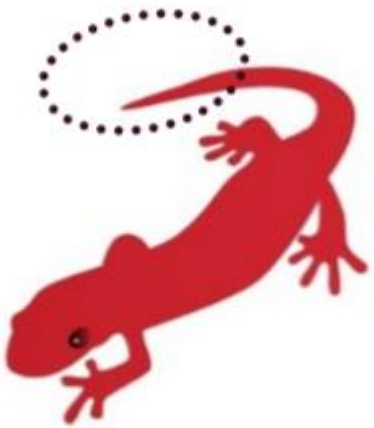
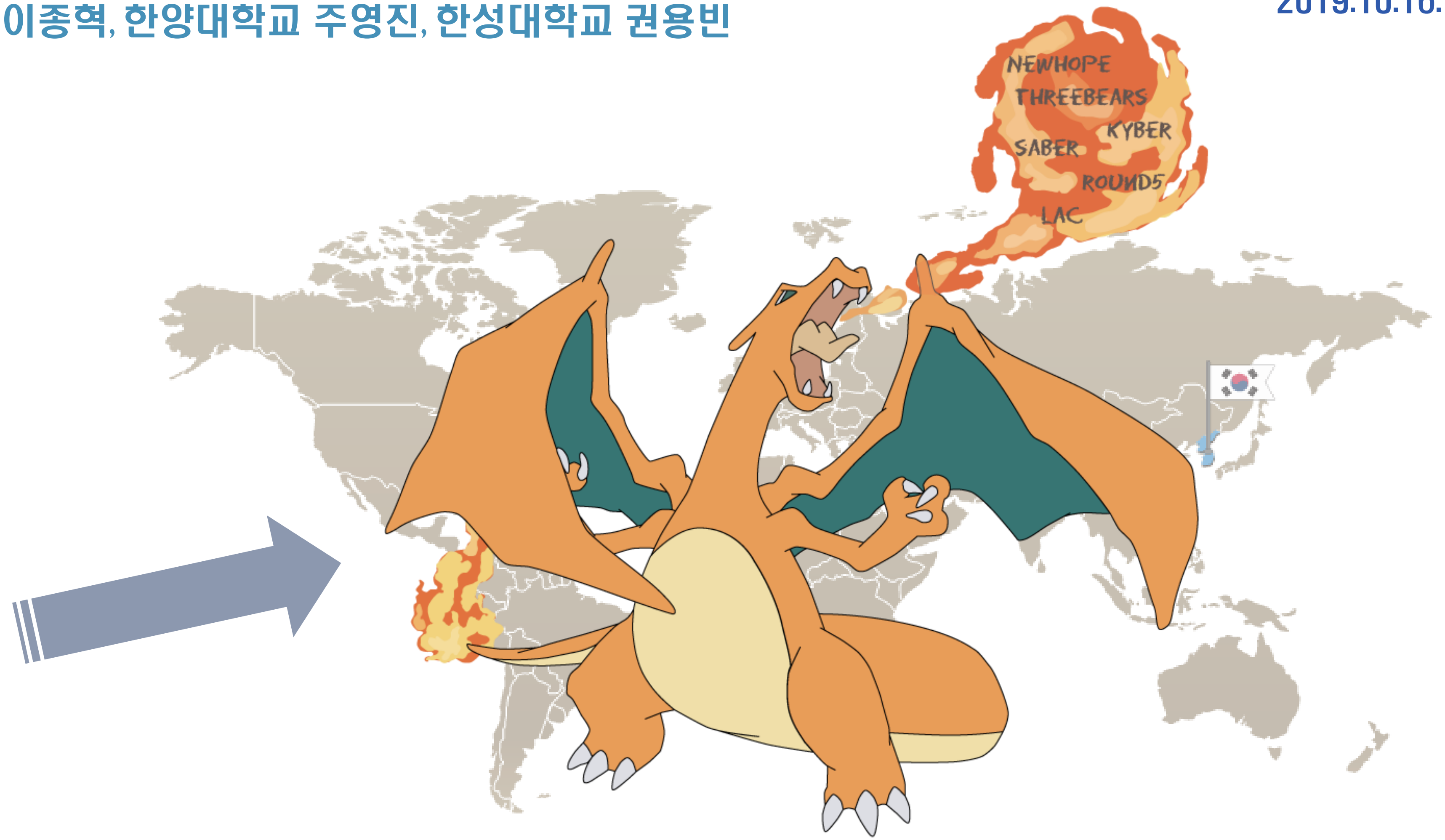
LizarMong: Excellent KEM/PKE based on RLWE and RLWR

서울대학교 정치곤, 국민대학교 이종혁, 한양대학교 주영진, 한성대학교 권용빈

2019.10.16.





<fig. refer: https://csrc.nist.gov/CSRC/media/Presentations/Lizard/images-media/Lizard-April2018.pdf >

