KPQC 공모전 2라운드 격자 기반 알고리즘 안전성 분석

2024.10.22.

Sungshin Women's University

Joohee Lee

Contact: jooheelee@sungshin.ac.kr

CONTENTS

KPQC ROUND 2 –LATTICE-BASED SCHEMES

- Among Round 1 candidates, 3 KEMs and 5 signatures were lattice-based schemes.
- 2 KEMs and 2 signatures are selected as Round 2 candidates

Category	Name	Base problem	Note
KEM	**NTRU+**	NTRU/RLWE	• RLWE with ternary secrets/errors • Analysis Reported (6/14/23)
	SMAUG-T	MLWE/MLWR	• MLWE/MLWR with sparse secrets • Analysis Reported (6/14/24)
Signature	HAETAE	MLWE/MSIS	
	NCC-Sign	RLWE/RSIS	

Improved Meet-LWE Attack via Ternary Trees

Eunmin Lee[1], Joohee Lee[1]*, Yongha Son[1], Yuntao Wang[2]

[1] Sungshin Women's University, Seoul, Republic of Korea
{20211089, jooheelee, yongha.son}@sungshin.ac.kr
[2] The University of Electro-Communications, Tokyo, Japan
y-wang@uec.ac.jp

Abstract. The Learning with Errors (LWE) problem with its variants over structured lattices has been widely exploited in efficient post-quantum cryptosystems. Recently, May [59] suggests the Meet-LWE attack, which poses a significant advancement in the line of work on the Meet-in-the-Middle approach to analyze LWE with ternary secrets.
In this work, we generalize and extend the idea of Meet-LWE by introducing ternary trees, which result in diverse representations of the secrets. More precisely, we split the secrets into three pieces with the same dimension and expand them into a ternary tree to leverage the increased representations to improve the overall attack complexity. We also suggest the matching criteria for the approximate matching of three lists via locality sensitive hash function accordingly. We carefully analyze and optimize the time and memory costs of our attack algorithm exploiting ternary trees, and compare them to those of the Meet-LWE attack. With asymptotic and non-asymptotic comparisons, we observe that our attack provides improved estimations for all parameter settings, including those of the practical post-quantum schemes, compared to the Meet-LWE attack. We also evaluate the security of the Round 2 candidates of the KpqC competition which aims to standardize post-quantum public key cryptosystems in the Republic of Korea and report that the estimated complexities for our attack applied to SMAUG-T are lower than the claimed for some of the recommended parameters.

Keywords: Learning with Errors, Meet-LWE, Meet-in-the-Middle, KpqC Competition

IMPROVED MEET-LWE ATTACK VIA TERNARY TREES

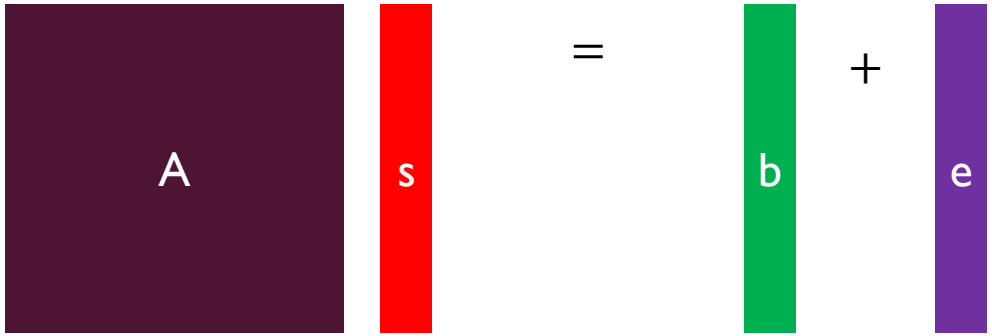
LEE, EUNMIN, JOOHEE LEE, YONGHA SON, AND YUNTAO WANG. "IMPROVED MEET-LWE ATTACK VIA TERNARY TREES." *CRYPTOLOGY EPRINT ARCHIVE* (2024).

TERNARY LWE PROBLEM

[Ternary LWE problem]

Given; $A \in Z_q^{n \times n}, b \in Z_q^n$ such that $A \cdot s = b + e$ for $\boldsymbol{s}, \boldsymbol{e} \in \{\boldsymbol{0}, \pm\boldsymbol{1}\}^{\boldsymbol{n}}$,

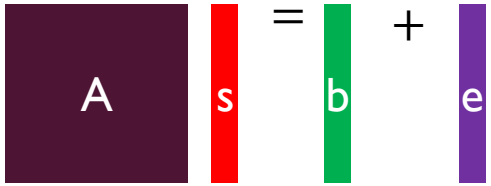
Find; $s \in \{0, \pm 1\}^n$



- Asymptotically, Brute Force < Odlyzko's MitM < Meet-LWE < Ours
- Meet-LWE and our attack are applicable to all KpqC Round 2 Lattice-based KEMs
  - SMAUG-T uses sparse secrets
  - NTRU+ uses the ternary LWE problem

BRUTE FORCE ATTACK FOR TERNARY LWE

- Equation : $A \cdot s = b + e \ \ mod\ q$



[Brute Force]
- Input : $A \in Z_q^{n \times n}, b \in Z_q^n$
- For all $s \in \{0, \pm 1\}^n$:
  - If $A \cdot s - b \in \{0, \pm 1\}^n$ then output s

- $S = 3^n$; search space size for ternary keys
- Running time is $T = S$

ODLYZKO'S MITM

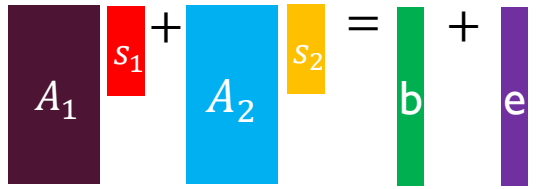
- Equation : $A_1 \cdot s_1 = -A_2 \cdot s_2 + b + e \ \ mod\ q$

  i.e. $A_1 \cdot s_1 \approx -A_2 \cdot s_2 + b \ \ mod\ q$

[Odlyzko's MitM]

- Input : $A = (A_1|A_2) \in Z_q^{n \times n}, b \in Z_q^n$
- For all $s_1 \in \{0, \pm 1\}^{n/2}$:
  - Construct L_1 with entries $(s_1, h(A_1 s_1))$
- For all $s_2 \in \{0, \pm 1\}^{n/2}$:
  - Construct L_2 with entries $(s_2, h(-A_2 s_2 + b))$
- Output $(s_1|s_2)$ with $h(A_1 s_1) = h(-A_2 s_2 + b)$

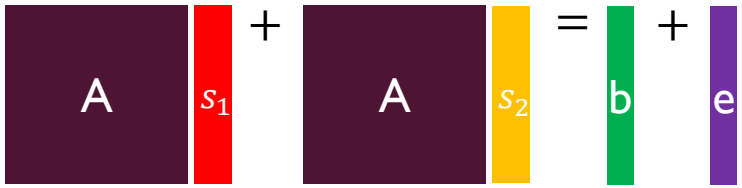
* h: locality sensitive hash



- $S = 3^n$; search space size for ternary keys
- Running time is $T = 3^{n/2} = S^{1/2}$ with same memory

REPRESENTATIONS (HOWGRAVE-GRAHAM, JOUX '10)

- **Idea** ; $s := s_1 + s_2$ for $s_1, s_2 \in \{0, \pm 1\}^n$
  - Allows redundancy



- (1,0,1,-1,0) = (1,0,0,-1,0) + (0,0,1,0,0)

  = (1,0,1,0,0) + (0,0,0,-1,0)

  = (0,0,1,0,0) + (1,0,0,-1,0)

  = (0,0,1,-1,0) + (1,0,0,0,0)

	1	0	-1
REP-0	• 1+0 • 0+1	-	• (-1)+0 • 0+(-1)
REP-1	• 1+0 • 0+1	• 1+(-1) • (-1)+1	• (-1)+0 • 0+(-1)
REP-2	• 1+0 • 2+(-1) • 0+1 • (-1)+2	• 1+(-1) • 2+(-2) • (-1)+1 • (-2)+2	• (-1)+0 • 1+(-2) • 0+(-1) • (-2)+1

