Lizard

Carl A. Miller
NIST Computer Security Division
August 14, 2018

NIST PQC Seminar (not for public distribution)

The Basics

- It's a public key encryption scheme and a key encapsulation scheme.

- It's a lattice-based scheme that exploits LWE (Learning with Errors) and LWR (Learning with Rounding).

- It has IND-CPA (chosen plaintext attack) and IND-CCA2 (adaptive chosen ciphertext attack) versions.

Simplified Protocols

(Based on this submission and [Regev 2010])

Learning With Errors

Suppose that $\mathbf{s}$ is an unknown vector in $\mathbb{Z}_q^n$, and that we know several approximate linear relations (mod q):

$$\begin{aligned} \mathbf{a}_1 \cdot \mathbf{s} &\approx b_1 \\ \mathbf{a}_2 \cdot \mathbf{s} &\approx b_2 \\ &\vdots \\ \mathbf{a}_m \cdot \mathbf{s} &\approx b_m \end{aligned}$$

Here, $\mathbf{a}_i \in \mathbb{Z}_q^n$ and $b_j \in \mathbb{Z}_q$. Can we determine $\mathbf{s}$?

Learning With Errors

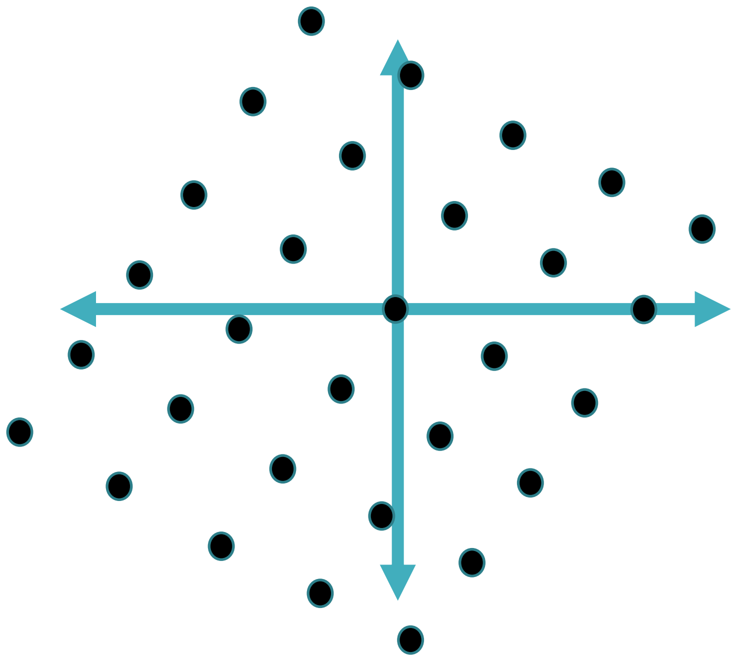
More precisely, suppose that we are given the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$ and we are given the values

$$\begin{aligned} b_1 &:= \mathbf{a}_1 \cdot \mathbf{s} + e_1 \\ b_2 &:= \mathbf{a}_2 \cdot \mathbf{s} + e_2 \\ &\vdots \\ b_m &:= \mathbf{a}_m \cdot \mathbf{s} + e_m \end{aligned}$$

where $e_1, \ldots, e_m \in \mathbb{Z}_q$ are chosen according to a discrete Gaussian distribution (with variance much smaller than q).