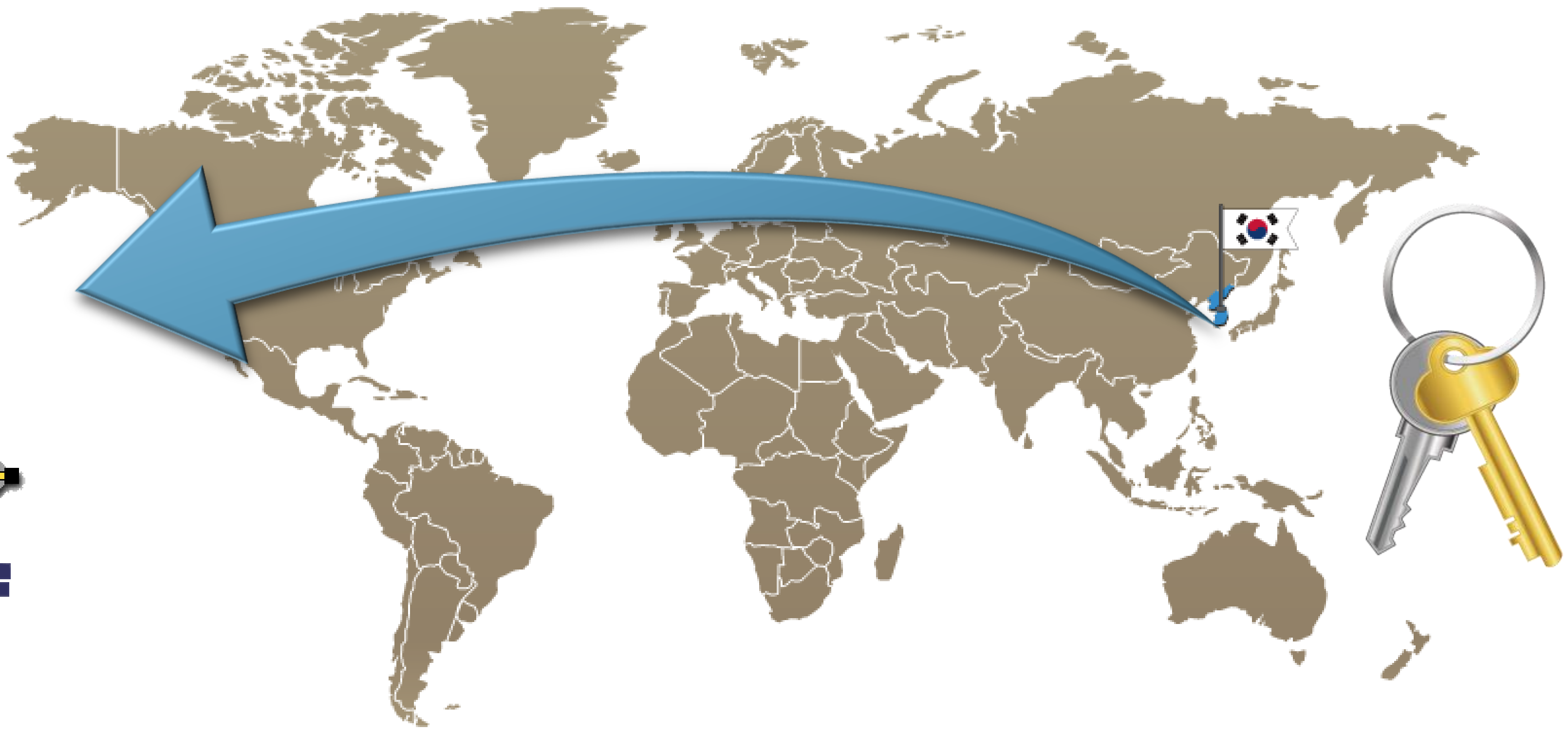
CONTENTS

MOTIVATION

Motivation

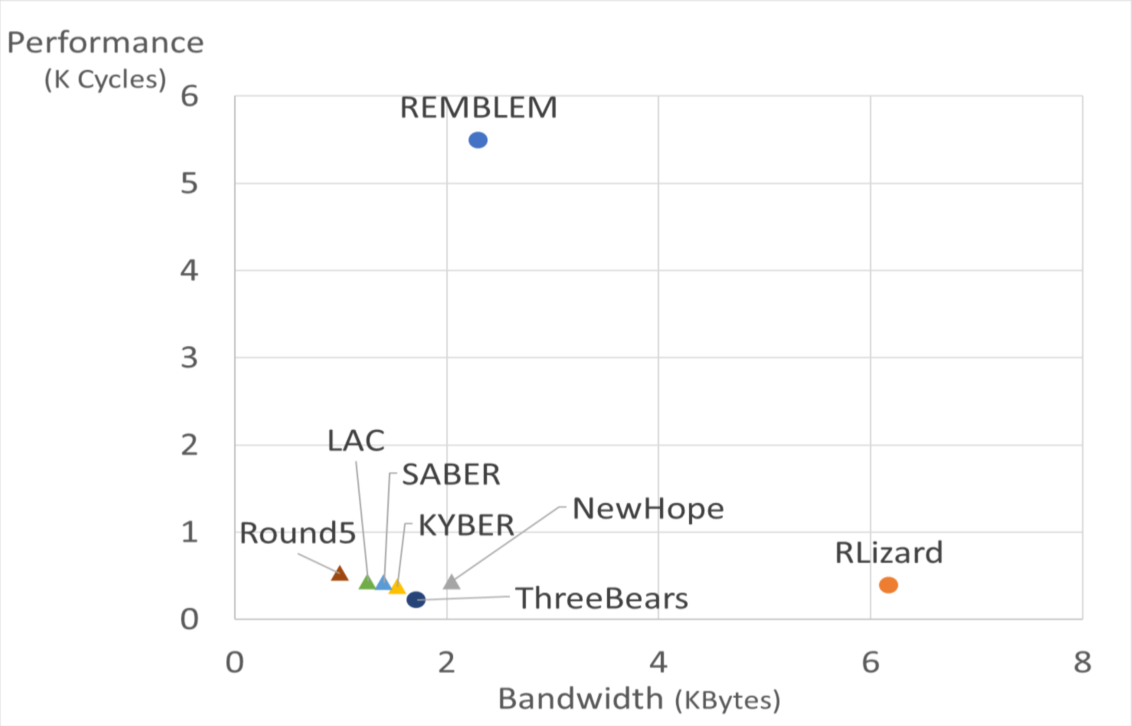
ISO

I E T F



국산 양자내성 암호를 국제 표준으로!!

