Lizard: 

A Practical Post-Quantum Public-Key Encryption from LWE and LWR

JUNG HEE CHEON, **DUHYEONG KIM**, JOOHEE LEE, YONGSOO SONG

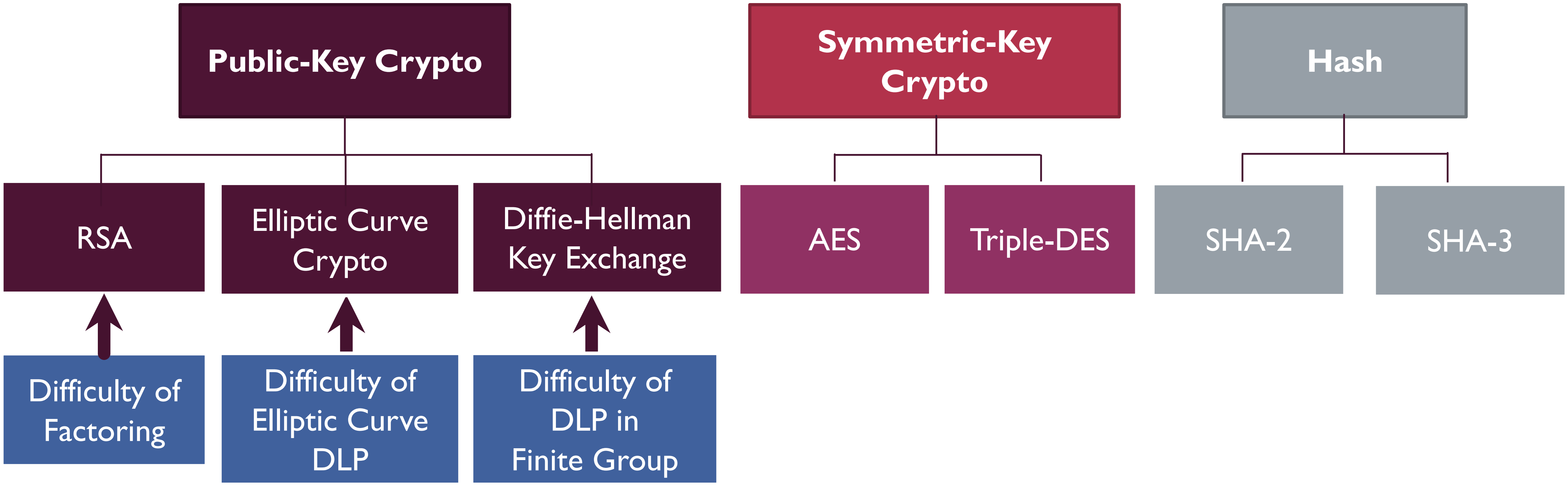
SEOUL NATIONAL UNIVERSITY

2018. 09.05

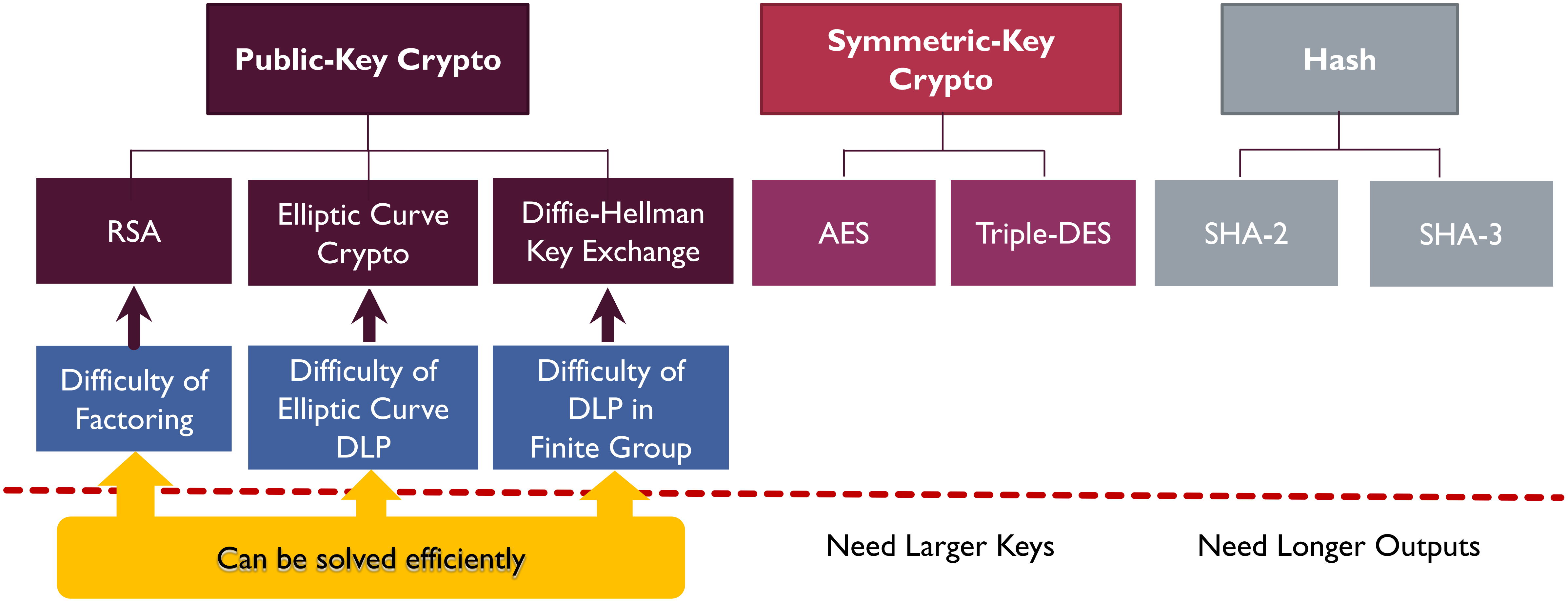
Overview of Lizard

- **[Post-Quantum]** One of the 64 Round1 (accepted) Submissions to NIST's PQC Standardization
- **[Novelty]** the first LWE + LWR based Public-Key Encryption
- **[Design Rationale]** Faster and Simpler!

Uprise of Post-Quantum Cryptography



Uprise of Post-Quantum Cryptography



< Quantum Computing Era >

Uprise of Post-Quantum Cryptography

- **NSA is transitioning to PQC in the "not too distant" future**

  http://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm

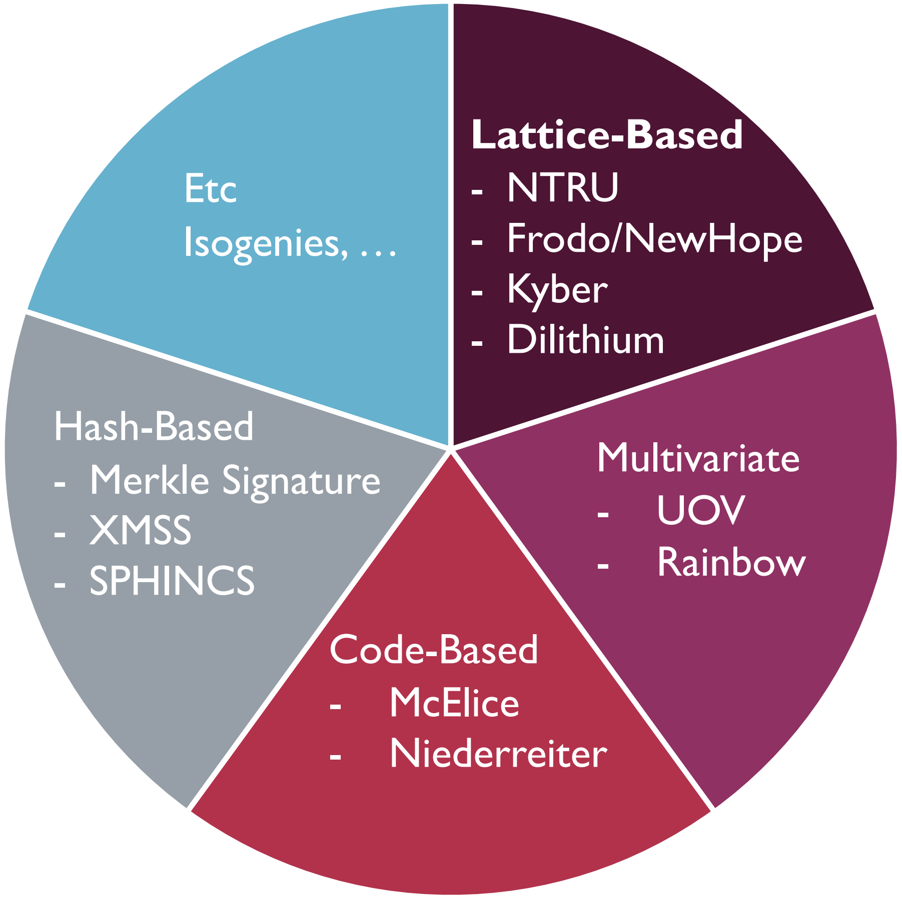
- **NIST launched PQC Standardization Project**

  http://csrc.nist.gov/groups/ST/post-quantum-crypto

  - To standardize Post-Quantum public-key crypto : Encryption / Signature / Key Encapsulation
  - Timeline

Aug 2016	Formal Call for Proposals
Nov 2017	Deadline for Submissions
Apr 2018	1st NIST PQC Workshop
Aug 2019	2nd NIST PQC Workshop (be expected)

Lattice-based Cyptosystem



- **Especially, Lattice-based cryptosystem gains increasing attentions**
  - Security based on the worst-case/average-case reductions from lattice hard problems (SVP, SIVP,…)
  - Fast implementation
  - Versatility in many applications: Homomorphic Encryption, Functional Encryption…
  - 26 of 64 Round1 Submissions to NIST's PQC standardization

LWE-based PKEs

Learning with Errors (LWE)

- Solving a linear equation system is easy! (e.g. Gaussian elimination..)
- Then how hard is it to solve a linear equation "with errors"?

