LIZARD NIST SUBMISSION

2018. 06. 29

JUNG HEE CHEON

CONTENTS

- LWE Problem
- LWE-based Encryptions
- Lizard NIST Submission
- Comparison to Other LWR-based Scheme(s)
- Further Improvements (in progress)

Learning with Errors (LWE) Problem

SOLVING A LINEAR EQUATION SYSTEM

- Q.

$$\underbrace{\begin{pmatrix} 1 & 3 & 7 \\ 4 & 5 & 7 \\ 6 & 6 & 9 \\ 2 & 7 & 3 \\ 3 & 8 & 7 \\ 5 & 4 & 2 \\ 1 & 0 & 5 \\ 4 & 5 & 3 \end{pmatrix}}_{\in\, \mathbb{Z}_{10}^{8\times 3}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 9 \\ 2 \\ 9 \\ 6 \\ 8 \\ 2 \\ 7 \end{pmatrix} \pmod{10} \;\Rightarrow\; \text{Find } \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}!$$

; Easy!

(We can solve it by using Gaussian elimination)

LEARNING WITH ERRORS (LWE) PROBLEM

- Q.

$$\underbrace{\begin{pmatrix} 1 & 3 & 7 \\ 4 & 5 & 7 \\ 6 & 6 & 9 \\ 2 & 7 & 3 \\ 3 & 8 & 7 \\ 5 & 4 & 2 \\ 1 & 0 & 5 \\ 4 & 5 & 3 \end{pmatrix}}_{\in \mathbb{Z}_{10}^{8\times 3}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ 2 \\ 9 \\ 1 \\ 0 \\ 1 \\ 0 \\ 8 \end{pmatrix}}_{\text{Small Error (unknown)}} = \begin{pmatrix} 7 \\ 9 \\ 2 \\ 9 \\ 6 \\ 8 \\ 2 \\ 7 \end{pmatrix} \pmod{10} \Rightarrow \text{Find } \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} !$$

; Hard!

DECISION-LWE PROBLEM

- Q. Distinguish

1	3	7
4	5	7
6	6	9
2	7	3
3	8	7
5	4	2
1	0	5
4	5	3

,

7
1
1
0
6
0
2
5

(mod 10)

From a uniform random sample in $\mathbb{Z}_{10}^{8\times 4}$

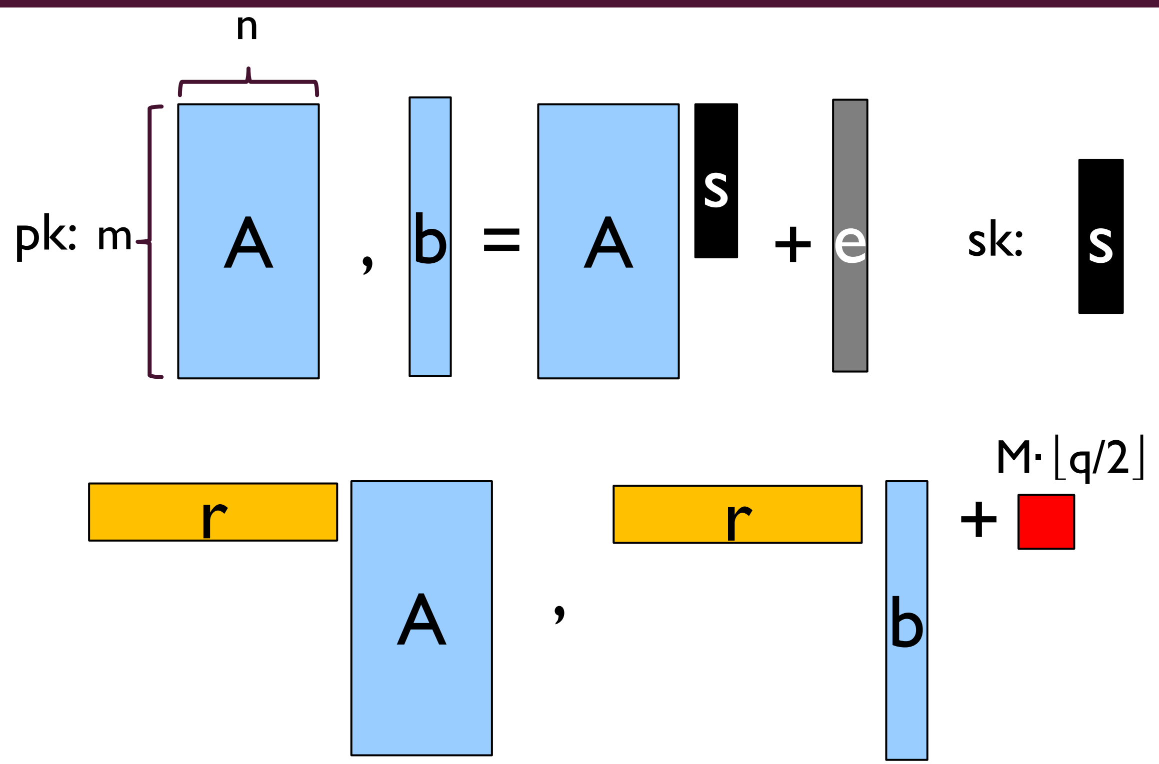
; Hard!