What is the Gap?



< Performance and Bandwidth of 128-bit security KEM >

Algorithm	Classical security (log)	Correctness (log)
RLizard	147	-188
REMBLEM	128	-140
NewHope	112	-213
KYBER	111	-178
SABER	125	-120
Round5	128	-88
LAC	147	-116
ThreeBears	154	-156

< Clamed security and Correctness of 128-bit security KEM >

- RLizard: Bandwidth
- REMBLEM: Performance (+ only support128-bit security level)

* Performance(keygen+enc+dec):The result of measuring optimal implementation code submitted to NIST in the same machine(i7-9700K and GCC –O3).
* Bandwidth(pk+ctx), Security, and Correctness: Referenced the paper submitted to NIST.

NIST candidate algorithms are perfect? (classification)

Large Modulus (2^{12}~2^{14})

NewHope RLWE

KYBER MLWE

SABER MLWR

Bandwidth ↑
Performance ↓
Correctness ↑

⇩

Fast Multiplication
Compress tech.

Small Modulus (2^{8}~2^{12})

LAC RLWE

Bandwidth ↓
Performance ↑
Correctness ↓

⇩

Error-correcting code
Tiny secret and errors.

Round5 RLWR

$\mathbb{Z}_q/X^{n+1}-1$
For small bandwidth

$\mathbb{Z}_q/X^n+1$
Common choice

ThreeBears I-MLWE

< General pros and cons >

Good performance and bandwidth, well studies than others.

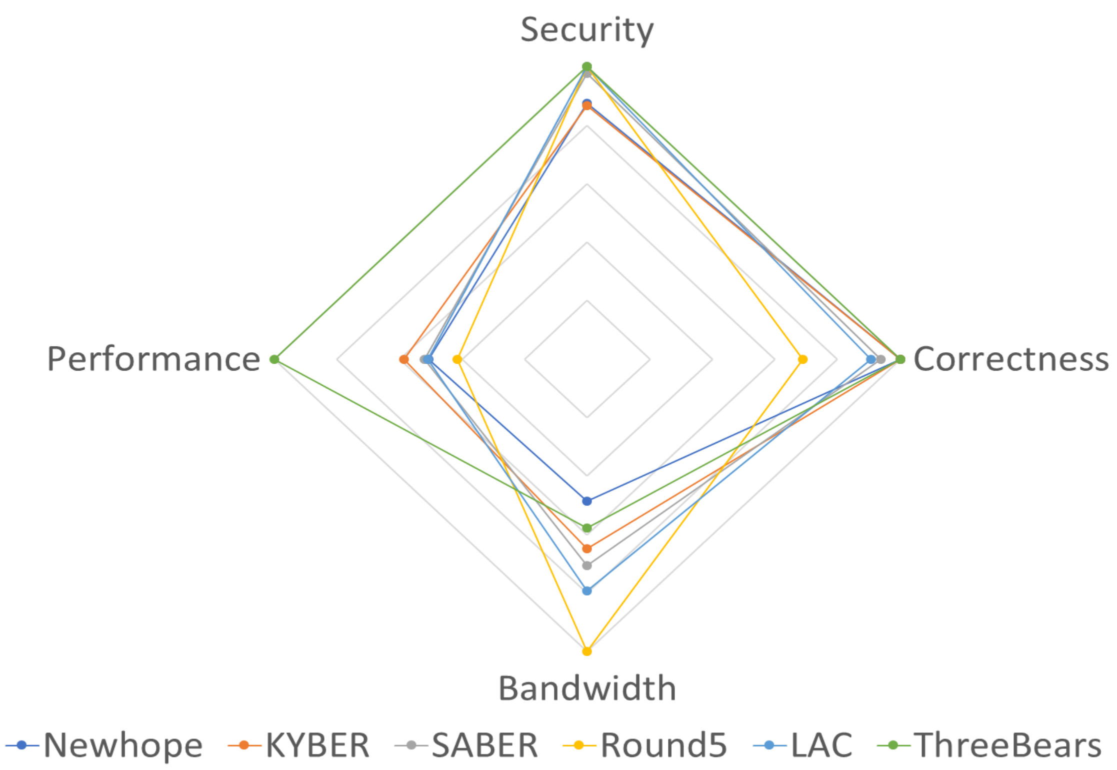
Reduce bandwidth, but more computation than same dimension RLWE.

Better performance and bandwidth compare to RLWE and MLWE.

Overall looks good. However, the hardness is unclear in practical parameters.

NIST candidate algorithms are perfect?

< Compare to 128-bit security KEM >



- Which is the best?
  - All evaluation criteria are important.
  - NIST said "Still open to mergers."
- Most of latest studies are not included.
  - Especially, Side-channel attacks?
  - Error in each bit occurs independently?

★Goal: Making the excellent algorithm of all aspect based on RLizard ★

DETAIL TO LIZARMONG

Specification of LizarMong

Design elements

- Reduce the bandwidth and maintain the RLizard's strengths.
- Minimized known side-channel attack points.