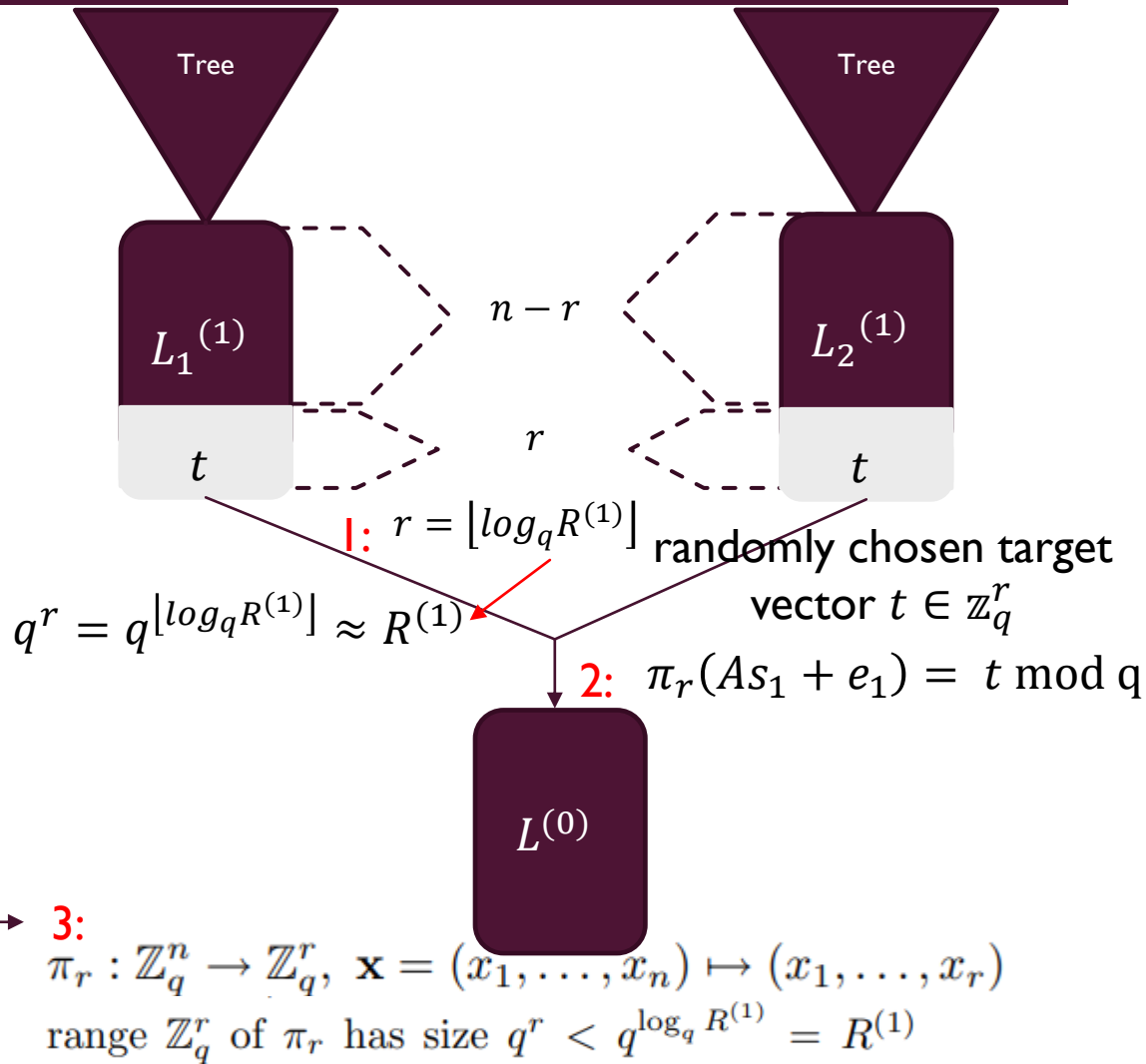
MEET LWE ATTACK [MAY21]

$e = e_2 - e_1$

- Equation: $A_1 \cdot s_1 = -A_2 \cdot s_2 + b + e \; mod \; q$
  i.e. $A_1 \cdot s_1 \approx -A_2 \cdot s_2 + b \; mod \; q$

[Meet LWE (high-level idea)]

- Input : $A \in Z_q^{n \times n}, b \in Z_q^n$
- Choose representation REP-0, REP-1, REP-2
- Guess r coordinates of e (say $\underline{t}$)
- For s_1, construct L_1 with entries $(s_1, h(As_1))$
- For s_2, construct L_2 with entries $(s_2, h(-As_2 + b))$
- Output $s_1 + s_2$ s.t.
  - $\pi_r(As_1 + e_1) = t = \pi_r(-As_2 + b + e_2)$
  - $h(As_1) = h(-As_2 + b)$ for $n - r$ coordinates



MEET LWE ATTACK [MAY21]

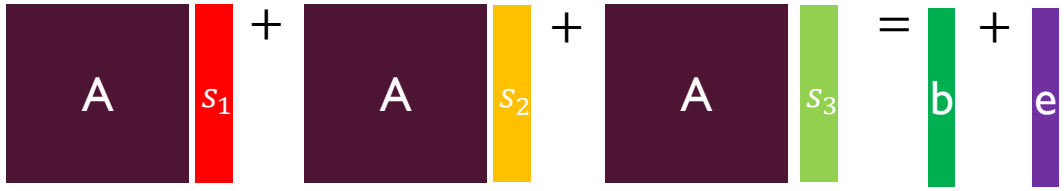
- By using representations $s = s_1 + s_2$, **the number of solutions ($:= R$) increases**
- We can reduce the list (L_1, L_2) sizes with a factor of R (by guessing r coordinates of e), expecting at least one solution exists
  - This strategy can be recursively applied to s_1, s_2, respectively (lists can be obtained by tree-based construction)
- Run-time $T = T_g \cdot T_\ell$ where T_g; guessing complexity, T_ℓ; list construction complexity

(n, q, w)	Odlyzko[bit]	REP-0 [bit]	REP-1 [bit]	REP-2 [bit]
NTRU (821,4096,510)	***T*** 643	**520** = 487+33	**393** = 334+59	**378** = 318+60

Table 1: Illustration of our non-asymptotic improvements.

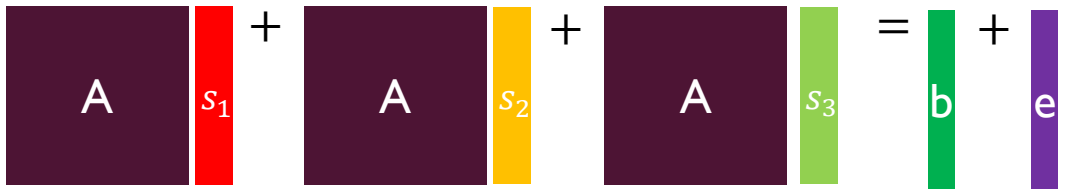
OUR IDEA

- **Idea** ; $s := s_1 + s_2 + s_3$ for $s_1, s_2, s_3 \in \{0, \pm 1\}^n$
  - Increased diversity of representations



	1	0	-1
REP-0	• 1+0+0 • 0+1+0 • 0+0+1	• 0+0+0	• (-1)+0+0 • 0+(-1)+0 • 0+0+(-1)
REP-1-0	• 1+0+0 • 0+1+0 • 0+0+1	• 0+1+(-1) • 0+(-1)+1 • 1+0+(-1) • 1+(-1)+0 • (-1)+0+1 • (-1)+1+0	• (-1)+0+0 • 0+(-1)+0 • 0+0+(-1)
REP-1-1	• 1+0+0 • 0+1+0 • 0+0+1 • 1+1+(-1) • (-1)+1+1 • 1+(-1)+1	• 0+1+(-1) • 0+(-1)+1 • 1+0+(-1) • 1+(-1)+0 • (-1)+0+1 • (-1)+1+0	• (-1)+0+0 • 0+(-1)+0 • 0+0+(-1) • (-1)+1+(-1) • (-1)+(-1)+1 • 1+(-1)+(-1)