LWE-based Encryptions

LWE + LHL [REG05]

KeyGen

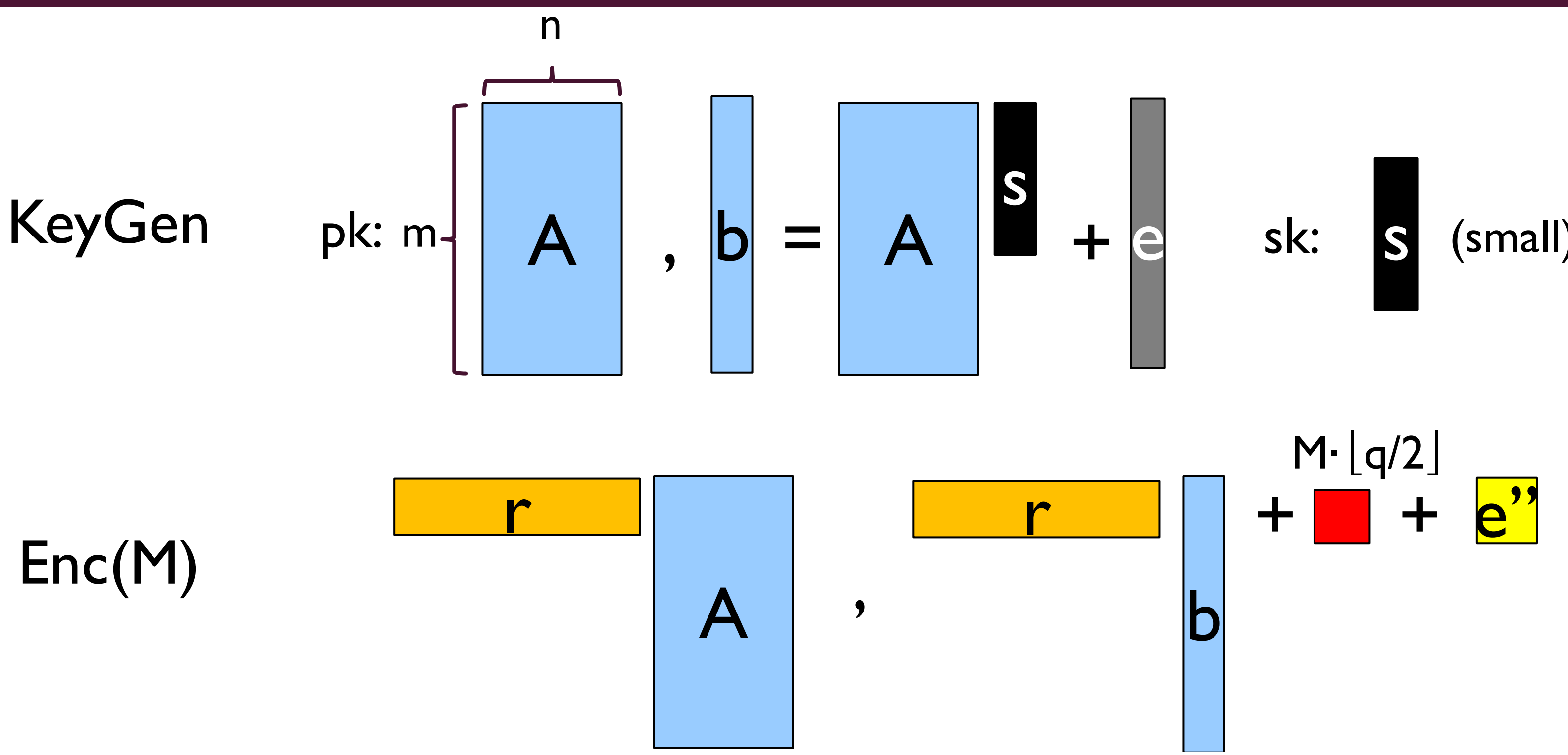
Enc(M)