Learning with Errors (LWE)

- Solving a linear equation system is easy! (e.g. Gaussian elimination..)
- Then how hard is it to solve a linear equation "with errors"?

The Learning with Errors Problem!

Learning with Errors (LWE)

- Q. $$\underset{\in Z_{10}^{8\times 3}}{\begin{bmatrix}1&3&7\\4&5&7\\6&6&9\\2&7&3\\3&8&7\\5&4&2\\1&0&5\\4&5&3\end{bmatrix}}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} + \underset{\substack{\text{Small Error}\\\text{(unknown)}}}{\begin{bmatrix}0\\2\\-1\\1\\0\\1\\0\\-2\end{bmatrix}} = \begin{bmatrix}7\\1\\1\\0\\6\\0\\2\\5\end{bmatrix} \pmod{10} \Rightarrow \text{Find } \begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} !$$

; Hard! ($\geq$ GapSVP, SIVP)

Learning with Errors (LWE)

- Q.

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

; Hard! ($\geq$ GapSVP, SIVP)

This is called Search-LWE,
while most of the (so-called) LWE-based cryptosystems are based on "Decisional-LWE"

Decisional-LWE

- 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

from a sample uniform randomly chosen in $Z_{10}^{8\times 4}$!

; Hard! (≥ GapSVP, SIVP)

Decisional-LWE

- Q. Distinguish

$$\begin{bmatrix} 1 & 3 & 7 \\ 4 & 5 & 7 \\ 6 & 6 & 9 \\ 2 & 7 & 3 \\ 3 & 8 & 7 \\ 5 & 4 & 2 \\ 1 & 0 & 5 \\ 4 & 5 & 3 \end{bmatrix}, \begin{bmatrix} 7 \\ 1 \\ 1 \\ 0 \\ 6 \\ 0 \\ 2 \\ 5 \end{bmatrix}$$

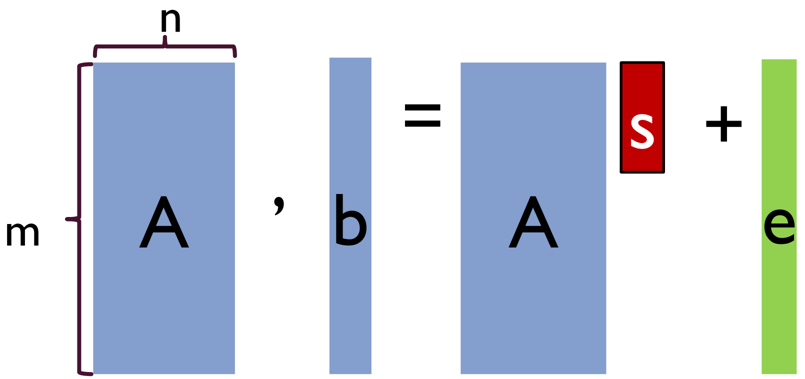
from a sample uniform randomly chosen in $Z_{10}^{8\times 4}$!

; Hard! (≥ GapSVP, SIVP)

From now on,
The term “LWE” always denotes the Decisional-LWE problem

Regev Scheme [Reg05]

- $\mathrm{pk} = \left(A, \vec{b} = A \cdot \color{red}{\vec{s}} + \vec{e}\right) \in Z_q^{m \times n} \times Z_q^m$