Compare	Underlying Problem	Ring	Compress	Modulus	ECC	Distributions	
						Secret	Error
LizarMong	RLWE+RLWR	$\mathbb{Z}_q/X^n+1$	**Yes**	**Small (fixed 2^8)**	**XE5**	Uniform sparse ternary	**Binomial (std≈0.7)**
RLizard	〃	〃	No	Small ($2^{10\sim12}$)	None	〃	Gaussian CDT (std≈1.15)
Why?	Key: conservative Enc/Dec: Fast	Fast / secure	Bandwidth	Bandwidth, Performance	Correctness, Side-channel	Correctness, Performance	Side-channel Correctness, Performance
Proved	-	-	Common in NIST's Alg.	[PRSD17]	[Saa17]	-	[ADPS16]

Specification of LizarMong

IND-CCA2 KEM

Algorithm 4 IND-CCA2-KEM.KeyGen

Input: The set of public *parameters*

Output: Public Key $pk = (Seed_a||\mathbf{b})$, Private Key $sk = (\mathbf{s}||\mathbf{u})$

$\boxed{Seed_a \xleftarrow{\$} \{0,1\}^{256}}$
$\boxed{\mathbf{a} \leftarrow \mathtt{SHAKE256}(Seed_a, n/8)}$
$\mathbf{s} \xleftarrow{\$} HWT_n(h_s)$, $\mathbf{u} \xleftarrow{\$} \{0,1\}^n$, $\mathbf{e} \leftarrow \boxed{\psi^n_{cb}}$
$\mathbf{b} \leftarrow -\mathbf{a} * \mathbf{s} + \mathbf{e}$
$\boxed{pk \leftarrow (Seed_a||\mathbf{b})}$, $sk \leftarrow (\mathbf{s}||\mathbf{u})$
return pk, sk

Algorithm 6 IND-CCA2-KEM.Decapsulation

Input: pk, sk, Ciphertext $\mathbf{c}$, *parameters*

Output: Shared Key $\mathbf{K}$

$\mathbf{c_{1a}}, \mathbf{c_{1b}}, \mathbf{d} \leftarrow \text{Parsing}(\mathbf{c})$
$\hat{\delta}' \leftarrow \lfloor (2/p) \cdot (\boxed{(p/k)} \cdot \mathbf{c_{1b}} + \mathbf{c_{1a}} * \mathbf{s}) \rceil$
$\boxed{\hat{\delta} \leftarrow \text{eccDEC}(\hat{\delta}')}$
$\hat{\mathbf{r}} \leftarrow H(\hat{\delta})$, $\hat{\mathbf{d}} \leftarrow H'(\hat{\delta})$, $\boxed{\hat{\delta}'' \leftarrow \text{eccENC}(\hat{\delta})}$
$\hat{\mathbf{c}} \leftarrow \lfloor (p/q) \cdot \mathbf{a} * \hat{\mathbf{r}} \rceil \;||\; \lfloor \boxed{(k/q)} \cdot ((q/2) \cdot \hat{\delta}'' + \mathbf{b} * \hat{\mathbf{r}}) \rceil \;||\; \mathbf{d}$
if $\mathbf{c} \neq \hat{\mathbf{c}}$ **then** $\mathbf{K} \leftarrow G(\mathbf{c}, \mathbf{u})$ **else** $\mathbf{K} \leftarrow G(\mathbf{c}, \hat{\delta})$
return $\mathbf{K}$

Algorithm 5 IND-CCA2-KEM.Encapsulation

Input: pk, *parameters*

Output: Ciphertext $\mathbf{c} = (\mathbf{c_1}||\mathbf{d})$, Shared Key $\mathbf{K}$

$\delta \xleftarrow{\$} \{0,1\}^{sd}$, $\boxed{\delta' \leftarrow \text{eccENC}(\delta)}$
$\mathbf{r} \leftarrow H(\delta)$ and $\mathbf{d} \leftarrow H'(\delta)$
$\mathbf{c_{1a}} \leftarrow \lfloor (p/q) \cdot \mathbf{a} * \mathbf{r} \rceil$ and $\mathbf{c_{1b}} \leftarrow \lfloor \boxed{(k/q)} \cdot ((q/2) \cdot \delta' + \mathbf{b} * \mathbf{r}) \rceil$
$\mathbf{c_1} \leftarrow \mathbf{c_{1a}}||\mathbf{c_{1b}}$
$\mathbf{K} \leftarrow G(\mathbf{c_1}, \mathbf{d}, \delta)$
return $\mathbf{c} = (\mathbf{c_1}||\mathbf{d}), \mathbf{K}$

< Parameters for each security level >

parameters	n	q	p	k	h_s	h_r	d	sd	cb
Comfort (128-bit)	512	256	64	16	128	128	256	256	1
Strong (256-bit)	1024	256	64	16	128	128	512	512	1

< Bandwidth for each security level (unit: bytes) >

Type	Comfort			Strong		
	CPA PKE	KEM	PKE	CPA PKE	KEM	PKE
Ciphertext	640	672	704	1,280	1,344	1,408
Public key	544	544	544	1,056	1,056	1,056
Secret key	512	544	512	1,024	1,088	1,024

Security analysis

Security Proof

RLWE / RLWR

↑ Lemma 1.

RLizard
(IND-CPA PKE)

↑ Theorem 1.
Under the assumption that SHAKE256 is ROM

LizarMong
(IND-CPA PKE)

↑ A variants of Fujisaki-Okamoto Transformation [HHK17]

LizarMong
(IND-CCA2 KEM/PKE)

Cryptanalytic attacks

- Assume the attacks are using BKZ.sieve.
- Computational complexity measure core SVP.
  - use 'online LWE estimator' [Alb17].
  - Consider Dual and Primal attack like RLizard.