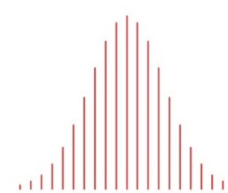
- **Require a large** **m** **to randomize LWE samples in Encryption**
  - ➢ Leftover Hash Lemma

- **Can We Reduce m?**

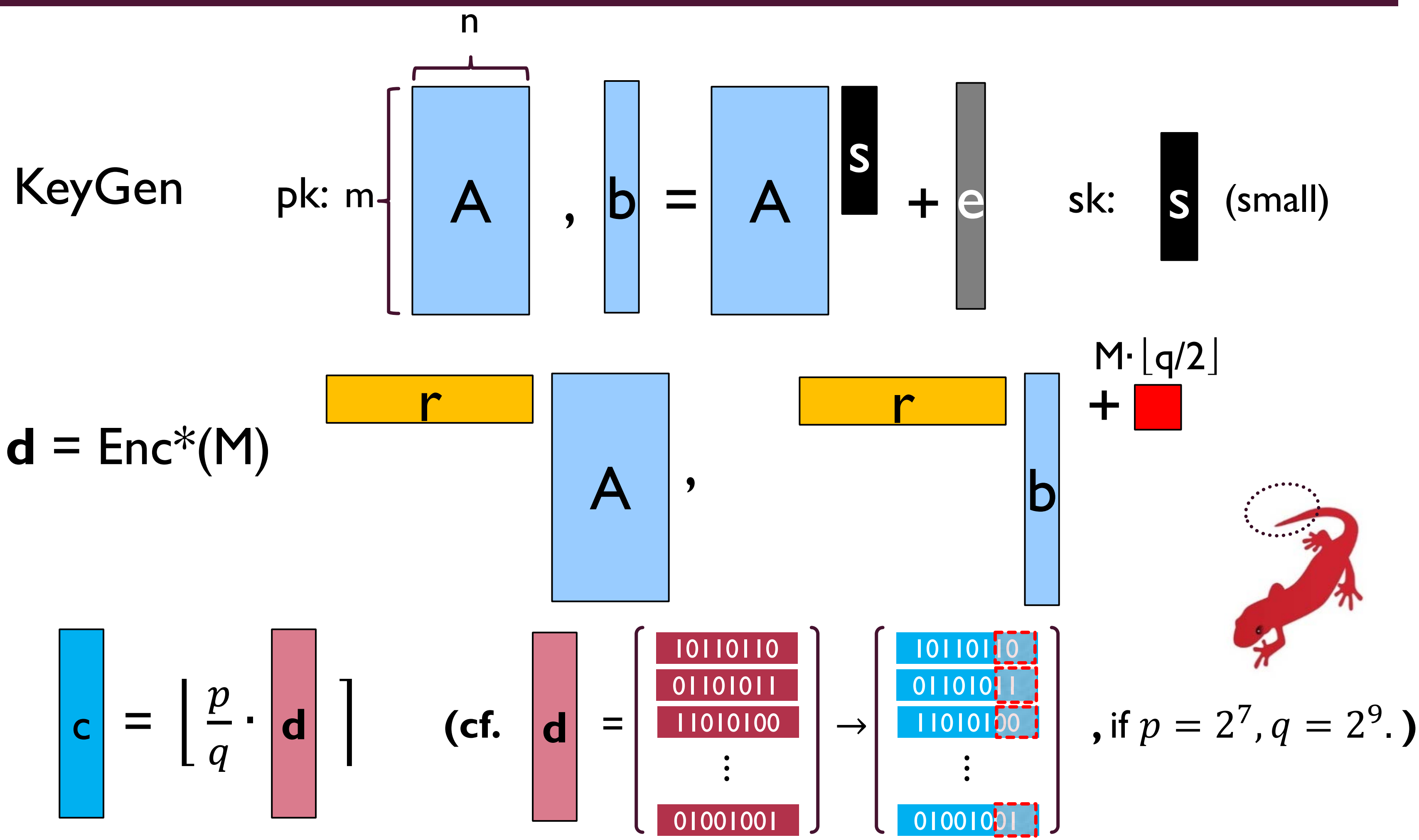
LWE + LWE [LP11]