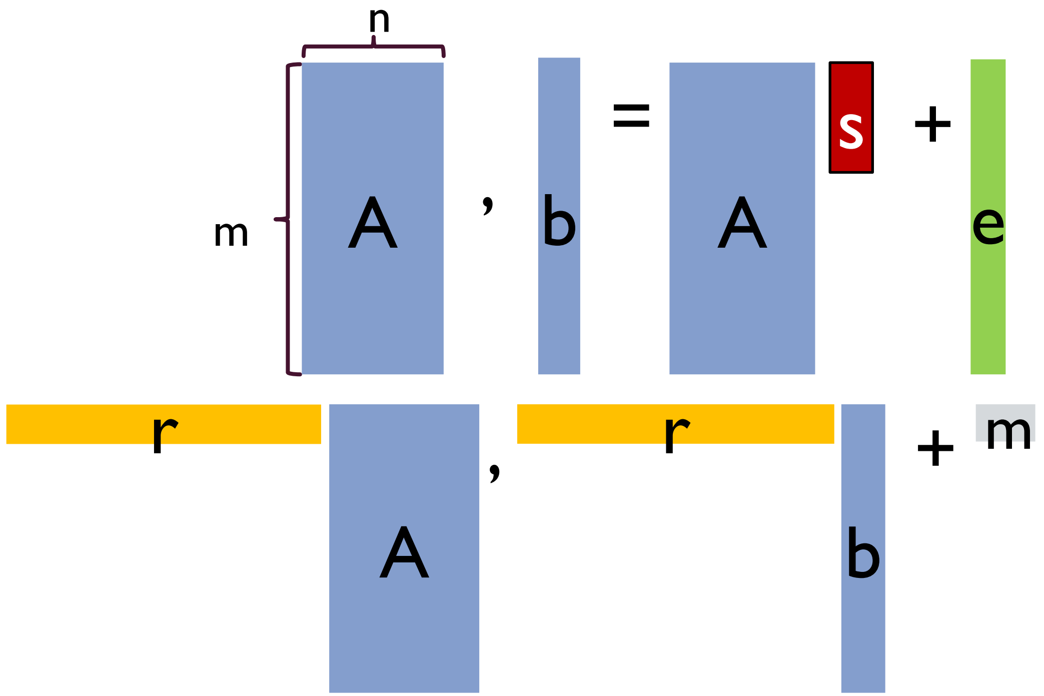


[Reg05] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." *Journal of the ACM (JACM)* 56.6 (2009): 34.

Regev Scheme [Reg05]

- $\text{pk} = \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m \times n} \times Z_q^m$

- $\text{Ctxt} = \left(\vec{r}^t \cdot A, \vec{r}^t \cdot \vec{b} + \left\lfloor \left(\frac{q}{2}\right) \cdot m \right\rceil\right) \in Z_q^n \times Z_q$

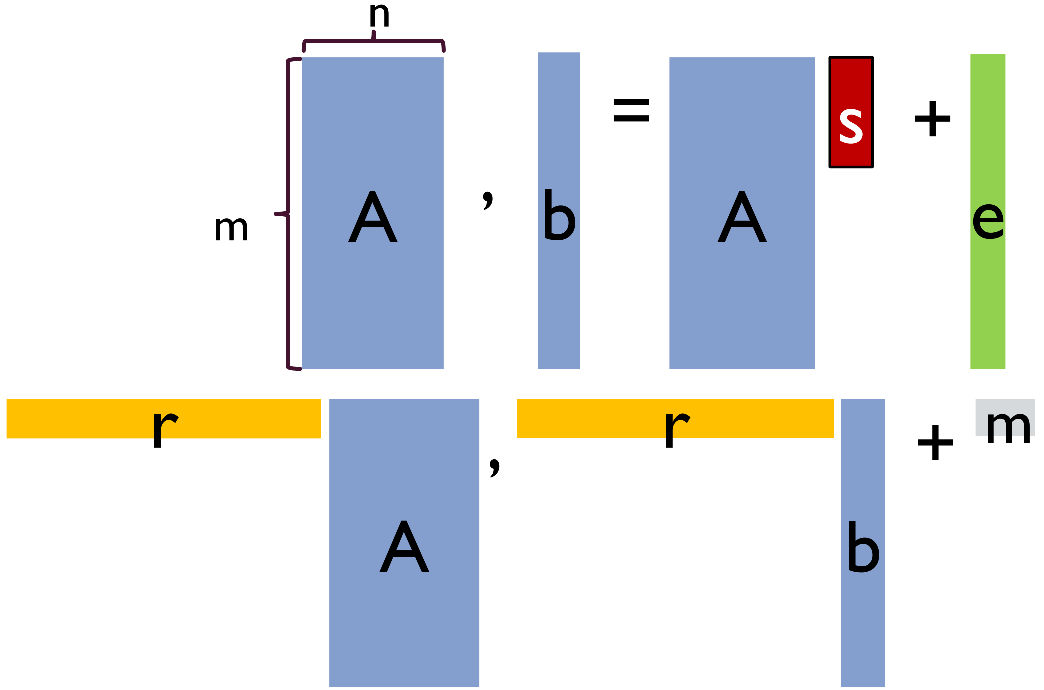


[Reg05] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." *Journal of the ACM (JACM)* 56.6 (2009): 34.

Regev Scheme [Reg05]

- $\text{pk} = \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m\times n} \times Z_q^m$

- $\text{Ctxt} = \left(\vec{r}^t \cdot A, \vec{r}^t \cdot \vec{b} + \left\lfloor \left(\frac{q}{2}\right) \cdot m \right\rceil\right) \in Z_q^n \times Z_q$