OUR IDEA

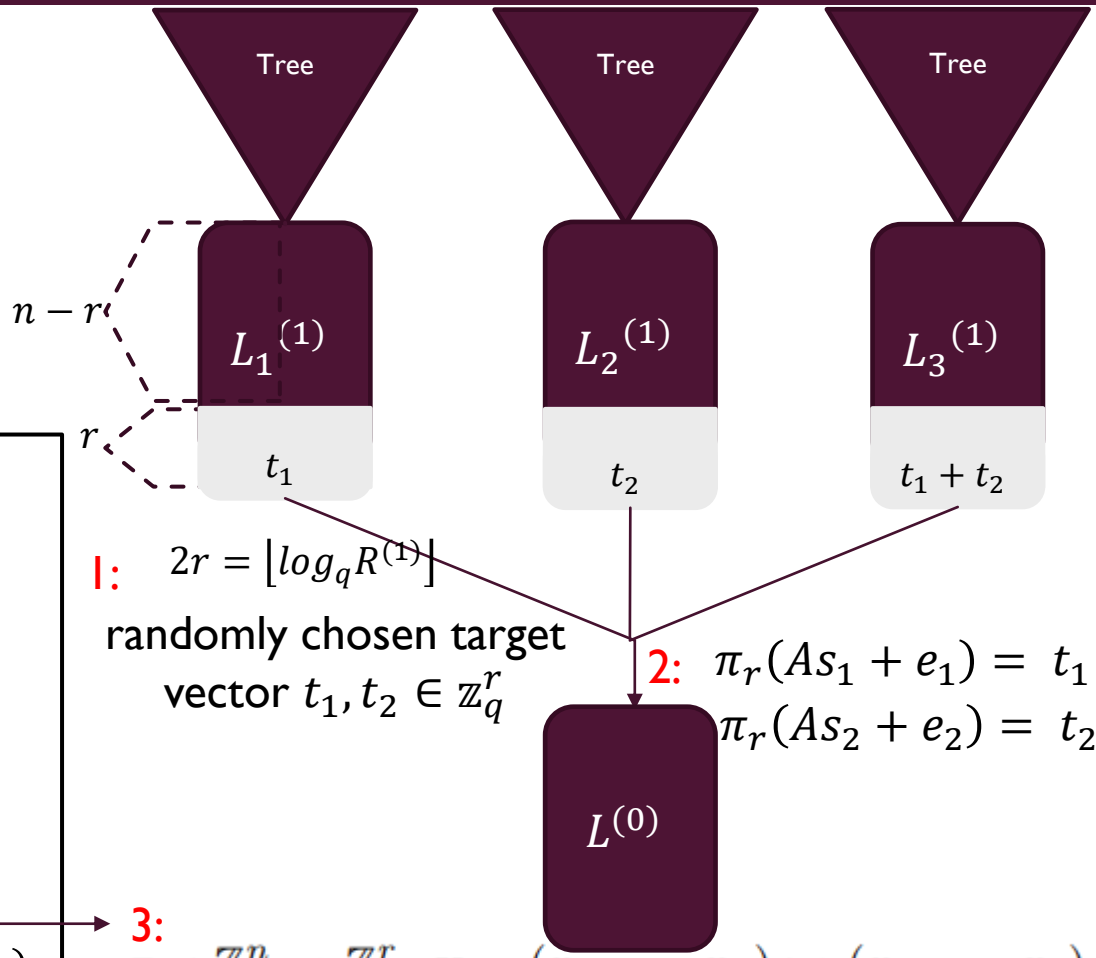


$$e = e_3 - e_1 - e_2$$

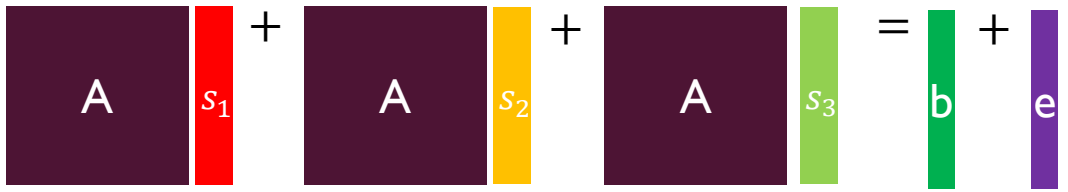
- Equation: $A_1 \cdot s_1 + A_2 \cdot s_2 = -A_3 \cdot s_3 + b + e \bmod q$
  i.e. $A_1 \cdot s_1 + A_2 \cdot s_2 \approx -A_3 \cdot s_3 + b \bmod q$

[Extended Meet-LWE Attack (high-level idea)]

- Input : $A \in Z_q^{n \times n}, b \in Z_q^n$
- Choose representation REP-0, REP-1-0, REP-1-1
- Guess r coordinates of e_1, e_2 (say t_1, t_2)
- For s_1, construct L_1 with entries $(s_1, h(As_1))$
- For s_2, construct L_2 with entries $(s_2, h(As_2))$
- For s_3, construct L_3 with entries $(s_3, h(b - As_3))$
- Output $s_1 + s_2 + s_3$ s.t.
  - $\pi_r(As_1 + e_1) = t_1, \pi_r(As_2 + e_2) = t_2$
  - $\pi_r(-As_3 + b + e_3) = t_1 + t_2 = \pi_r(As_1 + e_1) + \pi_r(As_2 + e_2)$
  - $h(As_1) \oplus h(As_2) = h(-As_3 + b)$ for $n - r$ coordinates



OUR IDEA

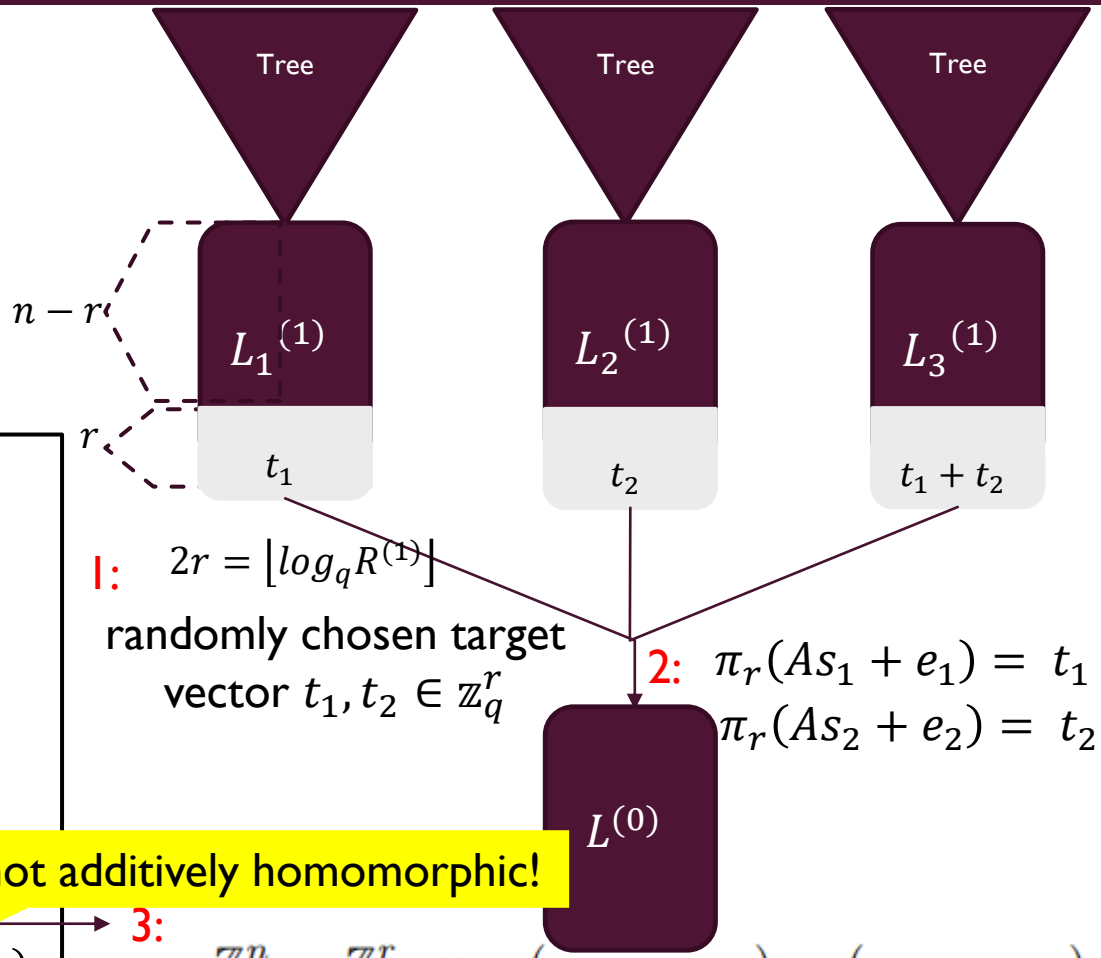


$$e = e_3 - e_1 - e_2$$

- Equation: $A_1 \cdot s_1 + A_2 \cdot s_2 = -A_3 \cdot s_3 + b + e \bmod q$
  i.e. $A_1 \cdot s_1 + A_2 \cdot s_2 \approx -A_3 \cdot s_3 + b \bmod q$