- **Pros: smaller m by replacing LHL with LWE**
- **Cons:** *Discrete Gaussian samplings*



LWE + LWR [CKLS16]



LEARNING WITH ROUNDING (LWR) PROBLEM

- **Surprisingly,** it is secure under LWR assumption

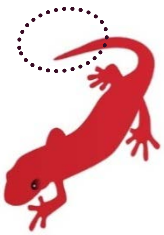
- LWR: Distinguish any m pairs of type

$$\left(\overbrace{a_i}^{n} ,\ b_i = \left\lfloor \frac{p}{q}\, a_i \, s \right\rceil \right) \in Z_q^n \times Z_p \text{ from uniform}$$

Discard the least significant bits of $<a_i,s>$

instead of adding small errors

- **Have reduction from LWE:** q is large or **m is small**



THE HARDNESS OF LWR PROBLEM

(q: LWR modulus, p: rounding modulus, n: LWR dimension.)

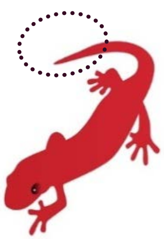
- Before 2016, security reduction <u>only when the modulus is somewhat large.</u>

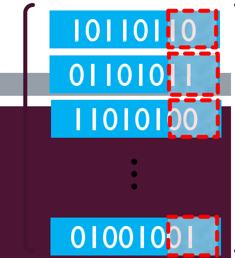
  ➢ Banergee, Peikert, Rosen [BPR12] introduced LWR, and showed *LWR ≥ LWE when q is sufficiently large.* ($q \geq p \cdot B \cdot n^{\omega(1)}$, B: LWE noise support bound)

  ➢ Alwen et al. [AKPW13] showed *LWR ≥ LWE when the modulus and modulus-to-error ratio are super-poly.*

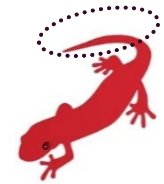
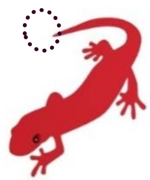
- Bogdanov et al. [BGM+16] in TCC 2016 showed LWR ≥ LWE when the number of samples is no larger than $O(q/Bp)$. (B: LWE noise support bound)

- Cryptanalytic hardness against best known lattice attacks: LWR = LWE when the variance of LWE noise is $12q^2/p^2$. (size of noise vectors are the same)



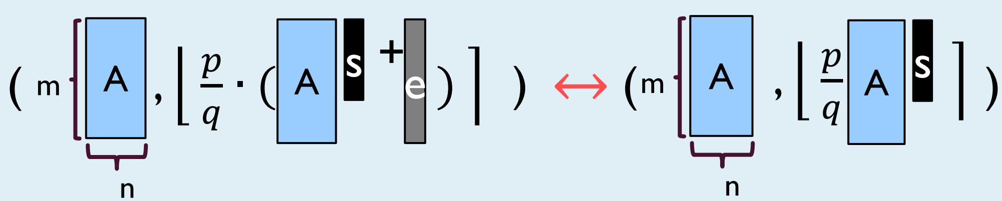


CAUTION! HOW MANY LSBs CAN BE DISCARDED?

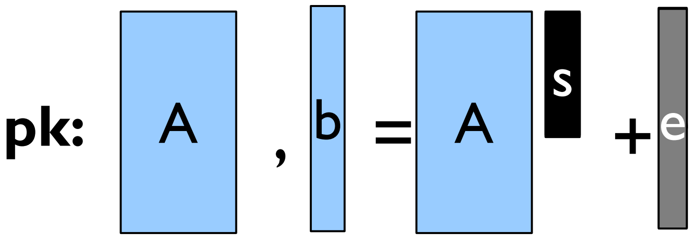
- **(Correctness)** If we cut a large proportion;  , the correctness will not hold.
- **(Security)** We can not remove noise addition in encryption if we cut very small; 

→ Since **the number of samples of LWR** in the Enc procedure is restricted to be **small**, we can choose a proper rounding modulus "p" to satisfy both security and correctness.