Table 3: Computational complexity of best RLWE and RLWR attacks

Parameters	Claim Security	Attacks		Classical	Quantum
Comfort	NIST Category 1 (AES 128-bit)	Primal	RLWE	**133**	**121**
			RLWR	144	131
		Dual	RLWE	165	154
			RLWR	180	170
Strong	NIST Category 5 (AES 256-bit)	Primal	RLWE	**256**	**236**
			RLWR	269	249
		Dual	RLWE	304	275
			RLWR	328	301

Correctness analysis

- Estimating the Correctness considering the dependency of each bit error.
  - The correctness of all RLWE estimates on the assumption that errors occur independently.
  - The independent assumption was disproved [DVV19]; Especially improper using ECC.

- Decryption failure is when satisfied $|e * r + s * f + g| \geq \frac{q}{4} - \frac{q}{2p}$.
  - $f = a * r - (q/p)c_1; g = v - \hat{v}; v = \lfloor (p/q) \cdot ((q/2) \cdot \mathbf{M}' + \mathbf{b} * \mathbf{r} \rceil$, $\hat{v} = v \ll (\log p - \log k)$

- $\Pr[Fail] \approx \sum_{\|S\|,\|C\|} \left(1 - Binom(d, l_m, p_b)\right) \cdot \Pr[\|S\|] \cdot \Pr[\|C\|]$
  - $S = (\mathbf{s}, \mathbf{e})^T, C = (\mathbf{f}, \mathbf{r})^T$, $Binom(k, n, p) = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i}$, $p_b = \Pr[F_0 \mid \|S\|, \|C\|]$

Prameters	without ECC	with XE5(5bit ECC)
Comfort	2^{-37}	2^{-179}
Strong	2^{-68}	2^{-302}

Resistance to known side-channel attacks

- We investigated the known major side-channel attacks and the points they exploited.

Attack methods	Attacks	Attack Points
Timing Attacks	[PH16]	Modulus operation doing or not.
	[KH18]	CDT sampling's branch.
Differential Attacks	[PPM17]	INV NTT operation
	[ATT$^+$18]	Multiplication using secrets.
	[HCY19]	Multiplication using secrets.
Template Attacks	[BFM$^+$18]	Multiplication using secrets.
Fault Attacks	[EFGT18]	Error sampling function.
	[RRB$^+$19]	Same distribution for secret and error sampling.
Cache Attacks	[BHLY16]	CDT sampling's table look up.

AND, ADD, and SHIFT instead of Modulus Op.

Do not use NTT

Devised sparse polynomial multiplication with Hiding

Check the final loop index

Distributions of secret and error are different

Replaced with centered binomial distribution

- Our strategy
  - First, ruled out the targeted by the known attacks during the design element selection.
  - Second, internalizes efficient countermeasures for unavoidable vulnerabilities.

Evaluation

- Compare to RLizard,
  - Band.: 80~85% smaller / Perfor.: 2.1~3.8x faster
- Compare to NIST's candidate Algorithms,
  - Band.: 3~41% smaller / Perfor.: 2.0~8.3x faster

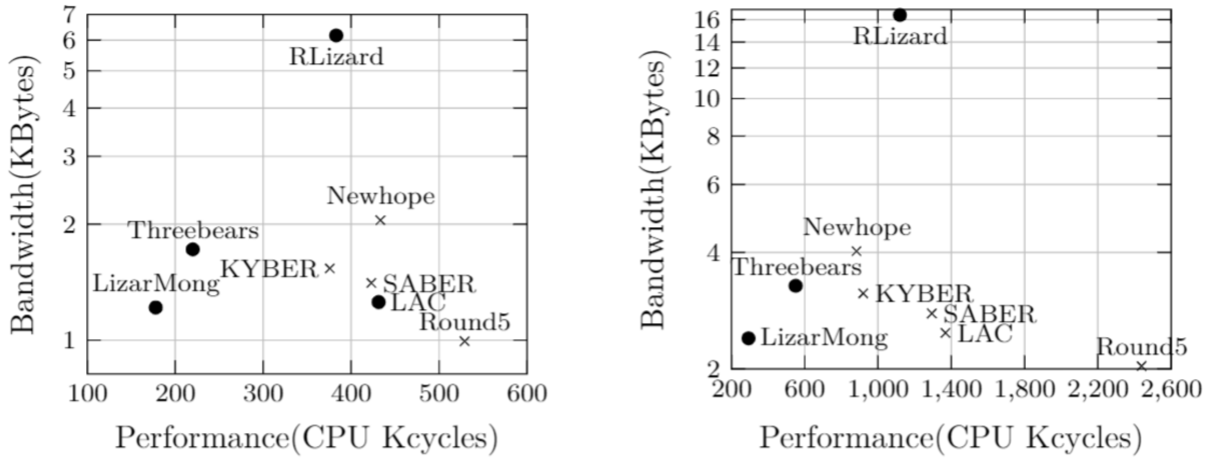
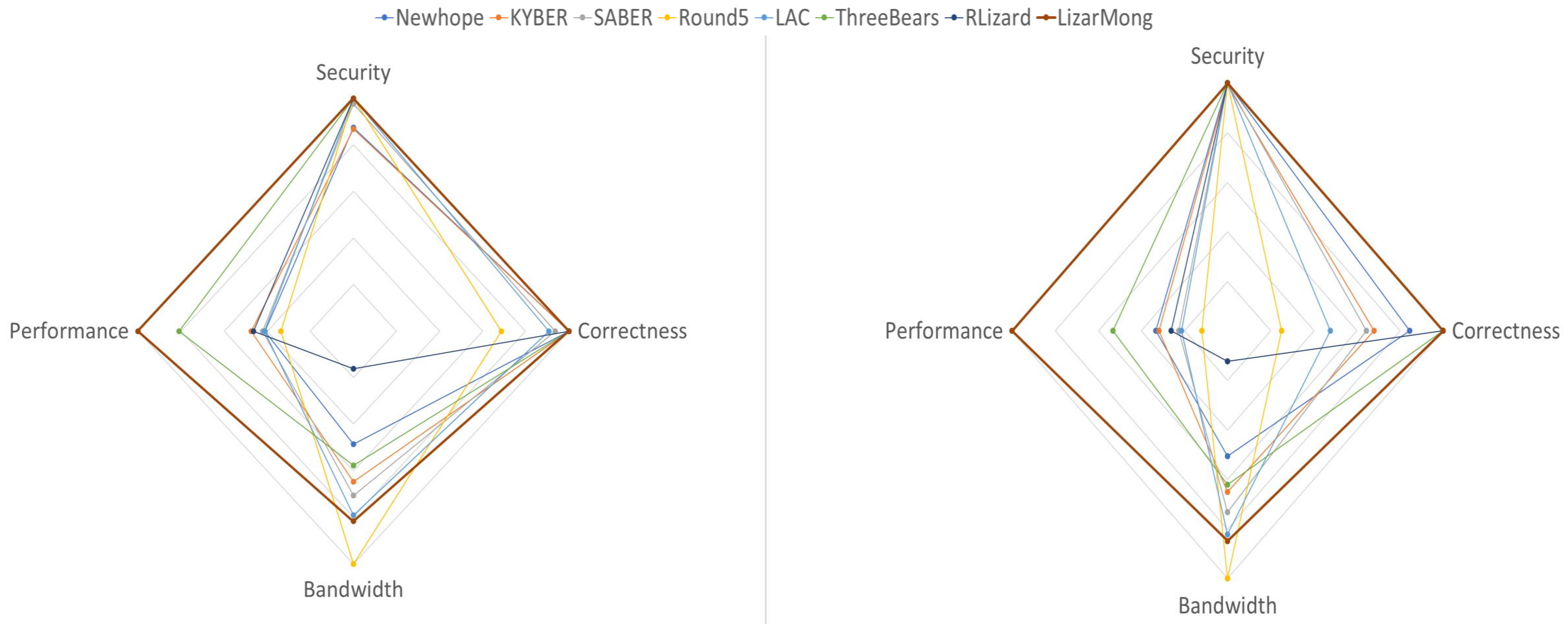