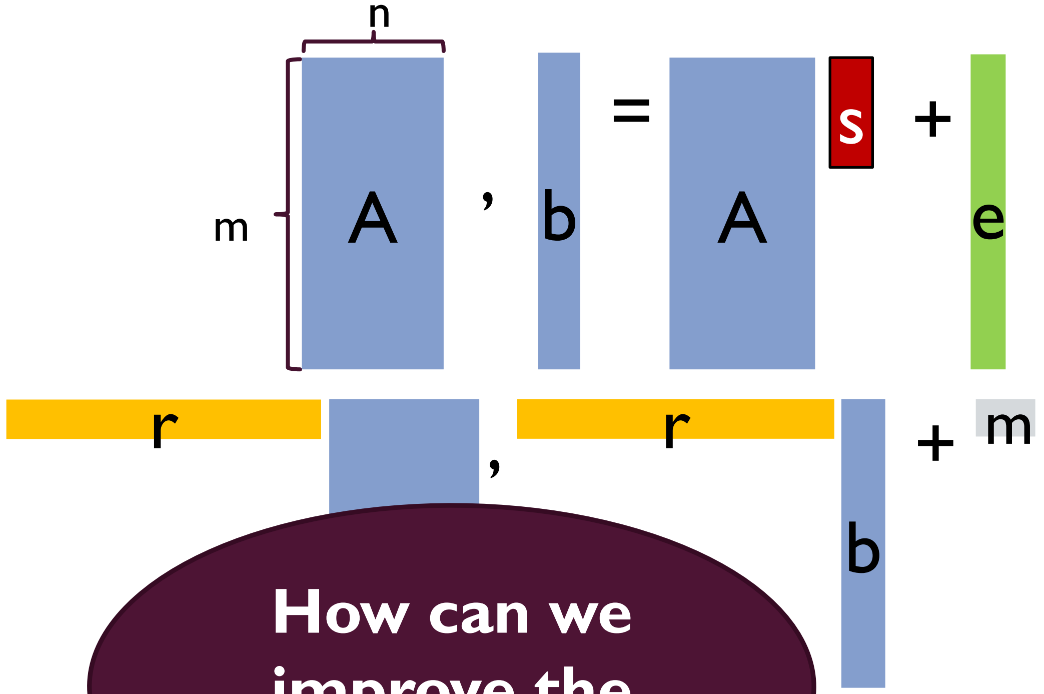
- pk ≈ uniform by LWE assumption (Computational)
- Ctxt ≈ uniform by Leftover Hash Lemma (Statistical)

$\Longrightarrow$ Too large $m = \omega(n \log q)$ $\Longrightarrow$ Too Large public key, Slow encryption

[Reg05] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." *Journal of the ACM (JACM)* 56.6 (2009): 34.

Regev Scheme [Reg05]

- $\text{pk} = \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m\times n} \times Z_q^{m}$
- $\text{Ctxt} = \left(\vec{r}^{\,t} \cdot A, \vec{r}^{\,t} \cdot \vec{b} + \left\lfloor \left(\frac{q}{2}\right) \cdot m \right\rceil\right) \in Z_q^{n} \times Z_q$



How can we improve the efficiency?

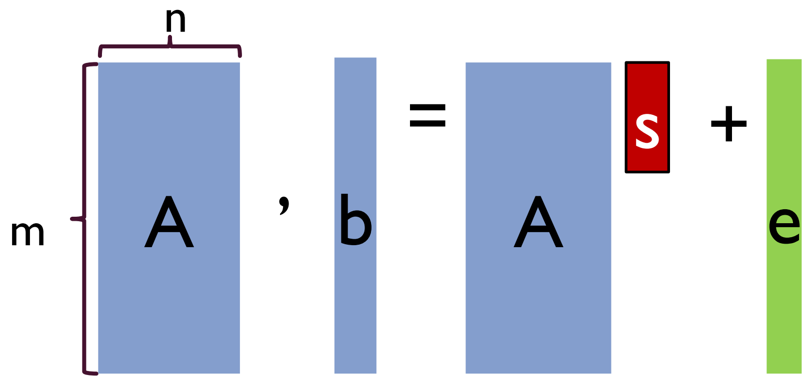
- pk ≈ uniform by LWE assumption (Computational)
- Ctxt ≈ uniform by Leftover Hash Lemma (Statistical)

$\Longrightarrow$ Too large $m = \omega(n \log q)$ $\Longrightarrow$ Too Large public key, Slow encryption

[Reg05] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." *Journal of the ACM (JACM)* 56.6 (2009): 34.

Lindner-Peikert Scheme [LP11]

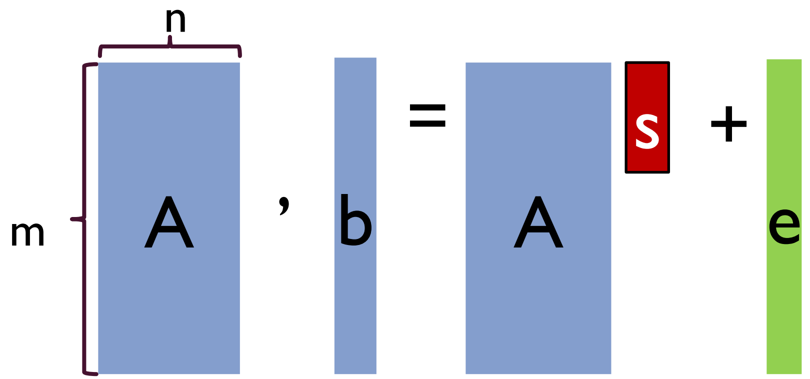
- $\mathrm{pk} = \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m \times n} \times Z_q^m$



