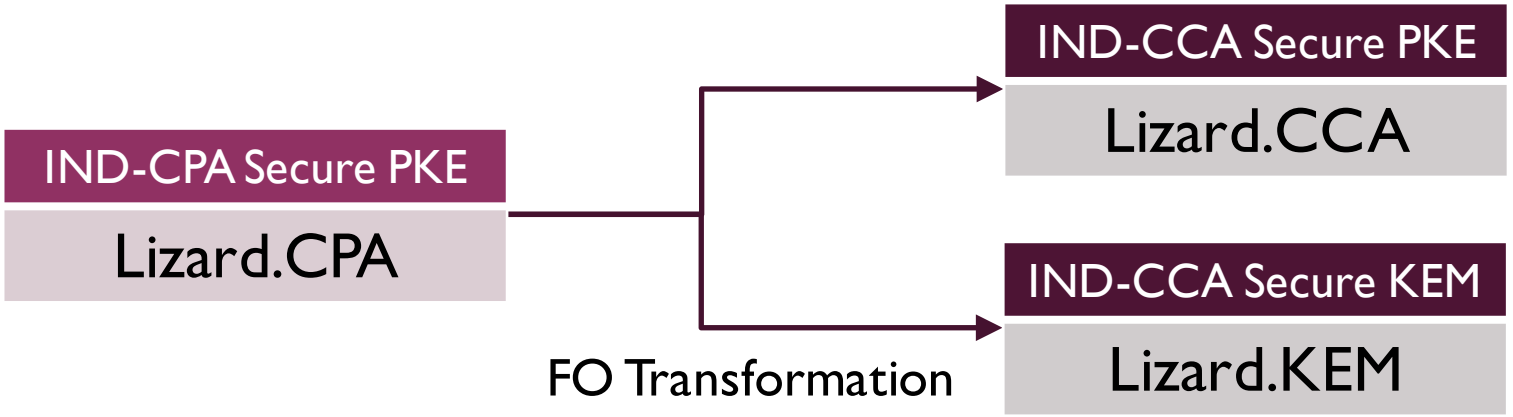
LIZARD:
PUBLIC KEY ENCRYPTION AND KEY ENCAPSULATION MECHANISM

JUNG HEE CHEON, SANGJOON PARK, **JOOHEE LEE**,
DUHYEONG KIM, YONGSOO SONG, SEUNGWAN HONG,
DONGWOO KIM, JINSU KIM, SEOUNG-MIN HONG,
AARAM YUN, JEONGSU KIM, HAERYONG PARK,
EUNYOUNG CHOI, KIMOON KIM, JUN-SUB KIM, JIEUN LEE

CONTENTS

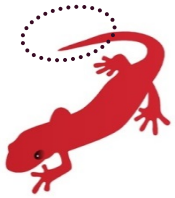
- Overview
- Main Strengths
- Security
- Parameter Selection
- Summary on Performance
- Flexibility
- Further Improvements (in progress)

OVERVIEW



- Ring versions are also provided
- Parameter Suggestions for Category 1/3/5, resp.

MAIN STRENGTHS



1. [Simpler and Faster] algorithms;
   - Use **LWR** in the Encryption/Encapsulation phases
   - Use sparse signed binary or signed binary secrets
2. [Ciphertext Compression] via LWR-style rounding (vs. LWE-style)
3. [Provable IND-CPA, IND-CCA2 Security] from (R)LWE & (R)LWR with small secrets
4. [Parameters Resist All Known Attacks] unless a significant breakthrough
   - Cryptographically negligible Dec. Fail. Rates
   - Based on the Core SVP hardness (Methodology of NewHope)

MAIN STRENGTHS -- SIMPLER

1. Enc of typical LWE based PKE requires two random components:
   - Ephemeral secret vector (or matrix)
   - Error vector (or matrix)
2. Using LWR rounding in Enc/Encaps rules out generating error vectors
3. Further use **sparse** signed binary or signed binary secrets

Encryption Procedure	Algorithm
1. **Generation**: random sparse binary vector	$\vec{r} \leftarrow \{-1, 0, 1\}^m$
2. **Subset-sum**: row vectors of PK **(simple & fast)**	$(a, b) \leftarrow (A^t\vec{r}, B^t\vec{r})$
3. **Addition**: an encoded msg vector **(simple & fast)**	$(a, b) \leftarrow (a, b + 2^k m)$
4. **Rounding**: via addition & bit shifting **(simple & fast)**	$(a, b) \leftarrow (\lfloor \frac{a+2^{\ell-1}}{2^\ell} \rfloor, \lfloor \frac{b+2^{\ell-1}}{2^\ell} \rfloor)$

$A \in Z_q^{m \times n}$, $B \in Z_q^{m \times \ell}$, (A, B): PK, $k = \log q - 1$, $\ell = \log q - \log p$

MAIN STRENGTHS -- FAST

- Our C implementation for Lizard.CCA shows that Enc takes only 0.031 ms fo r Category 1 and 32-byte msgs (0.036 ms for RLizard.CCA)

Scheme	Parameter	KeyGen (# kcycles)	Enc (# kcycles)	Dec (# kcycles)
Lizard.CCA	CCA_CATEGORY1_N536	406 432	81	88
	CCA_CATEGORY1_N663	459 082	83	94
RLizard.CCA	RING_CATEGORY1	1 167	94	101

* Performance measured on Linux with CPU Intel Xeon E5-2640 v3 at 2.60GHz

SECURITY

- Lizard.CPA is ***IND-CPA secure*** under the hardness assumption of LWE and LWR problems with small secrets, both of which have reductions from the standard LWE

- Lizard.KEM and Lizard.CCA are obtained by applying a variant of **Fujisaki-Okamoto transform** [HHK'17] for Lizard.CPA, so they are ***IND-CCA2*** *secure* in the quantum random oracle model (QROM)

- Same arguments can be applied to ring versions (RLizard.CPA, RLizard.KEM and RLizard.CCA)

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A modular analysis of the Fujisaki-Okamoto transformation." *Theory of Cryptography Conference* 2017.