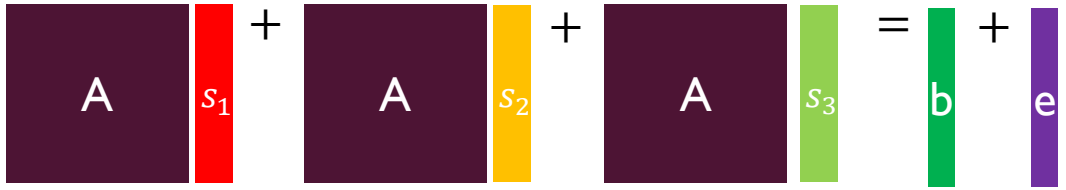
[Extended Meet-LWE Attack (high-level idea)]

- Input : $A \in Z_q^{n \times n}, b \in Z_q^n$
- Choose representation REP-0, REP-1-0, REP-1-1
- Guess r coordinates of e_1, e_2 (say t_1, t_2)
- For s_1, construct L_1 with entries $(s_1, h(As_1))$
- For s_2, construct L_2 with entries $(s_2, h(As_2))$
- For s_3, construct L_3 with entries $(s_3, h(b - As_3))$
- Output $s_1 + s_2 + s_3$ s.t.
  - $\pi_r(As_1 + e_1) = t_1, \pi_r(As_2 + e_2) = t_2$
  - $\pi_r(-As_3 + b + e_3) = t_1 + t_2 = \pi_r(As_1 + e_1) + \pi_r(As_2 + e_2)$
  - $h(As_1) \oplus h(As_2) = h(-As_3 + b)$ for $n - r$ coordinates



Problem: LSH is not additively homomorphic!

OUR IDEA



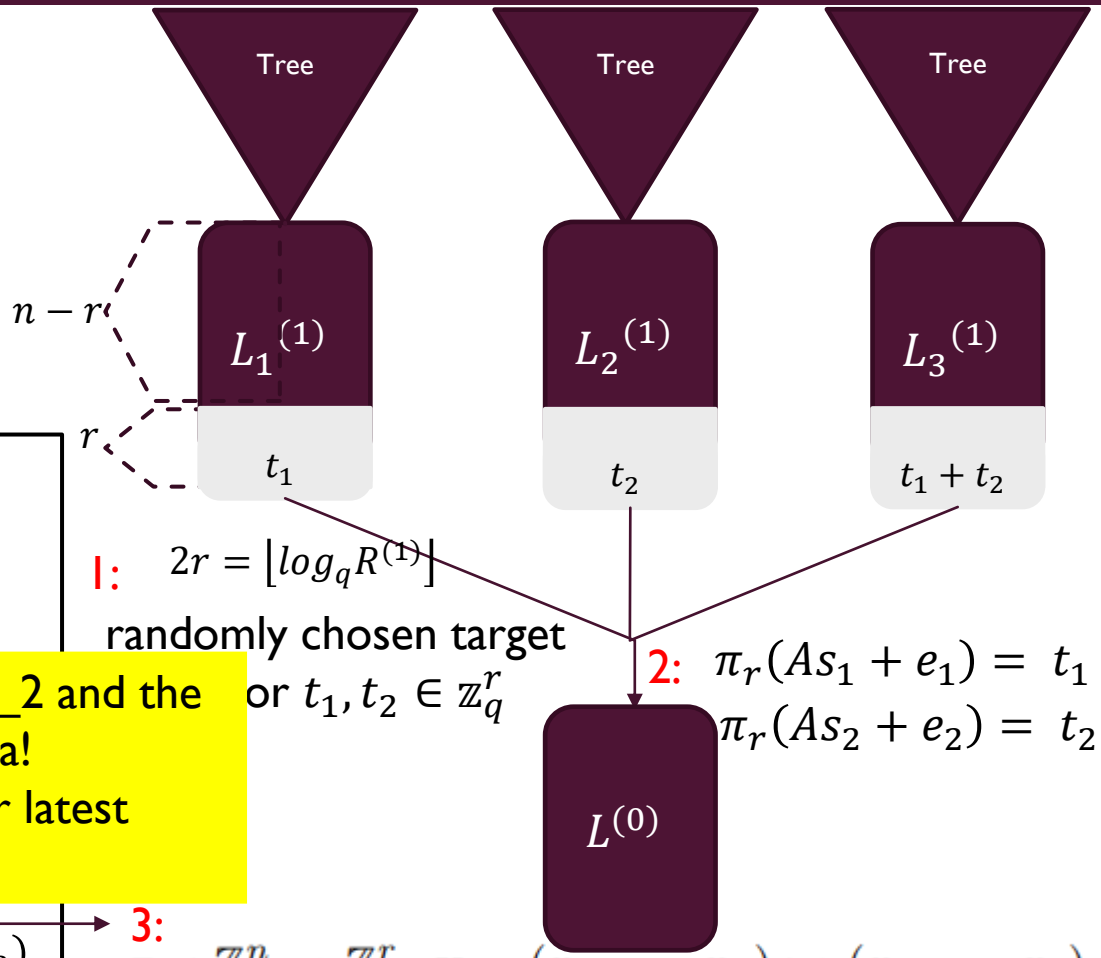
$$e = e_3 - e_1 - e_2$$

- Equation: $A_1 \cdot s_1 + A_2 \cdot s_2 = -A_3 \cdot s_3 + b + e \bmod q$
  i.e. $A_1 \cdot s_1 + A_2 \cdot s_2 \approx -A_3 \cdot s_3 + b \bmod q$

[Extended Meet-LWE Attack (high-level idea)]

- Input : $A \in Z_q^{n \times n}, b \in Z_q^n$
- Choose representation REP-0, REP-1-0, REP-1-1
- Guess r coordinates of e_1, e_2 (say t_1, t_2)
- For s_1, construct L_1 with entries (s_1, As_1)
- For s_2, construct L_2 with entries (s_2, As_2)
- For s_3, construct L_3 with entries $(s_3, h(b - As_3))$
- Output $s_1 + s_2 + s_3$ s.t.
  - $\pi_r(As_1 + e_1) = t_1, \pi_r(As_2 + e_2) = t_2$
  - $\pi_r(-As_3 + b + e_3) = t_1 + t_2 = \pi_r(As_1 + e_1) + \pi_r(As_2 + e_2)$
  - $h(As_1 + As_2) = h(-As_3 + b)$ for $n - r$ coordinates

Changed L_1, L_2 and the matching criteria! (reflected in our latest eprint version)