Figure 2: Comparison of bandwidth and performance based on IND-CCA2 KEM(Round5 is IND-CPA). (left) 128-bit security level (right) 256-bit security level (Note: • are algorithms with security and correctness similar to each security level, and × are not.)

Table 5: Comparison KEM with NIST candidate algorithms and RLizard

Algorithms	**Security** (log)	**Correctness** (log)	**Bandwidth** (Bytes)	**Performance** (K cycles)	
				Enc+Dec	**KeyGen**
LizarMong	133	−179	1,216	133.9	44.0
	256	−302	2,400	231.5	62.1
RLizard	147	−188	6,176	217.8	165.3
	195	−246	8,240	416.9	232.7
	318	−306	16,448	737.3	382.7
Newhope	112	−213	2,048	329.6	103.6
	257	−216	4,032	673.5	209.2
KYBER	111	−178	1,536	278.2	97.5
	181	−164	2,272	463.6	174.3
	254	−174	3,136	656.0	263.1
SABER	125	−120	1,408	316.9	106.1
	203	−136	2,080	587.6	213.6
	283	−165	2,784	934.8	359.2
LAC	147	−116	1,256	341.2	90.0
	286	−143	2,244	840.1	235.6
	320	−122	2,480	1,101.6	266.6
Round5 (IND-CPA)	128	−88	994	384.4	114.6
	193	−117	1,639	857.2	311.3
	256	−64	2,035	1,794.9	643.4
Threebears	154	−156	1,721	167.8	52.1
	235	−206	2,501	271.4	91.9
	314	−256	3,281	402.5	148.2