PARAMETER SELECTION

- Mainly considered: **Dual attack** [Alb17] and **Primal attack** [AGVW18]
- Assume the attacks are using BKZ algorithm with Sieve (equipped with Grover's quantum search algorithm); Security measured based on the **Core SVP hardness** as in [NewHope]
- Dec. Fail. Rates are set to be cryptographically negligible

- [Alb17] Albrecht, Martin R. "On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL." *Eurocrypt* 2017.
- [AGVW18] Albrecht, M. R., Göpfert, F., Virdia, F., and Wunderer, T. "Revisiting the expected cost of solving uSVP and applications to LWE." *Asiacrypt 2018.*
- [NewHope] Alkim, E., Ducas, L., Pöppelmann, T., & Schwabe, P. "Post-quantum key exchange-a new hope." *USENIX 2016.*

BEST ATTACK COMPLEXITIES

Parameter Sets	$\log_2(DFR)$	$\log_2 T_{LWE}$	$\log_2 T_{LWR}$
KEM_CATEGORY1_N663 CCA_CATEGORY1_N663	-153.500	**131**	147
KEM_CATEGORY3_N952 CCA_CATEGORY3_N952	-337.189	203	**195**
KEM_CATEGORY5_N1300 CCA_CATEGORY5_N1300	-332.810	**264**	291

DFR : Dec. Fail. Rate exactly calculated by python code

T_{LWE} : Time complexity of the best known attacks of LWE

T_{LWR} : Time complexity of the best known attacks of LWR

BEST ATTACK COMPLEXITIES

Parameter Sets	$\log_2$(DFR)	$\log_2 T_{LWE}$	$\log_2 T_{LWR}$
RING_CATEGORY1	-188.248	153	**147**
RING_CATEGORY3_N1024	-245.897	**195**	195
RING_CATEGORY5	-305.684	**318**	348

DFR : Dec. Fail. Rate exactly calculated by python code

T_{LWE} : Time complexity of the best known attacks of LWE

T_{LWR} : Time complexity of the best known attacks of LWR

SUMMARY ON PERFORMANCE

- Sizes
  - Lizard.CCA
    - Sizes for 256-bit msgs (Category 1):

	pk	sk	ctxt
Sizes	1.8 MB	170 KB	0.98 KB
Compressed upto	0.3 MB	-	-

  - RLizard.CCA
    - Sizes for 1024-bit msgs (Category 1):

	pk	sk	ctxt
Sizes	4.1 KB	0.3 KB	2.2 KB
Compressed upto	1.3 KB	-	-

- Speeds
  - Enc of Lizard is fast (from **81** kcycles for Category 1), and RLizard has balanced perfor mances

FLEXIBILITY OF THE LIZARD

- Lizard can be implemented flexibly for different purposes
- We implemented Lizard.CPA in 3 different ways for 3 different usages:
  - On ARM Core (Android, Galuxy S7); Enc: 0.077 ms, Dec: 0.023 ms
  - For 32-bit msgs; Enc: 0.009 ms, Dec: 0.001 ms
  - AHE; Enc: 0.014 ms, Dec: 0.012 ms

FURTHER IMPROVEMENTS (IN PROGRESS)

- KeyGen can be done faster by generating a random seed for each random component and then using AES-CTR mode to expand it :

Scheme	Parameter	Submitted KeyGen (ms)	Improved KeyGen (ms)
Lizard.CCA	CCA_CATEGORY1_N536	71.993	3.182
	CCA_CATEGORY1_N663	87.848	3.863

- Use AVX2 Instruction :

Scheme	Parameter	Enc (# kcycle)	Dec (# kcycle)
Lizard.CCA	CCA_CATEGORY1_N536	52	62
	CCA_CATEGORY1_N663	52	66

Thank You !

EX 1. APPLICATION ON SMARTPHONE

- Implemented Lizard.CPA on Android application

- Used parameters (128-bit security):

m	n	$\log_2 q$	$\log_2 p$	α^{-1}	ρ	h_r
960	608	10	8	1822	1/2	128

- Performance:

KeyGen (ms)	Enc (ms)	Dec(ms)
288.618	0.0770	0.0229

EX 2. FOR 32-BIT MESSAGES

- Implemented Lizard.CPA with 32-bit message space
- Can be utilized on low-end devices
- Used parameters (119-bit security):

m	n	$\log_2 q$	$\log_2 p$	α^{-1}	ρ	h_r	$\log_2 \epsilon$
724	480	11	9	303	1/2	128	-154

- Performance:

	ctxt (bytes)	pk (bytes)	sk (bytes)	KeyGen (ms)	Enc (ms)	Dec (ms)
A as matrix (A as seed)	576	741,376 (46,368)	3,840	4.749 (1.891)	0.009 (0.052)	0.001

EX 3. ADDITIVE HOMOMORPHIC ENCRYPTION

- Post-quantum alternative for AHE
- Parameters (128-bit security):

m	n	$\log_2 q$	$\log_2 p$	α^{-1}	ρ	h_r
1024	816	16	14	21000	1/2	136

- Performance:

	ctxt (bytes)	pk (bytes)	sk (bytes)	KeyGen (ms)	Enc (ms)	Dec (ms)	Add (ms)
A as matrix (A as seed)	1,876	2,195,456 (524,320)	52,224	25.923 (21.444)	0.014 (0.092)	0.012	0.0005