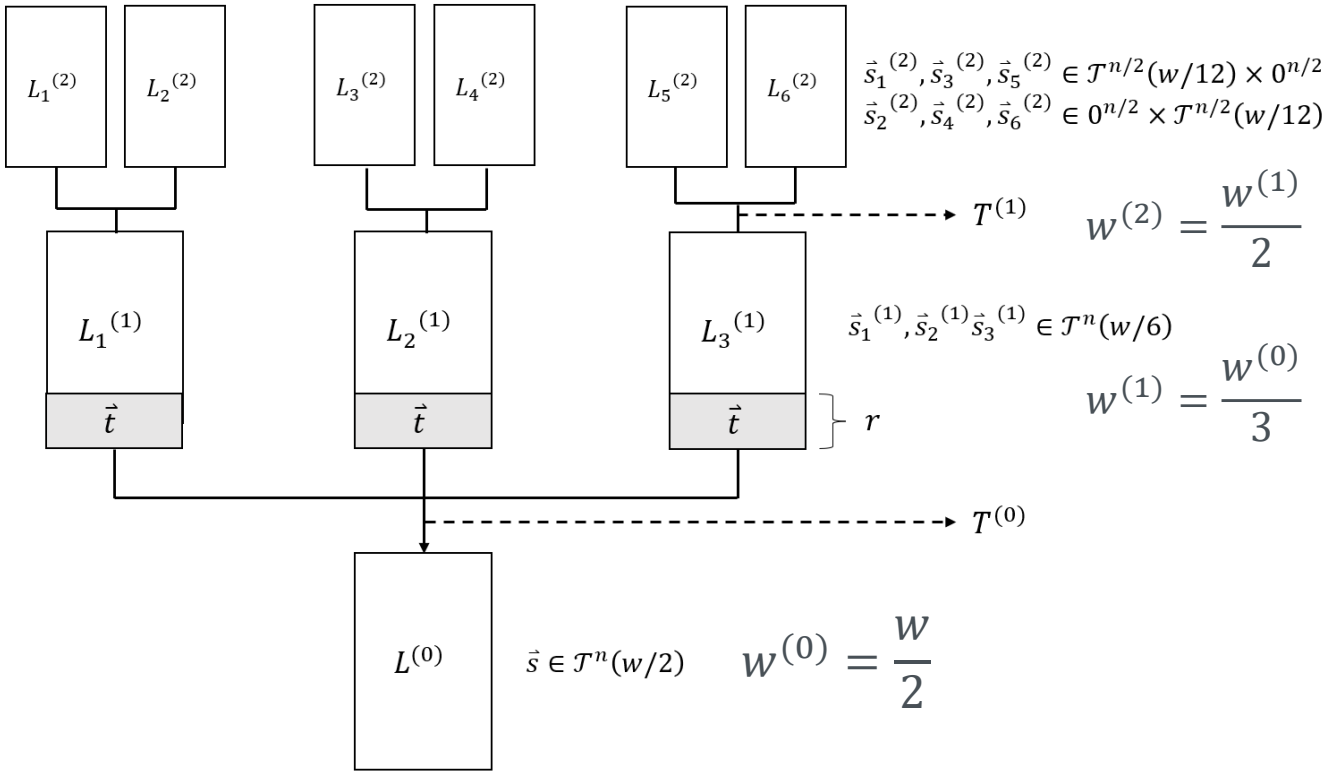


OUR IDEA

- By using ternary representations $s = s_1 + s_2 + s_3$, **the number of solutions ($:= R$) even more increases**
- We can also reduce the list (L_1, L_2, L_3) sizes with a factor of this increased R (by guessing r coordinates of e), expecting at least one solution exists
- $r \approx \frac{1}{2} \cdot \log_q R$ rather than $r \approx \log_q R$ in the original Meet-LWE ; note that r determines guessing complexity

→ Putting all these together, the extended Meet-LWE via ternary tree shows lower complexity than the that of the Meet-LWE attack.

REP-0



	I	0	-I
REP-0	• I+0+0 • 0+I+0 • 0+0+I	-	• (-I)+0+0 • 0+(-I)+0 • 0+0+(-I)

REP-0 : RESULTS

Table 5: Comparison on Non-asymptotic Complexity, May's [59] vs. Ours for REP-0. Each row represents the attack complexities for each parameter, which is $\log T = \log T_{\mathsf{list}} + \log T_{\mathsf{guess}}$.

(n, q, w)	May [bit]	Ours [bit]
NTRU-Encrypt		
$(509, 2048, 254)$	$305 = 287{+}18$	$\mathbf{231} = 213{+}18$
$(677, 2048, 254)$	$364 = 347{+}18$	$\mathbf{268} = 250{+}18$
$(821, 4096, 510)$	$520 = 487{+}33$	$\mathbf{414} = 379{+}34$
NTRU Prime		
$(653, 4621, 288)$	$370 = 352{+}18$	$\mathbf{279} = 260{+}19$
$(761, 4591, 286)$	$408 = 390{+}18$	$\mathbf{299} = 380{+}19$
$(857, 5167, 322)$	$459 = 439{+}20$	$\mathbf{337} = 316{+}21$
BLISS I+II		
$(512, 12289, 154)$	$247 = 238{+}9$	$\mathbf{175} = 167{+}9$
GLP I		
$(512, 8383489, 342)$	$325 = 314{+}12$	$\mathbf{257} = 246{+}12$

REP-0 : RESULTS

Table 8: Security Evaluation of KpqC Round 2 Schemes (NTRU+, SMAUG-T) with Our Attack Instantiated with REP-0. Each row represents the attack complexities for each parameter, which is $\log T = \log T_{\mathsf{list}} + \log T_{\mathsf{guess}}$.

Parameters	(n, q, w)	May [bit]	Ours [bit]
NTRU+576	$(576, 3457, 288)$	$341 = 323+18$	$\mathbf{261} = 242+20$
NTRU+768	$(768, 3457, 384)$	$455 = 430+25$	$\mathbf{349} = 322+26$
NTRU+864	$(864, 3457, 432)$	$513 = 484+29$	$\mathbf{392} = 363+30$
NTRU+1152	$(1152, 3457, 576)$	$684 = 646+38$	$\mathbf{524} = 484+40$
TiMER	$(512, 1024, 100)$	$192 = 185+7$	$\mathbf{135} = 128+7$
SMAUG-T128	$(512, 1024, 132)$	$227 = 217+10$	$\mathbf{165} = 155+10$
SMAUG-T192	$(768, 2048, 151)$	$289 = 279+10$	$\mathbf{208} = 197+11$
SMAUG-T256	$(1280, 2048, 160)$	$361 = 351+10$	$\mathbf{253} = 241+11$

[May21]: REP-0 vs. [Ours]: REP-0

REP-1 [MAY21] → REP-1-0, 1-1 [OURS]

	1	0	-1
REP-1	• 1+0 • 0+1	• 1+(-1) • (-1)+1	• (-1)+0 • 0+(-1)

	1	0	-1
REP-1-0	• 1+0+0 • 0+1+0 • 0+0+1	• 1+(-1)+0 • (-1)+1+0 • 0+1+(-1) • 0+(-1)+1 • 1+0+(-1) • (-1)+0+1	• (-1)+0+0 • 0+(-1)+0 • 0+0+(-1)

	1	0	-1
REP-1-1	• 1+0+0 • 0+1+0 • 0+0+1 • 1+(-1)+1 • (-1)+1+1 • 1+1+(-1)	• 1+(-1)+0 • (-1)+1+0 • 0+1+(-1) • 0+(-1)+1 • 1+0+(-1) • (-1)+0+1	• (-1)+0+0 • 0+(-1)+0 • 0+0+(-1) • (-1)+(-1)+1 • (-1)+1+(-1) • 1+(-1)+(-1)

REP-1 : RESULTS

Table 6: Non-asymptotic Attack Complexity of Our Attack Instantiated with REP-1-0 and REP-1-1. Each row represents the attaack complexities for each parameter, which is $\log T = \log T_{\mathsf{list}} + \log T_{\mathsf{guess}}$.

(n, q, w)	REP-1-0 [bit]	params	REP-1-1 [bit]	params
NTRU-Encrypt				
$(509, 2048, 254)$	**192** = 173+19	3: 13,1	202 = 167+35	3: 8,0,0,2,0,0
$(677, 2048, 254)$	**214** = 190+24	3: 19,1	233 = 196+37	3: 9,0,0,1,0,0
$(821, 4096, 510)$	**346** = 318+28	3: 14,1	351 = 280+71	3: 19,3,0,10,0,0
NTRU Prime				
$(653, 4621, 288)$	**232** = 213+19	3: 13,1	240 = 201+39	3: 11,1,0,2,0,0
$(761, 4591, 286)$	**242** = 218+24	3: 21,1	256 = 206+50	3: 19,1,0,2,0,0
$(857, 5167, 322)$	**273** = 247+26	3: 22,1	291 = 250+41	3: 11,0,0,1,0,0
BLISS I+II				
$(512, 12289, 154)$	**151** = 136+15	3: 17,1	157 = 137+20	3: 6,0,0,2,0,0
GLP I				
$(512, 8383489, 342)$	230 = 220+10	3: 13,4	**217** = 194+23	3: 9,3,0,10,0,0

[Ours] REP-1-0 vs. REP-1-1

REP-1 : RESULTS

Table 7: Comparison on Non-asymptotic Complexity, May's [59] REP-2 vs. Ours. Each row represents the attack complexities for each parameter, which is $\log T = \log T_{\mathsf{list}} + \log T_{\mathsf{guess}}$.