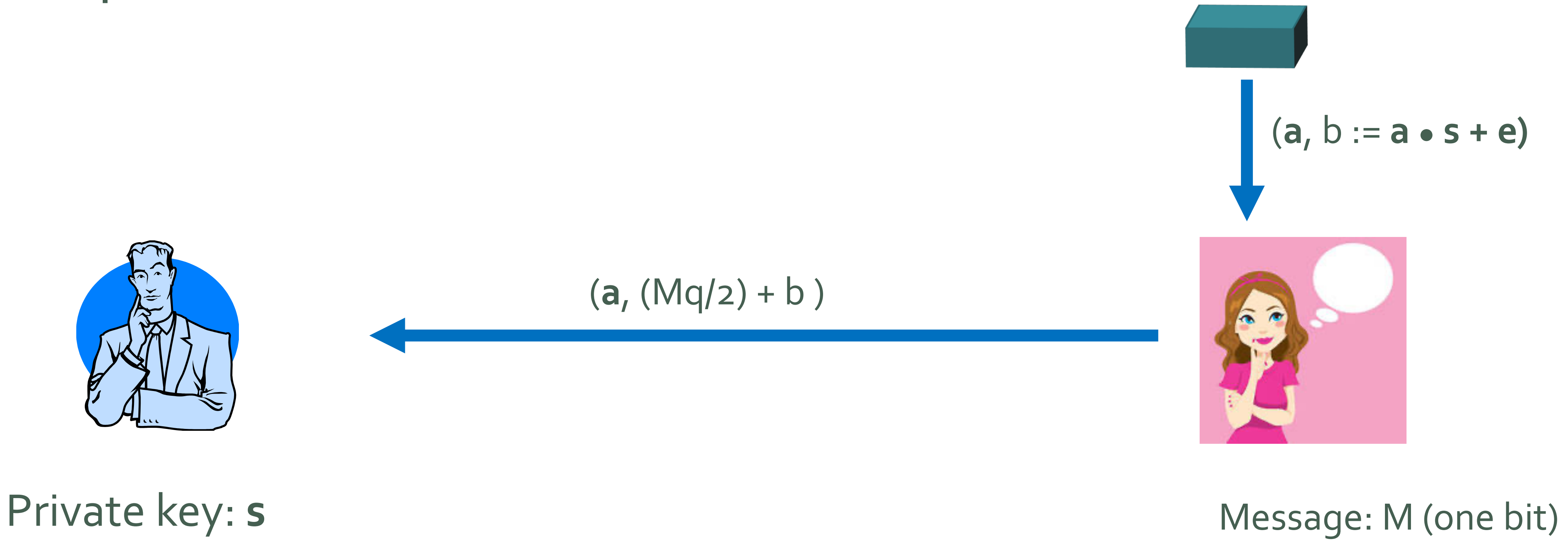
Learning With Errors

LWE is at least as hard as determining the length of the shortest vector in a lattice.



Encryption with LWE

Suppose Alice has access to a black box that generates LWE samples.



Encryption with LWE

Bob can approximately determine b, and therefore determine m. But, to everyone else the msg. looks random.

$(\mathbf{a}, b := \mathbf{a} \bullet \mathbf{s} + \mathbf{e})$

$(\mathbf{a}, (Mq/2) + b)$

Private key: **s**

Message: M (one bit)

How does Alice generate LWE samples?

Alice adds & subtracts random equations from this system to get a new equation.

Public key →

$$\begin{aligned}
b_1 &:= \mathbf{a}_1 \cdot \mathbf{s} + e_1 && \times 0\\
b_2 &:= \mathbf{a}_2 \cdot \mathbf{s} + e_2 && \times (-1)\\
b_3 &:= \mathbf{a}_3 \cdot \mathbf{s} + e_3 && \times 0\\
b_4 &:= \mathbf{a}_4 \cdot \mathbf{s} + e_4 && \times (+1)\\
b_5 &:= \mathbf{a}_5 \cdot \mathbf{s} + e_5 && \times 0\\
b_6 &:= \mathbf{a}_6 \cdot \mathbf{s} + e_6 && \times 0\\
&\vdots\\
b_m &:= \mathbf{a}_m \cdot \mathbf{s} + e_m && \times (-1)
\end{aligned}$$

$b = \mathbf{a} \bullet \mathbf{s} + \mathbf{e}$

↑ (unknown) Gaussian in $\mathbf{Z}_q$

Encryption with LWE

The linear combination of equations from the public key gives Alice an equation she can use for the transmission.

($\mathbf{a}$, (Mq/2) + b)

Private key: $\mathbf{s}$

Message: M (one bit)
Public key: $\{\mathbf{a}_i\}$, $\{b_i\}$

Alternatively …

The public key can consist of equations with different **s**'s.