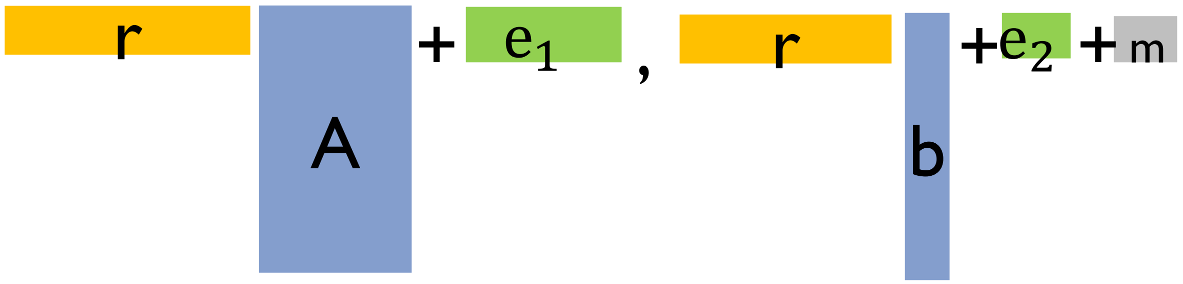
[LP11] R. Lindner, and C. Peikert. "Better key sizes (and attacks) for LWE-based encryption." *Cryptographers' Track at the RSA Conference*. Springer, Berlin, Heidelberg, 2011.

Lindner-Peikert Scheme [LP11]

- pk $= \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m\times n} \times Z_q^{m}$



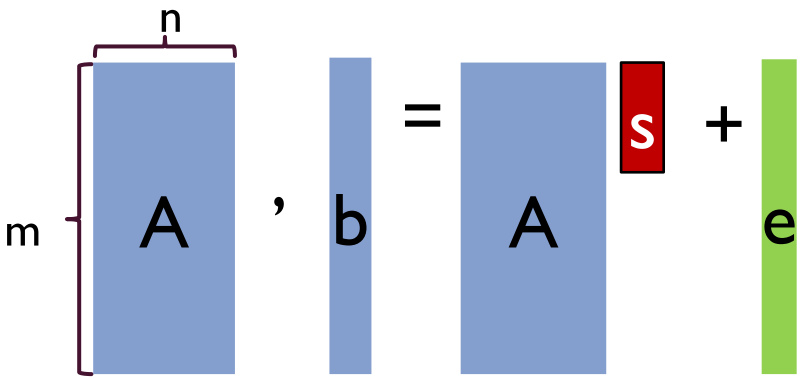
- Ctxt $= \left(\vec{r}^{\,t} \cdot A + \vec{e_1}, \vec{r}^{\,t} \cdot \vec{b} + e_2 + \left\lfloor \left(\frac{q}{2}\right) \cdot m \right\rceil\right) \in Z_q^{n} \times Z_q$



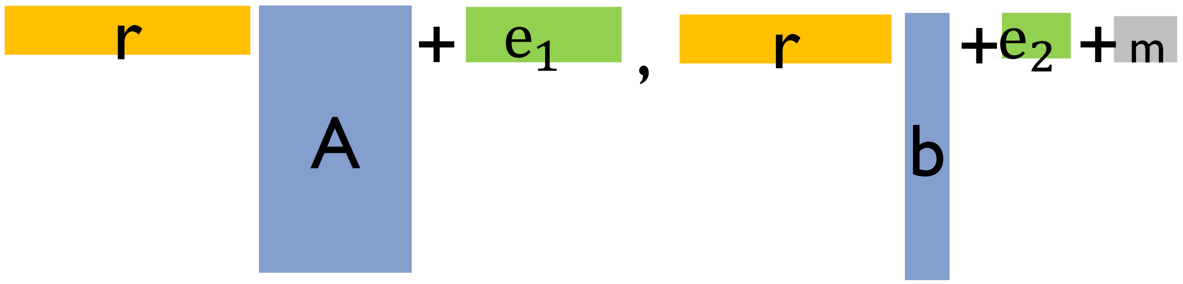
[LP11] R. Lindner, and C. Peikert. "Better key sizes (and attacks) for LWE-based encryption." *Cryptographers'Track at the RSA Conference*. Springer, Berlin, Heidelberg, 2011.

Lindner-Peikert Scheme [LP11]

- $\text{pk} = \left(A, \vec{b} = A \cdot \vec{s} + \vec{e}\right) \in Z_q^{m\times n} \times Z_q^{m}$



- $\text{Ctxt} = \left(\vec{r}^{\,t} \cdot A + \vec{e_1}, \vec{r}^{\,t} \cdot \vec{b} + e_2 + \left\lfloor \left(\frac{q}{2}\right) \cdot m \right\rceil\right) \in Z_q^{n} \times Z_q$



- pk ≈ uniform by LWE assumption (Computational)
- Ctxt ≈ uniform by ~~Leftover Hash Lemma~~ LWE assumption (Computational)

  $\Longrightarrow$ Smaller parameter m!

[LP11] R. Lindner, and C. Peikert. "Better key sizes (and attacks) for LWE-based encryption." *Cryptographers' Track at the RSA Conference*. Springer, Berlin, Heidelberg, 2011.

Lindner-Peikert Scheme [LP11]

- By substituting Leftover Hash Lemma by the LWE assumption, Discrete Gaussian Sampling is required in every encryption stage.