(n, q, w)	May [bit]	params	Ours [bit]	params
NTRU-Encrypt				
$(509, 2048, 254)$	$227 = 189{+}38$	4: 26,2,17,3	$\mathbf{192} = 173{+}19$	3: 13,1
$(677, 2048, 254)$	$273 = 231{+}42$	4: 32,1,15,1	$\mathbf{214} = 190{+}24$	3: 19,1
$(821, 4096, 510)$	$378 = 318{+}60$	4: 34,5,30,6	$\mathbf{346} = 318{+}28$	3: 14,1
NTRU Prime				
$(653, 4621, 288)$	$272 = 229{+}42$	4: 36,2,22,5	$\mathbf{232} = 213{+}19$	3: 13,1
$(761, 4591, 286)$	$301 = 258{+}43$	4: 36,1,17,2	$\mathbf{242} = 218{+}24$	3: 21,1
$(857, 5167, 322)$	$338 = 291{+}47$	4: 37,2,19,2	$\mathbf{273} = 247{+}26$	3: 22,1
BLISS I+II				
$(512, 12289, 154)$	$187 = 163{+}24$	4: 27,0,11,1	$\mathbf{151} = 136{+}15$	3: 17,1
GLP I				
$(512, 8383489, 342)$	$225 = 206{+}20$	4: 22,3,19,4	$\mathbf{217} = 194{+}23$	3: 9,3,0,10,0,0

[May21]: REP-2 vs. [Ours]: REP-1

REP-1 : RESULTS

Table 10: Security Evaluation of KpqC Round 2 schemes (NTRU+, SMAUG-T) with May's [59] REP-2 vs. Ours. Each row represents the attack complexities for each parameter, which is $\log T = \log T_{\mathsf{list}} + \log T_{\mathsf{guess}}$.

Parameters	May [bit]	params	Ours [bit]	params
NTRU+576	263 = 228+36	4: 21,2,9,0	**221** = 200+21	3: 16,1
NTRU+768	349 = 302+47	4: 24,3,14,1	**287** = 261+26	3: 18,1
NTRU+864	392 = 339+53	4: 29,3,14,3	**319** = 288+31	3: 22,1
NTRU+1152	519 = 448+71	4: 35,5,19,3	**433** = 397+36	3: 20,1
TiMER	144 = 124+21	4: 14,0,4,0	**122** = 98+24	3: 8,0,0,0,0,0
SMAUG-T128	167 = 147+20	4: 10,0,2,0	**147** = 132+16	3: 11,2
SMAUG-T192	214 = 192+21	4: 12,0,1,0	**182** = 161+21	3: 17,1
SMAUG-T256	283 = 255+29	4: 15,1,3,0	**231** = 210+21	3: 15,1

[May]: REP-2 vs. [Ours]: REP-1

- The claimed security using the 'beyond core-SVP' from SMAUG-T document were 135.3 bits, 144.7 bits, 202.0 bits, and 274.6 bits, respectively.
- For TiMER, SMAUGT192, and SMAUG-T256 parameters, the estimated attack complexities are lower in security by **13.3** bits, **20** bits, and **43.6** bits than claimed.

MINOR COMMENTS FOR THE SECURITY PROOF OF NTRU+ (ROUND 2 DOCUMENT VERSION)

SUMMARY ON NTRU+ KEM

- Security based on the **NTRU**, **RLWE** assumptions
  - RLWE here uses (random) binary secrets and ternary errors
- Uses **NTT-friendly rings**
  - $R_q = Z_q[x]/(f(x))$, where $f(x) = x^n - x^{n/2} + 1$ and $n = 2^i 3^j$ [1,2]
- Uses a new encoding named **SOTP** (Semi-generalized One Time Pad)
- In the CCA-secure KEM, they **remove re-encryption** in decapsulation by adjusting Fujisaki-Okamoto transform

- [1] Vadim Lyubashevsky and Gregor Seiler. NTTRU: Truly fast NTRU using NTT. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019 (https://tches. iacr.org/index.php/TCHES/article/view/8293).
- [2] Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, Gregor Seiler, and Dominique Unruh. A thorough treatment of highly-efficient NTRU instantiations. Public-Key Cryptography – PKC 2023 (https://eprint.iacr.org/2021/1352).

Parameters	Security level	n	q	Sizes (Bytes)			Cycles(ref)			Cycles(AVX2)		
				pk	ct	sk	Keygen	Encaps	Decaps	Keygen	Encaps	Decaps
NTRU+576	I	576	3,457	864	864	1,728	321,405	110,754	163,277	17,440	14,307	12,445
NTRU+768	I	768	3,457	1,152	1,152	2,304	313,669	145,658	227,028	16,032	17,514	15,848
NTRU+864	3	864	3,457	1,296	1,296	2,592	339,912	169,634	262,017	14,068	19,293	17,671
NTRU+1152	5	1,152	3,457	1,728	1,728	3,456	905,131	230,448	348,076	42,993	25,592	24,063

COMMENTS FOR THE SECURITY PROOF

[FO transform without Re-Encryption (Lemma 4.3)]

- To show: For input ciphertext c,

  $c = Enc'(pk, m'; R')$ if and only if $r' = r''$

- ($\rightarrow$) Assume $c = Enc'(pk, m'; R')$ in Decaps.

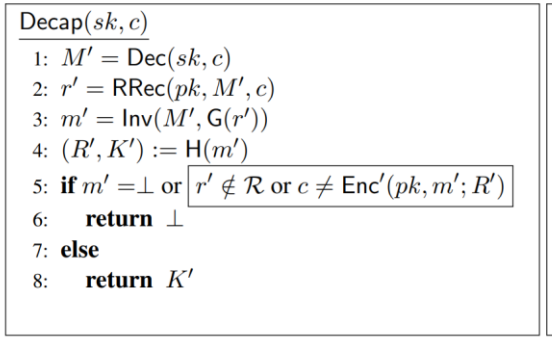


Figure 11: Modified KEM $= \mathsf{FO}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

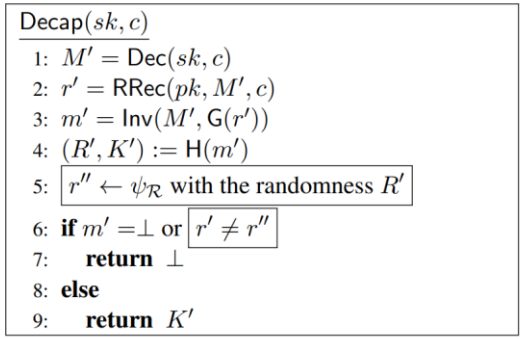


Figure 12: KEM $= \overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

By the definition of Enc', $c = Enc(pk, SOTP(m', G(r'')); r'')$ holds where $r'' \leftarrow \psi_R$ is sampled using R'

Also, since $M' = Dec(sk, c) \in \mathcal{M}$ and $r' = RRec(pk, M', c) \in \mathcal{R}$, the rigidity of the PKE leads to

$c = Enc(pk, M'; r')$.

⋮

COMMENTS FOR THE SECURITY PROOF

[FO transform without Re-Encryption (Lemma 4.3)]

- To show: For input ciphertext c,

  $c = Enc'(pk, m'; R')$ if and only if $r' = r''$

- ($\rightarrow$) Assume $c = Enc'(pk, m'; R')$ in Decaps.

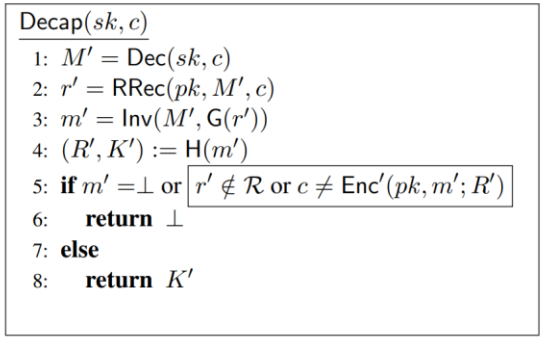


Figure 11: Modified KEM $= \mathsf{FO}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

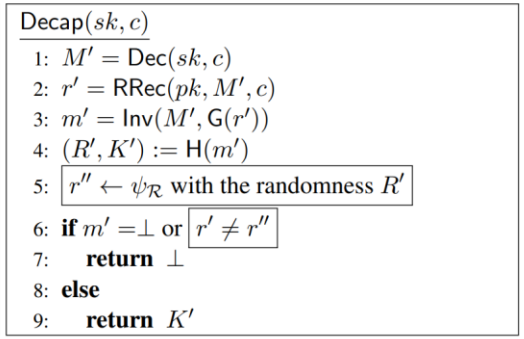


Figure 12: KEM $= \overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

By the definition of Enc', $c = Enc(pk, SOTP(m', G(r'')); r'')$ holds where $r'' \leftarrow \psi_R$ is sampled using R'

Also, since $M' = Dec(sk, c) \in \mathcal{M}$ and $r' = RRec(pk, M', c) \in \mathcal{R}$, the rigidity of the PKE leads to

$c = Enc(pk, M'; r')$.