Conclusion

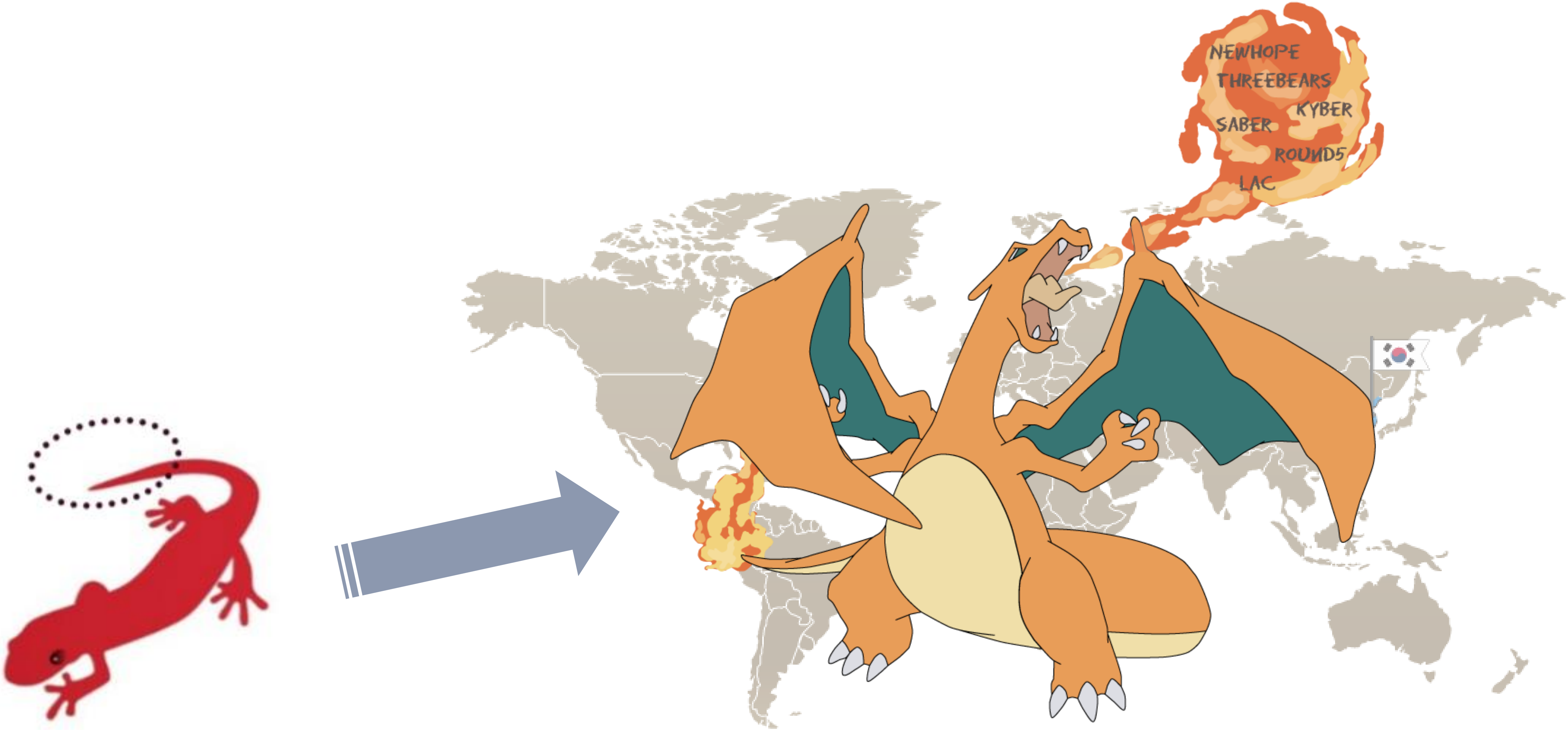


< 128bit security >

< 256bit security >

★ **LizarMong is excellent of all aspect! Let's Go International standard!** ★

Have any Questions? Thank you!



<fig. refer: https://csrc.nist.gov/CSRC/media/Presentations/Lizard/images-media/Lizard-April2018.pdf >

REFERENCES

[ATT+18] Aydin Aysu, Youssef Tobah, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. Horizontal side-channel vulnerabilities of post-quantum key exchange protocols. In 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 81–88. IEEE, 2018.

[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.

[Alb17] MR Albrecht. A sage module for estimating the concrete security of learning with errors instances (2017).

[BFM+18] Joppe W Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Assessing the feasibility of single trace power analysis of frodo. In International Conference on Selected Areas in Cryptography, pages 216–234. Springer, 2018.

[BHLY16] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme. In International Conference on Cryptographic Hardware and Embedded Systems, pages 323–345. Springer, 2016.

[DVV19] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. The impact of error dependencies on ring/mod-lwe/lwr based schemes. In International Conference on Post-Quantum Cryptography, pages 103–115. Springer, 2019.

[EFGT18] Thomas Espitau, Pierre-Alain Fouque, Benoit Gerard, and Mehdi Tibouchi. Loop-abort faults on lattice-based signature schemes and key exchange protocols. IEEE Transactions on Computers, 67(11):1535–1549, 2018.

[HCY19] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. Correlation power analysis on ntru prime and related countermeasures. IACR Cryptology ePrint Archive, 2019:100, 2019.

[KH18] Suhri Kim and Seokhie Hong. Single trace analysis on constant time cdt sampler and its countermeasure. Applied Sciences, 8(10):1809, 2018

[PH16] Aesun Park and Dong-Guk Han. Chosen ciphertext simple power analysis on software 8-bit implementation of ring-lwe encryption. In 2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST), pages 1–6. IEEE, 2016.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In International Conference on Cryptographic Hardware and Embedded Systems, pages 513–533. Springer, 2017.

[PRSD17] Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-lwe for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing. pp.461–473. ACM (2017)

[RRB+19] Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "not used" once-practical fault attack on pqm4 implementations of nist candidates. In International Workshop on Constructive Side-Channel Analysis and Secure Design, pages 232–250. Springer, 2019.

[Saa17] Markku-Juhani O. Saarinen. Hila5: On reliability, reconciliation, and error correction for ring-lwe encryption. Cryptology ePrint Archive, Report 2017/424, 2017. https://eprint.iacr.org/2017/424.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Theory of Cryptography Conference, pages 341–371. Springer, 2017.

Compress techniques

- All NIST candidate algorithms commonly use compression techniques.
  - Public-key: Sending only the *gen_a_seed* instead of $\boldsymbol{a} \in R_q$, and recovers using a hash.

    * pk size: $2n \log q \ \rightarrow |gen\_\boldsymbol{a}\_seed| + n \log q$.
  - Ciphertext: Discarding a few bits of LSB in $\boldsymbol{c}_2$.

    * ctx size: $2n \log p \ \rightarrow n \log p + n \log k$, where $k < p$ is compress modulus

- How does compress affect the scheme?

- How is the size of *gen_a_seed*?
  - NIST candidate algorithms use 128 or 256bit. we choose 256bit from a conservative.
- Ciphertext compress reduce the correctness?
  - Yes! However, we already include it in calculation of the failure rate.

