benchmark

Performance (cpu cycles)

- GNU/Linux with Linux kernel version 5.4.0
- AMD Ryzen 3700x
- The compiler gcc 9.4.0 with -O3 and -fomit-frame-pointer.

	Smaug128	Smaug192	Smaug256
N	256	256	256
k	2	3	5
(p, q)	(1024, 256)	(2048, 256)	(2048, 256)
Keygen	68,148	106,020	201,600
Encap	78,012	121,248	210,744
Decap	84,348	125,460	225,972
Secret key	172 (844)	230 (1318)	208 (2000)
Public key	672	1088	1792
Ciphertext	768	1024	1536

Change log

1. Update the parameter sets of the security level IIIand V.

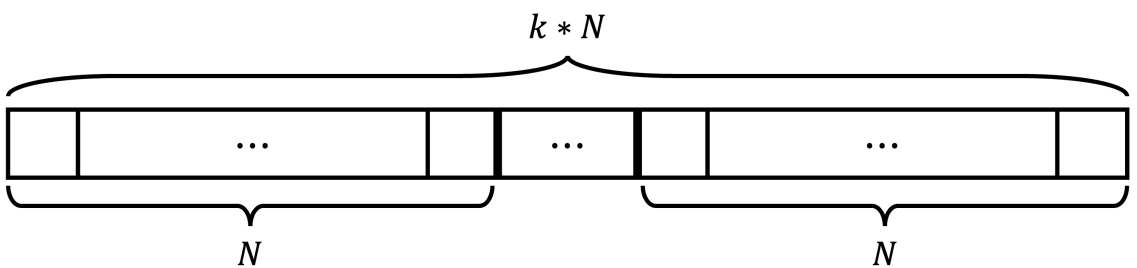
	III	V
q	1024	1024
h_s	150	145
h_r	147	140
σ	1.0625	1.0625
sk	182	177
pk	992	1632

$\rightarrow$

	III	V
q	2048	2048
h_s	198	176
h_r	151	160
σ	1.453713	1.0625
sk	230	208
pk	1088	1792

2. Update additional security estimation on some algebraic and combinatorical attacks
   - Arora-Ge [AG11],Coded-BKW [GJS15], and Meet-LWE [May21]

3. Update hamming weight sampling implementation



Comparison - simple

Performance comparison

- GNU/Linux with Linux kernel version 5.4.0
- AMD Ryzen 3700x
- The compiler gcc 9.4.0 with -O3 and -fomit-frame-pointer.

	Smaug128	Ring-Lizard128	TiGER128	Kyber512
Keygen	68,148	95,614	45,504	104,044
Encap	78,012	107,298	77,544	133,252
Decap	84,348	117,170	109,512	163,493
Secret key	172	289	528	832
Public key	672	1312	480	800
Ciphertext	768	2080	768	768

Thanks

References I

[ACF+14] Martin Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret.
Algebraic algorithms for lwe problems.
2014.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe.
Post-quantum key {Exchange—A} new hope.
In 25th USENIX Security Symposium (USENIX Security 16), pages 327–343, 2016.

[AG11] Sanjeev Arora and Rong Ge.
New algorithms for learning in presence of errors.
In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, Automata, Languages and Programming, pages 403–415, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BCD+16] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila.
Frodo: Take off the ring! practical, quantum-secure key exchange from lwe.
In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 1006–1018, 2016.

References II

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven.
New directions in nearest neighbor searching with applications to lattice sieving.
In Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, pages 10–24. SIAM, 2016.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé.
Crystals-kyber: a cca-secure module-lattice-based kem.
In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pages 353–367. IEEE, 2018.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman.
Noise-tolerant learning, the parity problem, and the statistical query model.
Journal of the ACM (JACM), 50(4):506–519, 2003.

[CHK+16] Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son.
A practical post-quantum public-key cryptosystem based on spLWE.
In International Conference on Information Security and Cryptology, pages 51–74. Springer, 2016.

References III

[CKLS18] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song.
Lizard: Cut off the tail! a practical post-quantum public-key encryption from lwe and lwr.
In International Conference on Security and Cryptography for Networks, pages 160–177. Springer, 2018.

[DKSRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren.
Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem.
In International Conference on Cryptology in Africa, pages 282–305. Springer, 2018.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto.
Secure integration of asymmetric and symmetric encryption schemes.
In Michael Wiener, editor, Advances in Cryptology — CRYPTO' 99, pages 537–554, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski.
Coded-bkw: Solving lwe using lattice codes.
In Annual Cryptology Conference, pages 23–42. Springer, 2015.

References IV

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz.
A modular analysis of the fujisaki-okamoto transformation.
In Yael Kalai and Leonid Reyzin, editors, Theory of Cryptography, pages 341–371, Cham, 2017. Springer International Publishing.

[LKL+18] Joohee Lee, Duhyeong Kim, Hyungkyu Lee, Younho Lee, and Jung Hee Cheon.
Rlizard: Post-quantum key encapsulation mechanism for iot devices.
IEEE Access, 7:2080–2091, 2018.

[LP11] Richard Lindner and Chris Peikert.
Better key sizes (and attacks) for lwe-based encryption.
In Cryptographers' Track at the RSA Conference, pages 319–339. Springer, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev.
On ideal lattices and learning with errors over rings.
In Annual international conference on the theory and applications of cryptographic techniques, pages 1–23. Springer, 2010.

[May21] Alexander May.
How to meet ternary lwe keys.
In Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41, pages 701–731. Springer, 2021.

References V

[Reg05] Oded Regev.
On lattices, learning with errors, random linear codes, and cryptography.
In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pages 84–93, 2005.

[SXY18] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa.
Tightly-secure key-encapsulation mechanism in the quantum random oracle model.
In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2018, pages 520–551, Cham, 2018. Springer International Publishing.