<Bogdanov et al.> If the # of samples(m) is no larger than $O(q/Bp)$, we cannot distinguish either one from uniform;

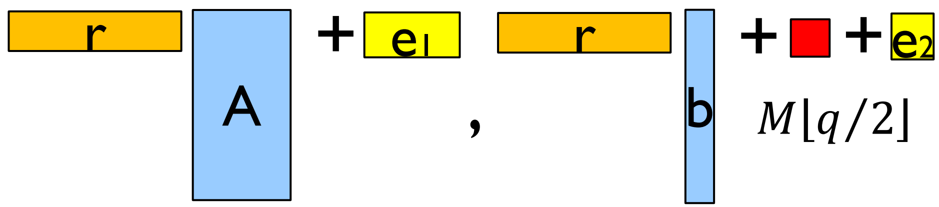


ADVANTAGE OF LWR ASSUMPTION



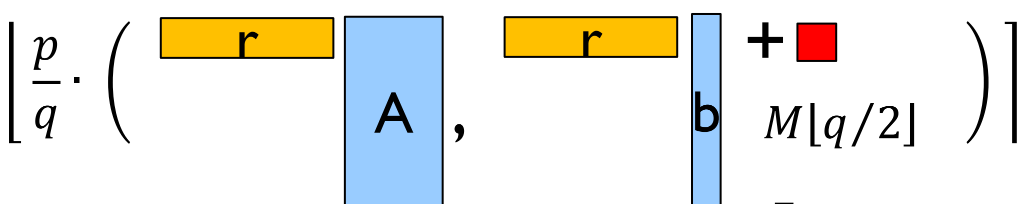
sk = (-s, 1)

LP11.Enc(M)



$Var(e_i) = \sigma^2$

Lizard.Enc(M)



Rounding error (e_1, e_2):
(uniform over $[\pm q/2p]$)
Variance $\sigma^2 = q^2/12p^2$

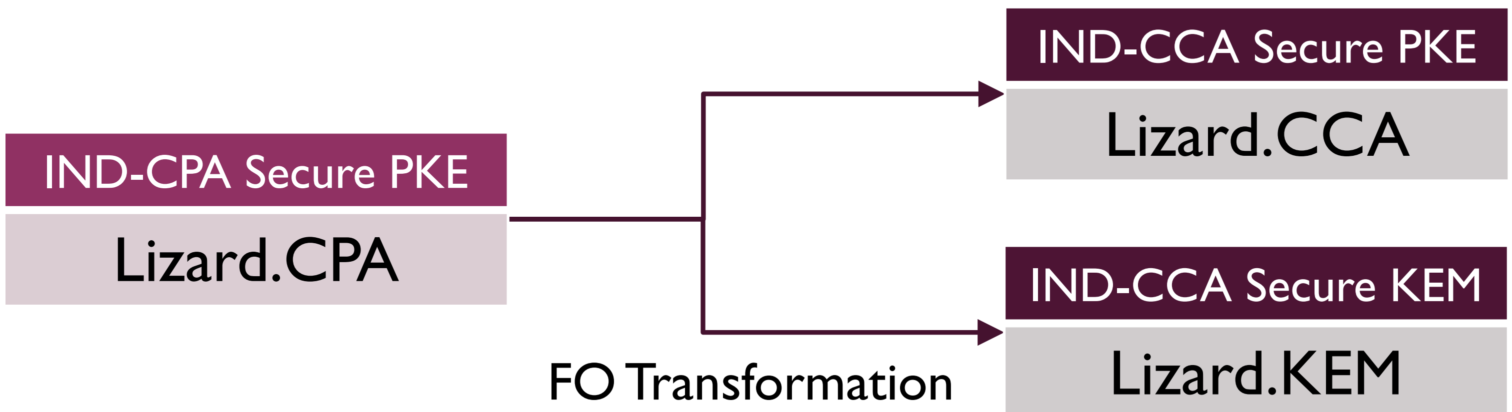
Encryption noise: $\langle r, e \rangle + \langle (e_1, e_2),\ sk \rangle$

Set the parameter $\sigma^2 = q^2/12p^2$: *Preserve cryptanalytic hardness* **LWE(m,q,σ) = LWR(m,q,p) and functionality (encryption noise)**