- How to deal with Discrete Gaussian Sampling? 

1. High-bit precision / exact Sampling [GPV08, BLP+13] $\Longrightarrow$ rather slow performance
2. Inverse Sampling Method via look-up table [BCD+16]

$\Longrightarrow$ much faster! But it still consumes a substantial portion of the encryption phase.

[GPV08] Gentry, Craig, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008.
[BLP+13] Brakerski, Zvika, et al. "Classical hardness of learning with errors." *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013.
[BCD+16] Bos, Joppe, et al. "Frodo: Take off the ring! practical, quantum-secure key exchange from LWE." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

LWE + LWR based PKE Lizard

Learning with Rounding (LWR)

- Proposed by Banerjee-Peikert-Rosen in Eurocrypt'12 [BPR12]
- Instead of adding an error in LWE, just discard some least significant bits

  $\Longrightarrow$"derandomized" LWE

Learning with Rounding (LWR)

- Proposed by Banerjee-Peikert-Rosen in Eurocrypt'12 [BPR12]
- Instead of adding an error in LWE, just discard some least significant bits

  $\Longrightarrow$ "derandomized" LWE
- The LWR problem is to distinguish m samples

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

  from m samples uniformly chosen in $Z_q^n \times Z_p$

[BPR12] Banerjee, Abhishek, Chris Peikert, and Alon Rosen. "Pseudorandom functions and lattices." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2012.

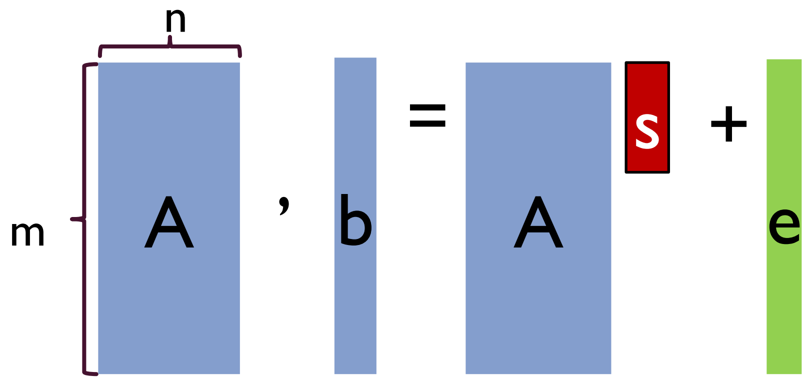
Hardness of LWR

- Researches on the hardness of LWR, which commonly aim to show LWE $\leq$ LWR
  - [BPR12]: a super-polynomial modulus $q \geq p \cdot B \cdot n^{\omega(1)}$ where B is an upper bound of LWE errors
  - [AKPW13]: a polynomial modulus and modulus-to-error ratio
  - [BGM+16]: more general modulus, a bounded number of samples $m = O(\frac{q}{Bp})$

[BPR12] Banerjee, Abhishek, Chris Peikert, and Alon Rosen. "Pseudorandom functions and lattices." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2012.
[AKPW13] Alwen, Joël, et al. "Learning with rounding, revisited." *Advances in Cryptology–CRYPTO 2013*. Springer, Berlin, Heidelberg, 2013. 57-74.
[BGM+16] Bogdanov, Andrej, et al. "On the hardness of learning with rounding over small modulus." *Theory of Cryptography Conference*. Springer, Berlin, Heidelberg, 2016.

The Lizard PKE scheme

- **KeyGen**: the same with other LWE-based PKEs



The Lizard PKE scheme

- **KeyGen**: the same with other LWE-based PKEs

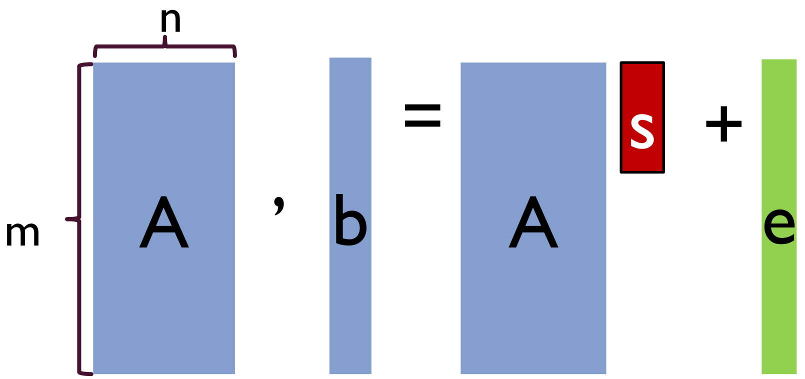
$$\underbrace{A}_{n},\ b = A\,s + e \qquad (A \in \mathbb{Z}^{m \times n})$$

- **Encryption**: Take a rounding instead of adding some errors

$$\mathrm{Ctxt} = \left(\left\lfloor \frac{p}{q} \cdot r\,A \right\rceil, \left\lfloor \frac{p}{q} \cdot r\,b \right\rceil + \left\lfloor \frac{p}{2} \cdot m \right\rceil \right)$$