⋮

Rigidity. Under the assumption that PKE is RR, we say that PKE is δ-rigid if for all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c \in \mathcal{C}$, we have

$$\Pr\left[\mathsf{Enc}(pk, m'; r') \neq c | m' = \mathsf{Dec}(sk, c) \in \mathcal{M} \wedge r' = \mathsf{RRec}(pk, m', c) \in \mathcal{R}\right] \leq \delta.$$

COMMENTS FOR THE SECURITY PROOF

[FO transform without Re-Encryption (Lemma 4.3)]

- To show: For input ciphertext c,

  $c = Enc'(pk, m'; R')$ if and only if $r' = r''$

- ($\rightarrow$) Assume $c = Enc'(pk, m'; R')$ in Decaps.

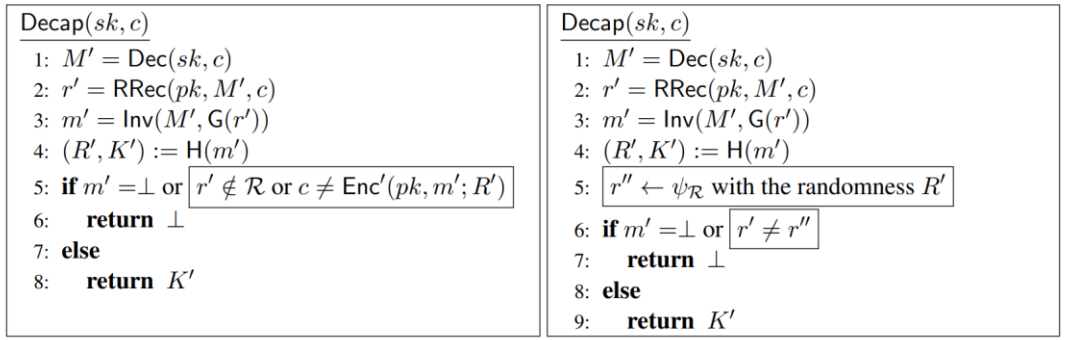


Figure 11: Modified KEM $= \mathsf{FO}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

Figure 12: KEM $= \overline{\mathsf{FO}}^{\perp}_{\mathsf{KEM}}[\mathsf{PKE}', \mathsf{H}]$

By the definition of Enc', $c = Enc(pk, SOTP(m', G(r'')); r'')$ holds where $r'' \leftarrow \psi_R$ is sampled using R'

Also, since $M' = Dec(sk, c) \in \mathcal{M}$ and $r' = RRec(pk, M', c) \in \mathcal{R}$, the rigidity of the PKE leads to

$c = Enc(pk, M'; r')$.

Rigidity. Under the assumption that PKE is RR, we say that PKE is δ-rigid if for all $(pk, sk) \leftarrow \mathsf{Gen}(1^{\lambda})$ and $c \in \mathcal{C}$, we have

$$\Pr\left[\mathsf{Enc}(pk, m'; r') \neq c | m' = \mathsf{Dec}(sk, c) \in \mathcal{M} \wedge r' = \mathsf{RRec}(pk, m', c) \in \mathcal{R}\right] \leq \delta.$$

They assumed that, for c (input of Decaps), $\boldsymbol{M' = Dec(sk, c) \in \mathcal{M}}$ **and** $\boldsymbol{r' = RRec(pk, M', c) \in \mathcal{R}}$.
But, there is no guarantee for that.

SECURITY EVALUATION OF {LWE, LWR}-BASED SCHEMES USING LATTICE ESTIMATOR

- Lattice Estimator — Lattice Estimator 0.1 documentation (lattice-estimator.readthedocs.io)
- Albrecht, Martin R., Rachel Player, and Sam Scott. "On the concrete hardness of learning with errors." *Journal of Mathematical Cryptology* 9.3 (2015): 169-203.

GOAL

- Better understanding for the security estimation of KpqC Round 2 candidates
  - Analysis reports for the respective attacks

- Estimate the security for all the LWE/LWR based schemes {NTRU+, SMAUG-T, HAETAE, NCC-Sign}

METHODS

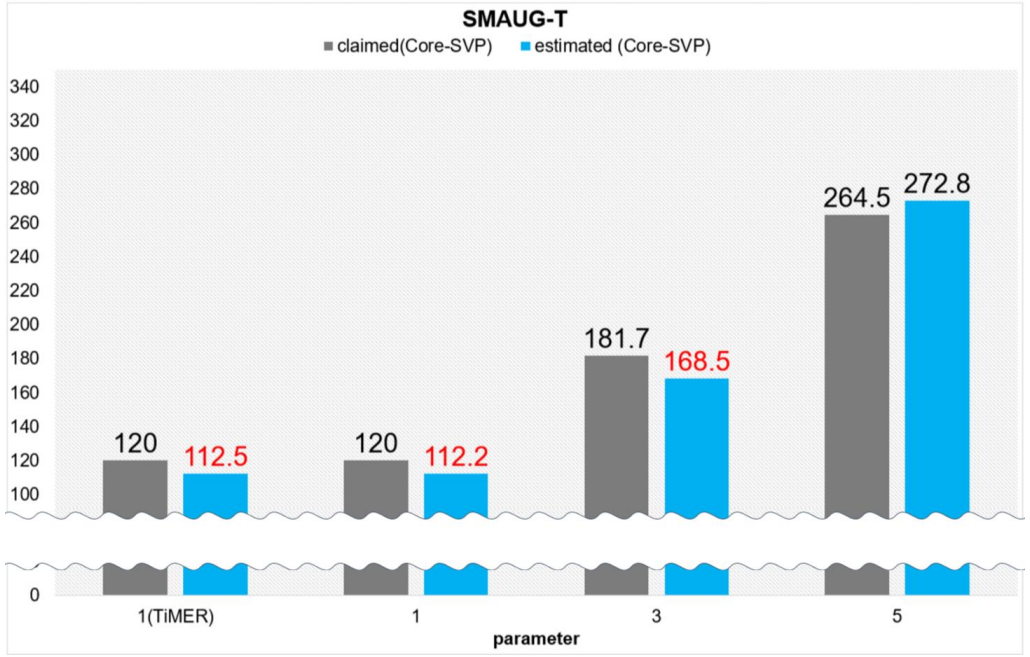
- **Lattice estimator**
  - For LWE/LWR security analysis, M. Albrecht's Lattice Estimator (Lattice Estimator — Lattice Estimator 0.1 documentation (lattice-estimator.readthedocs.io)) is used. Lattice Estimator is a Sage open source that calculates the attack complexities and additional parameters required for attack by taking LWE/LWR parameters as input values.

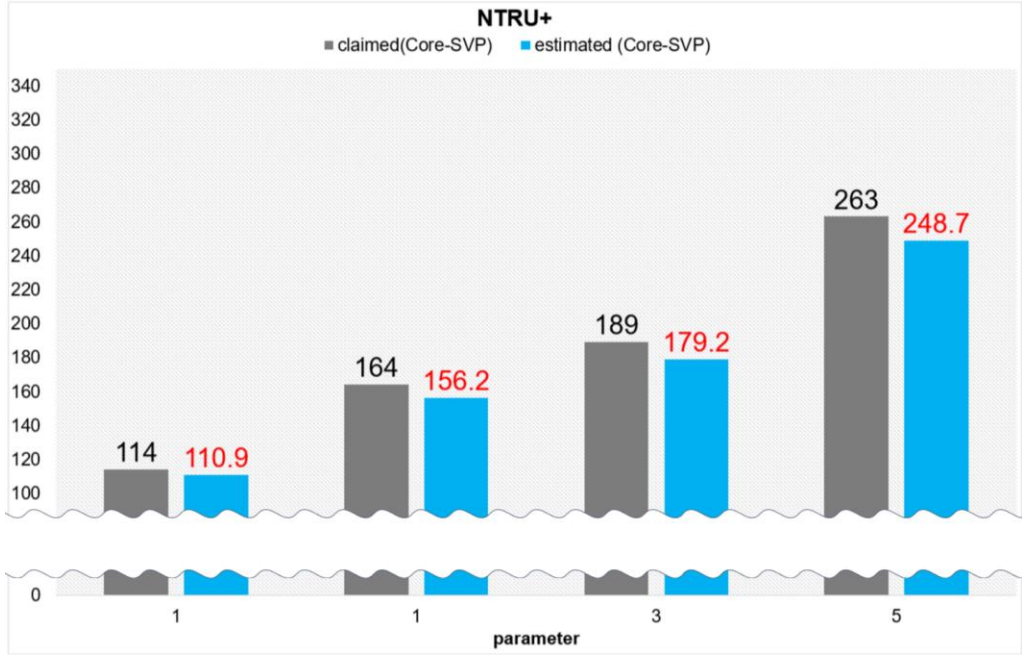
- **The BKZ Algorithm Complexity – Core-SVP model**
  - The principle of the BKZ algorithm is to repeatedly apply the SVP solver, an algorithm that finds the shortest vector, for a sub-lattice of dimension (β) smaller than that of a given lattice.
  - The Core-SVP model from the NewHope paper (USENIX'16) is a model for estimating the time complexity of the BKZ algorithm. The classical security in bits is estimated as $2^{c \cdot \beta}$ using $c = 0.292$, and the quantum security (bit) can be also estimated by calculating the classical security (bit) $\times\, c_q/0.292$ in the Core-SVP model.