Public key

$$\begin{aligned} b_1 &:= \mathbf{a}_1 \cdot \mathbf{s} + e_1 & \times\ 0 \\ b_2 &:= \mathbf{a}_2 \cdot \mathbf{s} + e_2 & \times\ (-1) \\ b_3 &:= \mathbf{a}_3 \cdot \mathbf{s} + e_3 & \times\ 0 \\ b_4 &:= \mathbf{a}_4 \cdot \mathbf{s} + e_4 & \times\ (+1) \\ b_5 &:= \mathbf{a}_5 \cdot \mathbf{s} + e_5 & \times\ 0 \\ b_6 &:= \mathbf{a}_6 \cdot \mathbf{s} + e_6 & \times\ 0 \\ &\vdots \\ b_m &:= \mathbf{a}_m \cdot \mathbf{s} + e_m & \times\ (-1) \end{aligned}$$

Alternatively …

The public key can consist of equations with different **s**'s.

Public key →

$$
\begin{aligned}
b_1 &:= \mathbf{a}_1 \cdot \mathbf{s}_1 + e_1 && \times 0 \\
b_2 &:= \mathbf{a}_2 \cdot \mathbf{s}_2 + e_2 && \times (-1) \\
b_3 &:= \mathbf{a}_3 \cdot \mathbf{s}_3 + e_3 && \times 0 \\
b_4 &:= \mathbf{a}_4 \cdot \mathbf{s}_4 + e_4 && \times (+1) \\
b_5 &:= \mathbf{a}_5 \cdot \mathbf{s}_5 + e_5 && \times 0 \\
b_6 &:= \mathbf{a}_6 \cdot \mathbf{s}_6 + e_6 && \times 0 \\
&\vdots \\
b_m &:= \mathbf{a}_m \cdot \mathbf{s}_m + e_m && \times (-1)
\end{aligned}
$$

A Hypothetical Protocol

Bob computes a uniformly random matrix A, and sends AS + E.

Public key: B = AS + E

Private key: S in $\mathbf{Z}_q^{n \times m}$

Message: M (one bit)

A Hypothetical Protocol

Bob computes a uniformly random matrix A, and sends A**S** + E.
Alice sends back her encryption of M.
Bob computes $S^\top A^\top r \approx B^\top r$, and recovers M.

Public key: B = AS + E

$(A^\top \mathbf{r}, (mq/2) + B^\top \mathbf{r})$

Private key: S in $\mathbf{Z}_q^{n\times m}$

Message: M (one bit)
Mask vector: **r** in $\{-1, 0, 1\}^m$

Main Protocols

Learning With Rounding (LWR)

Let $p \mid q$ and $\mathbf{s} \in \mathbb{Z}_q^n$. In LWR, we are given the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{Z}_q^n$ and we are given the values

$$\begin{aligned} b_1 &:= \lfloor (p/q)\mathbf{a}_1 \cdot \mathbf{s} \rceil \in \mathbb{Z}_p \\ b_2 &:= \lfloor (p/q)\mathbf{a}_2 \cdot \mathbf{s} \rceil \in \mathbb{Z}_p \\ &\vdots \\ b_m &:= \lfloor (p/q)\mathbf{a}_m \cdot \mathbf{s} \rceil \in \mathbb{Z}_p \end{aligned}$$

($\lfloor \cdot \rceil$ = "round off to the nearest integer.")

Lizard.CPA.KeyGen.

Operation:

1. Generate a random matrix $A \leftarrow \mathbb{Z}_q^{m \times n}$.
2. Set a secret matrix $S := (\mathbf{s}_0 \| \cdots \| \mathbf{s}_{\ell-1})$ by sampling each $\mathbf{s}_i$ independently from the distribution $\mathcal{ZO}_n(\rho)$.
3. For $0 \leq i \leq m-1$ and $0 \leq j \leq \ell - 1$, sample an integer $E_{ij} \leftarrow \mathcal{DG}_{\alpha q}$, and then set $E = (E_{ij}) \in \mathbb{Z}_q^{m \times \ell}$.
4. Compute $B := -AS + E \in \mathbb{Z}_q^{m \times \ell}$.
5. Output the public key $\mathsf{pk} := (A\|B) \in \mathbb{Z}_q^{m \times (n+\ell)}$ and the private key $\mathsf{sk} := S \in \{-1, 0, 1\}^{n \times \ell}$.

Vectors with entries in {-1, 0, 1}

Discrete Gaussian

Lizard.CPA.Enc.