- **Smaller CTXT**
- **No Gaussian sampling in Encryption**

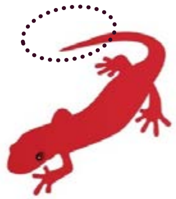
Lizard NIST Submission

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)**	$(a, b) \leftarrow \left(\left\lfloor \frac{a+2^{\ell-1}}{2^\ell} \right\rfloor, \left\lfloor \frac{b+2^{\ell-1}}{2^\ell} \right\rfloor\right)$

$A \in Z_q^{m \times n}$, $B \in Z_q^{m \times \ell}$, (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 for 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

Comparison to Other LWR-based Scheme(s)

SABER

- Submitted by J. P. D' Anvers, A. Karmakar, S. S. Roy, F. Vercauteren – KU Leuven (Belgium)
- Concept: Module-LWR + Module-LWR based
- Main Strengths: Simplicity, Small Parameter Sizes (Smaller pk / sk / ctxt compared to Kyber)
  - Rounding is simple ('add constant and chop' which is the same as Lizard), Secrets from Centered Binomial dist.
  - For 115 bit security (Category 1),

	Kyber	Saber
pk	736 bytes	672 bytes
sk	1632 bytes	992 bytes
ctxt	800 bytes	736 bytes

- No NTT, but Toom-Cook & Karatsuba: Constant-time implementation
- DFR is not $2^{-\lambda}$, but very small
  - 2^{-120}, 2^{-136}, 2^{-165} for Category I, III, V, resp.
- Recently, they reported some implementations on ARM Core

ROUND2

- Submitted by O. Garcia-Morchon, Z. Zhang, S. Bhattacharya, R. Rietman, Ludo Tolh uizen, J.L. Torre-Arce
- Concept: LWR + LWR based, RLWR + RLWR based
- Main Strength: Lower Bandwidths

	uRound2	nRound2	Saber
pk	565 bytes	581 bytes	672 bytes
ctxt	636 bytes	652 bytes	736 bytes
DFR	2^{-66}	2^{-54}	2^{-120}

- Very Similar Approach with Lizard Except for the Higher Dec. Fail. Rates
  - Sparse trinary secret
  - Power-of-2 modulus, rounding by 'add constant and chop'
- uRound2: w/o NTT, nRound2: with NTT

Scheme	Lizard	Saber	Round2
Category	KEM / PKE	KEM	KEM
Main Strength	Fast Enc/Dec Ctxt Compression via LWR Conservative params, negl DFR	Simplicity Fixed ring Compact sizes	Unified Design (GLWR)* Lower bandwidths Great speed*
Assumption	(Ring-)LWE + (Ring-)LWR	Module-LWR	(Ring-)LWR
Ring Choice	$Z_q[X]/(X^n+1)$	$Z_{8192}[X]/(X^{256}+1)$	$Z_q[X]/(X^{p-1}+X^{p-2}+\cdots+1)$
Modulus	Power of 2	Power of 2	Power of 2 / Prime
Dec. Failure	O	O	O
Const. Time	X	O	O (Doubtful)
Some Other Differences	Sparse signed binary secret DFR << $2^{-\lambda}$	Binomial distribution DFR > $2^{-\lambda}$ Toom-Cook and Karatsuba	Sparse signed binary secret DFR: $2^{-65}, 2^{-128} > 2^{-\lambda}$ for all parameter sets NTT Const time*, but still vulnerable to Cache attack*

* ; they insisted so in the submitted documentation

DFR ; Decryption Failure Rate

COMPARISON VIA IMPLEMENTATION (MEASURED ON THE SAME ENVIRONMENT)

Scheme	Parameter Name	KeyGen	Enc	Dec
Round2	u_n1_fn0_l1	4.765	5.436	5.463
	u_n1_fn1_l1	0.432	0.707	0.728
	u_n1_fn2_l1	0.961	1.226	1.248
	u_nd_l1	0.069	0.082	0.089
	n_nd_l1	1.940	3.800	5.652
Lizard	CCA_CATEGORY1_N536	151.485	0.022	0.024
	CCA_CATEGORY1_N663	166.564	0.023	0.026
RLizard	RING_CATEGORY1	0.428	0.028	0.028
Saber*	LightSaber	0.084	0.168	0.245

Table: Performance Comparison for LWR-based Schemes with Category I Parameters

* Saber is a Key Encapsulation Mechanism

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