	Classical	Quantum[1]
c	0.292	0.257
T	$2^{0.292\beta}$	$2^{0.257\beta}$

[1] Chailloux, A., Loyer, J. Lattice Sieving via Quantum Random Walks. ASIACRYPT 2021

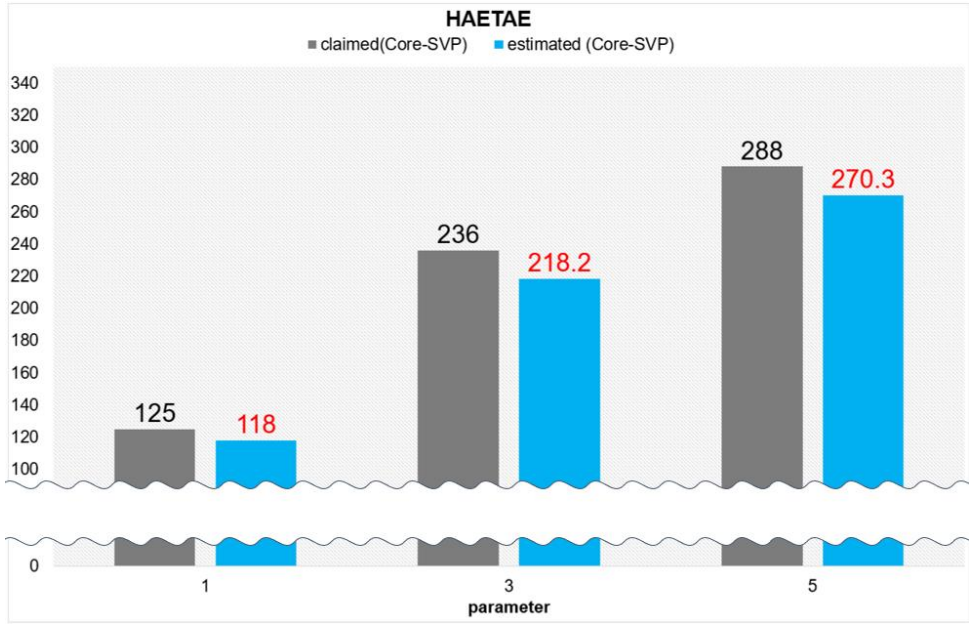
RESULTS - KEMS





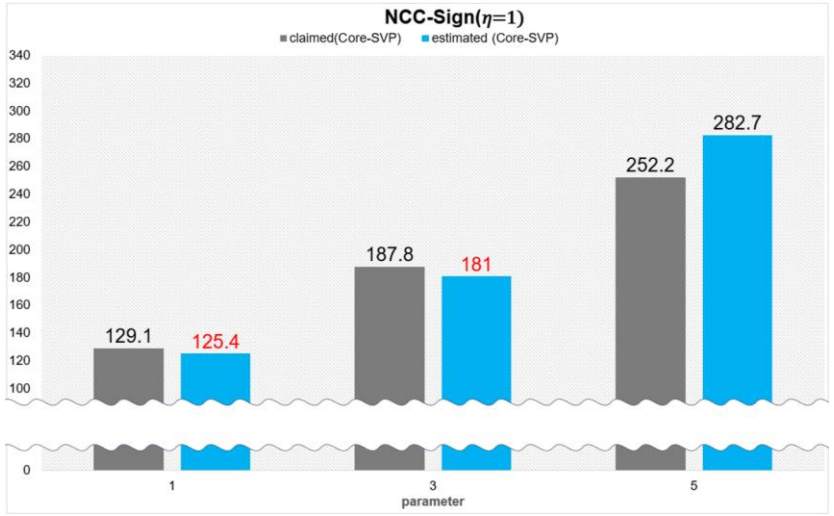
- Estimated security for SMAUG-T ; 7.5~13.2 bits lower than claimed for security category 1 and 3
- Estimated security for NTRU+ ; 3.1~14.3 bits lower than claimed

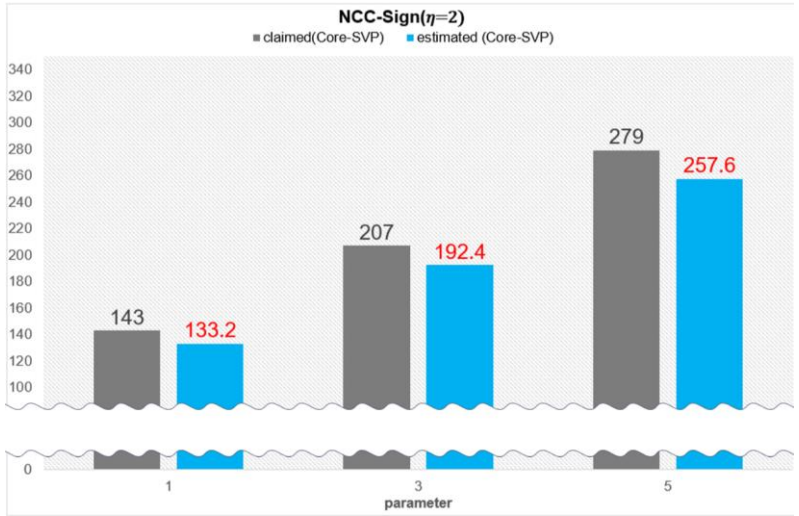
RESULTS - SIGNATURES

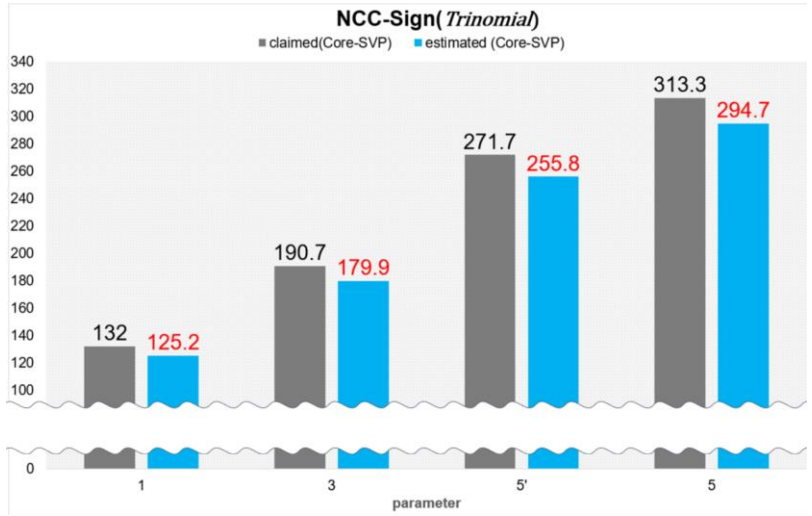


- Estimated security for HAETAE ; 7~17.8 bits lower than claimed

RESULTS - SIGNATURES







- Estimated security for NCC-Sign ; 3.7~21.4 bits lower than claimed

SUMMARY

- We generalize the Meet LWE attack via ternary trees which implies
  - Increased diversity of representations and
  - Decreased guessing complexity

- By analyzing and optimizing the attack complexities, we show that our approach gives better time complexity in the regime of practical parameters compared to May.

- For SMAUG-T, by exploiting the sparsity of the LWE secrets, we achieve the reduced attack complexity estimation for parameters {TiMER, SMAUGT192, SMAUGT256}.
  - the estimated attack complexities are lower in security by **13.3** bits, **20** bits, and **43.6** bits than claimed, respectively.

- For NTRU+, we point out some (minor) bugs in the security proof.

- Additionally, we report the discrepancies between estimated and claimed security bits, when estimating the security of each scheme using the Albrecht's lattice estimator.

THANK YOU!

ANY QUESTIONS OR COMMENTS?

JOOHEELEE@SUNGSHIN.AC.KR