Vectors with entries in {-1, 0, 1}

Operation:

1. Generate an m dimensional vector $\mathbf{r} \in B_{m,h_r}$ from the distribution $\mathcal{HWT}_m(h_r)$.
2. Compute $\mathbf{a} := \lfloor (p/q) \cdot A^t \mathbf{r} \rceil \in \mathbb{Z}_p^n$ and $\mathbf{b} := \lfloor (p/q) \cdot ((q/2) \cdot \mathbf{M} + B^t \mathbf{r}) \rceil \in \mathbb{Z}_p^\ell$.
3. Output the ciphertext $\mathbf{c} := (\mathbf{a}, \mathbf{b}) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^\ell$.

Lizard.CPA.Dec.

Operation:

1. Parse the ciphertext $\mathbf{c} = (\mathbf{a}, \mathbf{b})$.
2. Compute $\mathbf{M} = \lfloor (2/p) \cdot (\mathbf{b} + S^t \mathbf{a}) \rceil \in \mathbb{Z}_2^\ell$.
3. Output the message $\mathbf{M}$.

Protocols in Polynomial Rings

The protocols

RLizard.CPA.KeyGen

RLizard.CPA.Enc

RLizard.CPA.Dec

are similar, except that that the matrices are restricted to

$$\mathcal{R}_q := \mathbb{Z}_q[x]/\left(x^n + 1\right)$$

(which is a subring of $\mathbb{Z}_q^{n \times n}$.)

IND-CCA2 Protocols

Lizard.CCA.Enc.

Operation:

1. Generate a random vector $\boldsymbol{\delta} \leftarrow \{0,1\}^{\ell}$.
2. Set $\mathbf{c}_1 := \mathbf{M} \oplus G(\boldsymbol{\delta}) \in \mathbb{Z}_2^d$ and $\mathbf{c}_3 := H'(\boldsymbol{\delta})$.
3. Set $\mathbf{r} := H(\boldsymbol{\delta}) \in \{-1,0,1\}^m$.
4. Compute $\mathbf{a} := \lfloor (p/q) \cdot A^t \mathbf{r} \rceil \in \mathbb{Z}_p^n$ and $\mathbf{b} := \lfloor (p/q) \cdot ((q/2) \cdot \boldsymbol{\delta} + B^t \mathbf{r} \rceil) \in \mathbb{Z}_p^{\ell}$.
5. Output $\mathbf{c} = (\mathbf{c}_1, (\mathbf{a}, \mathbf{b}), \mathbf{c}_3)$.

A random bit string is used both to pad the message, and to choose **r**.

Possible parameters:

q = 2048	d = 384	m = 1024
p = 512	n = 816	ell = 384

Analyses & Performance

Security Proofs

The authors prove that the original protocol is IND-CPA secure, under the assumption that both LWE and LWR distributions are indistinguishable from random.

Idea (?): Replacing the public key and the ciphertext with a random string makes only a negligible amount of difference, so an adversary can get only a negligible amount of information from both.

Key & Message Sizes

Operations	Parameter	Plaintext (bytes)	Ciphertext (bytes)	Public Key (bytes)	Private Key (bytes)
Lizard.CCA	CCA_CATEGORY1_N536	32	1,648	1,622,016	137,216
	CCA_CATEGORY1_N663	32	983	1,882,112	169,728
	CCA_CATEGORY3_N816	48	2,496	2,457,600	313,344
	CCA_CATEGORY3_N952	48	2,768	2,736,128	365,568
	CCA_CATEGORY5_N1088	64	3,328	6,553,600	557,056
	CCA_CATEGORY5_N1300	64	3,752	3,710,976	665,600
RLizard.CCA	RING_CATEGORY1	32	2,208	4,096	257
	RING_CATEGORY3_N1024	48	4,272	4,096	513
	RING_CATEGORY3_N2048	48	8,496	8,192	369
	RING_CATEGORY5	64	8,512	8,192	513

Table 4: Size of Lizard.CCA and RLizard.CCA

Performance

Operations	Parameter	KeyGen (ms)	Enc (ms)	Dec (ms)
Lizard.CCA	CCA_CATEGORY1_N536	156.320	0.031	0.034
	CCA_CATEGORY1_N663	176.570	0.032	0.036
	CCA_CATEGORY3_N816	250.555	0.052	0.064
	CCA_CATEGORY3_N952	275.555	0.057	0.072
	CCA_CATEGORY5_N1088	663.879	0.062	0.086
	CCA_CATEGORY5_N1300	392.828	0.071	0.101
RLizard.CCA	RING_CATEGORY1	0.449	0.036	0.039
	RING_CATEGORY3_N1024	0.513	0.057	0.075
	RING_CATEGORY3_N2048	0.875	0.078	0.093
	RING_CATEGORY5	0.920	0.108	0.135

Table 5: Performance of Lizard.CCA and RLizard.CCA

Hardware Implementation?