The Lizard PKE scheme

- **KeyGen**: the same with other LWE-based PKEs



- **Encryption**: Take a rounding instead of adding some errors

$$\mathrm{Ctxt} = \left(\left\lfloor \frac{p}{q} \cdot r A \right\rceil, \left\lfloor \frac{p}{q} \cdot r b \right\rceil + \left\lfloor \frac{p}{2} \cdot m \right\rceil \right)$$

- **Decryption**: Compute and Output $\left\lfloor \left(\frac{2}{p}\right) \cdot (c_2 - \vec{c_1} \cdot \vec{s}) \right\rceil$

The Lizard PKE scheme

- **KeyGen**: the same with other LWE-based PKEs

$$\begin{array}{cc} & n \\ m & A \end{array}, \quad b = A s + e$$

- **Encryption**: Take a rounding instead of adding some errors

$$\text{Ctxt} = \left(\left\lfloor \frac{p}{q} \cdot r A \right\rceil, \left\lfloor \frac{p}{q} \cdot r b \right\rceil + \left\lfloor \frac{p}{2} \cdot m \right\rceil \right)$$

- **Decryption**: Compute and Output $\left\lfloor \left(\frac{2}{p}\right) \cdot (c_2 - \vec{c_1} \cdot \vec{s}) \right\rceil$

Satisfy IND-CPA Security under LWE and LWR assumptions

Parameter Choices

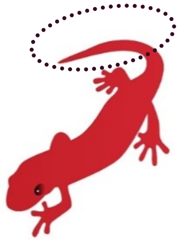
- In practice, the performance highly depends on the choice of secret distributions and parameters
- **Our Choices** for Efficiency:
  - The secret key s is chosen as a binary vector
  - The ephemeral secret vector r is chosen as a sparse binary vector

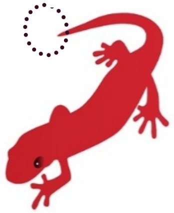
    $\Longrightarrow$ Faster computation of $(\vec{r}^t \cdot A, \vec{r}^t \cdot \vec{b})$
  - Moduli q and p are chosen to be power-of-two

    $\Longrightarrow$ Simpler rounding process done by just adding a constant and then right-shift (almost for free)

$$\left(\text{e.g.}, \left\lfloor \left(\frac{p}{q}\right) \cdot x \right\rceil = \left\lfloor \left(\frac{p}{q}\right) \cdot \left(x + \frac{q}{2p}\right) \right\rfloor\right)$$

Caution! - How many LSBs can be discarded?

- **(Correctness)** If we cut a large proportion  , the correctness will not hold.

- **(Security)** If we cut too small proportion  , the security of a message will not hold.

$\Longrightarrow$ We choose a proper rounding modulus "p" to satisfy both security and correctness.

Features of Lizard

- **Design Rationale; boost up the Encryption speed**
  - Encrypting w/o Discrete Gaussian Sampling
  - Use sparse binary ephemeral secret $\vec{r}$.
- **Feasible parameters**
  - "Slam dunk"; achieve all the negligible dec.fail.rate & conservative quantum security & better efficiency
  - Parameters were chosen by a core-SVP hardness methodology proposed in NewHope [ADPS16] considering all best known attacks (dual attack, primal attack)
- **Smaller Ciphertext size**
  - By discarding some LSBs, the ciphertext size is reduced with the factor $\frac{\log p}{\log q}$

[ADPS16] Alkim, Erdem, et al. "Post-quantum Key Exchange-A New Hope." *USENIX Security Symposium*. Vol. 2016. 2016.

Implementation

CCALizard: IND-CCA variant of Lizard

- Apply a variant of Fujisaki-Okamoto CCA conversion [HHK17]

  (HHK conversion : IND-CPA PKE $\Longrightarrow$ IND-CCA KEM)

- IND-CCA PKE = IND-CCA KEM + One-Time Pad

$\Longrightarrow$ CCALizard satisfies IND-CCA security under Quantum Random Oracle Model

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

Performance and Comparison of PKEs

- Encrypting a 256-bit plaintext with quantum 128-bit security

Scheme	Enc	Dec	Ctxt (bytes)
RSA-3072	116,894	8,776,864	75
NTRU EES743EP1	102,008	109,582	980
CCA-CHK+	≈ 813,800	≈ 785,200	804
CCALizard	**32,272**	**47,125**	**955**

[Table] Performance of our Enc/Dec procedures in cycles

- Our schemes were measured on an Intel Xeon E5- 2620 CPU running at 2.10GHz w/o Turbo Boost and Multithreading.
- CCA-CHK+ : [CHK+16], measured on Macbook pro Intel core i5 running at 2.60GHz
- RSA, NTRU schemes: measured on a PC with Intel quad-core i5-6600 running at 3.3 GHz processor, drawn from ECRYPT Benchmarking of Crypto Systems.
- RSA does not achieve a quantum 128-bit security.
- CCA-CHK+ achieves only a *quantum 58-bit security w.r.t. a core-SVP hardness methodology*

thank you!