Small modulus fixed at 2^8

- ☐ Reduce bandwidth: to make lattice dimension n and RLWE modulus q small.
  - Since the ring $\mathbb{Z}_q/X^n + 1$, the smaller n is 256; however, difficult to satisfy security.
- ☐ Therefore our choice is only to make q smaller.

❖ How does small q affect the scheme?

- ☐ Harmful to RLWE hardness?
  - No! [PRSD17] showed that RLWE is hardness on any integer Ring!
  - LAC also use q=251.
- ☐ Reduce the Correctness?
  - Yes! Because decryption fails when $|error \geq q/4 - q/2p|$.
  - LizarMong adopts error-correcting code (ECC) to solve this problem.

Error-correcting code, XE5

- ☐ According to our analysis, 4-5 bit error correction capability is required.
- ☐ We adopted XE5 [saa17] that is specialized in the RLWE.
  - 256bit message p, 234bit parity check r, codeword $c = p||r$, correction capability is 5bit.

❖ How does XE5 affect the scheme?

- ☐ Performance overhead? Yes! But, it is very small (only 600 cycles).
- ☐ Side-channel attacks?
  - No! [saa17] argues XE5 resist timing attack as avoid table look-up and branch;
- ☐ The impact of error dependencies?
  - Yes! The calculation is improper when the error-correcting code is used [DVV19].
  - To solve, calculate the failure rate under the assumption that error occurs dependently.

Resistance known side-channel attacks (1/2)

☐ According to the 1st strategy, against known cache and timing attacks, also some differential and fault attacks.

- Modulus operation ☞ choice all modulus are power-of-two. So, AND and ADD instead of it.
- CDT branch and table look-up ☞ CDT was replaced with centered binomial distribution.
- Same distribution for error and secret ☞ Distribution of secret and error are different.
- INV-NTT ☞ do not use NTT.

❖ How does this design choice affect the scheme?

☐ Centered binomial distribution ?

- Proved similarity with the Gaussian distribution. Most of NIST candidates used it.

☐ Each distribution for error and secret?

- Original RLWE defines each distribution for error and secret.

Resistance known side-channel attacks (2/2)

- According to the 2nd strategy, added countermeasures against the remaining attacks.

Attack methods	Attacks	Attack Points
~~Timing Attacks~~	[PH16]	Modulus operation doing or not.
	[KH18]	CDT sampling's branch.
Differential Attacks	[PPM17]	~~INV NTT operation~~
	[ATT+18]	Multiplication using secrets.
	[HCY19]	Multiplication using secrets.
Template Attacks	[BFM+18]	Multiplication using secrets.
Fault Attacks	[EFGT18]	Error sampling function.
	[RRB+19]	~~Same distribution for secret and error sampling.~~
~~Cache Attacks~~	[BHLY16]	CDT sampling's table look-up.

Algorithm 7 Sparse Polynomial Multiplication with Hiding Countermeasure

Input: $\mathbf{a}=\sum_{i=0}^{n-1}[\mathbf{a}]_i\cdot x^i\in R_q$, $\mathbf{r}=\sum_{i=0}^{g-1}x^{[\mathbf{r}]_i}+\sum_{i=g}^{h}\left(-x^{[\mathbf{r}]_i}\right)\in R_q$

Output: $\mathbf{v}=\mathbf{a}*\mathbf{r}=\sum_{i=0}^{n-1}[\mathbf{v}]_i\cdot x^i\in R_q$

1: initialize $\mathbf{v}$ to zero polynomial ▷ size of $\mathbf{v}=2n$
2: $R\xleftarrow{\$}\{0,1,\ldots,g-1\}$ ▷ random starting index
3: **for** $i\in\{0,\ldots,g-1\}$, $j\in\{0,\ldots,n-1\}$ **do**
4: $\quad[\mathbf{v}]_{[\mathbf{r}]_{R+i \pmod g}+j}=[\mathbf{v}]_{[\mathbf{r}]_{R+i \pmod g}+j}+[\mathbf{a}]_j$
5: **for** $i\in\{0,\ldots,n-1\}$ **do** $[\mathbf{v}]_i=[\mathbf{v}]_i-[\mathbf{v}]_{n+i}$
6: **return** $\mathbf{v}$

```
unsigned char b0, b1, tmp2[LWE_N/4];
randombytes(tmp2,LWE_N/4);          // tmp2[0]'s 0, 1
for(j=0; j<LWE_N/4; ++j){           // Centered Binom
        b0 = tmp2[j] & 0x01;
        tmp2[j] = tmp2[j] >> 1;
        b1 = tmp2[j] & 0x01;
        pk_b[j*4+0] = b0 -b1;
        tmp2[j] = tmp2[j] >> 1;
        b0 = tmp2[j] & 0x01;
        tmp2[j] = tmp2[j] >> 1;
        b1 = tmp2[j] & 0x01;
        pk_b[j*4+1] = b0 -b1;
        tmp2[j] = tmp2[j] >> 1;
        b0 = tmp2[j] & 0x01;
        tmp2[j] = tmp2[j] >> 1;
        b1 = tmp2[j] & 0x01;
        pk_b[j*4+2] = b0 -b1;
        tmp2[j] = tmp2[j] >> 1;
        b0 = tmp2[j] & 0x01;
        tmp2[j] = tmp2[j] >> 1;
        b1 = tmp2[j] & 0x01;
        pk_b[j*4+3] = b0 -b1;
        tmp2[j] = tmp2[j] >> 1;
}

if (j != (LWE_N/4)) { // fault detecting
        return 3;
}
```