Architecture of Lizard.CPA The Fig. 1 shows the hardware architecture of Lizard.CPA.

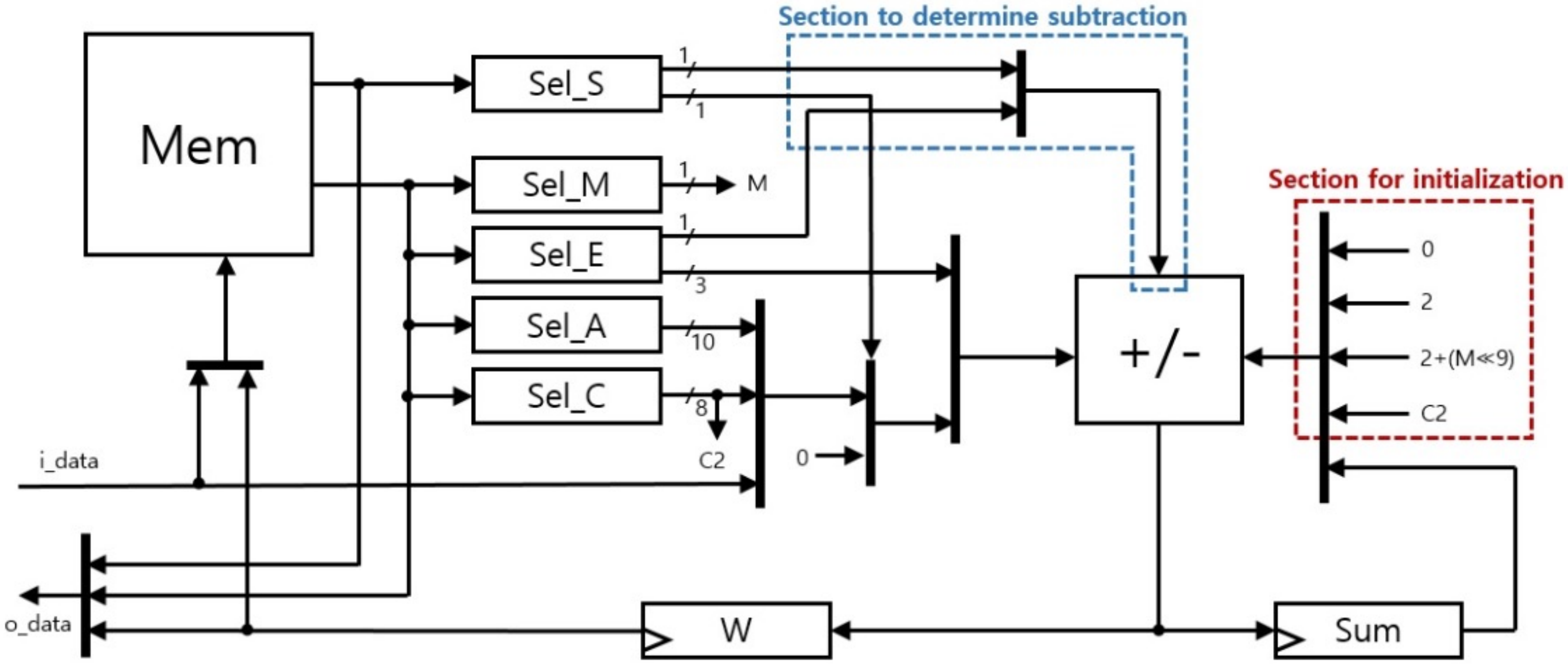


Fig. 1: Data path of Lizard.CPA

Claimed Advantages

The protocol is efficient in cases where the message space is small (e.g., 32 bits)?

Because of the structure of the encryption, a receiver can ”add” plaintexts without decrypting them (i.e., limited homomorphic encryption).

Lizard

Carl A. Miller
NIST Computer Security Division
August 14, 2018

NIST PQC Seminar (not for public distribution)