Learning With Errors (LWE)

... or " What if Gauss had been a little lazier?"

Fredrik Meisingseth

Winter 2020

Contents

“Learning without errors”

Consider the equation system:

$$
\begin{aligned}
\boldsymbol{s} * \boldsymbol{a}_1 &= b_1 \; (mod \;\; p) \\
\boldsymbol{s} * \boldsymbol{a}_2 &= b_2 \; (mod \;\; p) \\
&\ldots \\
\boldsymbol{s} * \boldsymbol{a}_m &= b_m \; (mod \;\; p)
\end{aligned}
$$

,where p prime, $\boldsymbol{s} \in \mathbb{Z}_p^n$, $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$. Find $\boldsymbol{s}$.

"Learning without errors"

Consider the equation system:

$$\begin{aligned} \boldsymbol{s} * \boldsymbol{a}_1 &= b_1 \ (mod \ \ p) \\ \boldsymbol{s} * \boldsymbol{a}_2 &= b_2 \ (mod \ \ p) \\ &\dots \\ \boldsymbol{s} * \boldsymbol{a}_m &= b_m \ (mod \ \ p) \end{aligned}$$

,where p prime, $\boldsymbol{s} \in \mathbb{Z}_p^n$, $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$. Find $\boldsymbol{s}$.
Easy to solve, use Gauss elimination!

"Learning without errors"

Consider the equation system:

$$\begin{aligned} \boldsymbol{s} * \boldsymbol{a}_1 &= b_1 \ (mod \ p) \\ \boldsymbol{s} * \boldsymbol{a}_2 &= b_2 \ (mod \ p) \\ &\ldots \\ \boldsymbol{s} * \boldsymbol{a}_m &= b_m \ (mod \ p) \end{aligned}$$

,where p prime, $\boldsymbol{s} \in \mathbb{Z}_p^n$, $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$. Find $\boldsymbol{s}$.

Easy to solve, use Gauss elimination!

<u>What if it was approximate instead?</u>

"Learning without errors"

Consider the equation system:

$$
\begin{aligned}
\boldsymbol{s} * \boldsymbol{a}_1 &= b_1 \ (mod \ \ p) \\
\boldsymbol{s} * \boldsymbol{a}_2 &= b_2 \ (mod \ \ p) \\
&\ldots \\
\boldsymbol{s} * \boldsymbol{a}_m &= b_m \ (mod \ \ p)
\end{aligned}
$$

,where p prime, $\boldsymbol{s} \in \mathbb{Z}_p^n$, $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$. Find $\boldsymbol{s}$.

Easy to solve, use Gauss elimination!

What if it was approximate instead?

Does it make the problem harder?

Learning with errors - idea

Consider the equation system:

$$\begin{aligned} \boldsymbol{s} * \boldsymbol{a}_1 &\approx b_1 \ (mod\ \ p) \\ \boldsymbol{s} * \boldsymbol{a}_2 &\approx b_2 \ (mod\ \ p) \\ &\ldots \\ \boldsymbol{s} * \boldsymbol{a}_m &\approx b_m \ (mod\ \ p) \end{aligned}$$

,where p prime, $\boldsymbol{s} \in \mathbb{Z}_p^n$, $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$. Find $\boldsymbol{s}$.
What does "approximate" mean?

Learning with errors - idea

Ex: $\approx$ means correct up to additive constant (± 1)

$$
\begin{aligned}
2 * s_1 + s_2 + s_3 &= 2 \pm 1 \; (mod\ 7) \\
s_1 + s_2 + 5 * s_3 &= 5 \pm 1 \; (mod\ 7) \\
s_1 + s_2 + s_3 &= 0 \pm 1 \; (mod\ 7)
\end{aligned}
$$

Learning with errors - idea

Ex: $\approx$ means correct up to additive constant (± 1)

$$
\begin{aligned}
2 * s_1 + s_2 + s_3 &= 2 \pm 1 \ (mod\ 7) \\
s_1 + s_2 + 5 * s_3 &= 5 \pm 1 \ (mod\ 7) \\
s_1 + s_2 + s_3 &= 0 \pm 1 \ (mod\ 7)
\end{aligned}
$$

We see that each of the equations have three possible RHS.
Some combinations of those RHS might not yield useable solutions. How to find a good combination?

Learning with errors - idea

Ex: $\approx$ means correct up to additive constant (± 1)

$$2 * s_1 + s_2 + s_3 = 2 \pm 1 \ (mod\ 7)$$
$$s_1 + s_2 + 5 * s_3 = 5 \pm 1 \ (mod\ 7)$$
$$s_1 + s_2 + s_3 = 0 \pm 1 \ (mod\ 7)$$

We see that each of the equations have three possible RHS.
Some combinations of those RHS might not yield useable solutions. How to find a good combination?
Moral: The addition of approximation makes the problem harder to solve.

Learning with errors - idea

Let us assign to each equation of our problem a random maximum accepted error e_i, only accept additive deviation and let us draw those e_i from some defined distribution χ.

Definition - LWE distribution

For a vector $\boldsymbol{s} \in \mathbb{Z}_p^n$, called the secret, and some probability distribution χ on $\mathbb{Z}_p$, the LWE distribution $A_{\boldsymbol{s},\chi}$ over $\mathbb{Z}_p^n \times \mathbb{Z}_p$ is sampled by:

- Uniformly randomly drawing sample $\boldsymbol{a}$ from $\mathbb{Z}_p^n$.
- Drawing random sample e from χ
- Outputting the pair $(\boldsymbol{a}, \boldsymbol{s} * \boldsymbol{a} + e \ mod \ p)$.

Learning with errors

Definition - LWE distribution

Take prime $p \in \mathbb{Z}$ and some $n \in \mathbb{Z}$. For a vector $\boldsymbol{s} \in \mathbb{Z}_p^n$, called the secret, and some probability distribution χ on $\mathbb{Z}_p$, the <u>LWE distribution</u> $A_{\boldsymbol{s},\chi}$ over $\mathbb{Z}_p^n \times \mathbb{Z}_p$ is sampled by:

- Uniformly randomly drawing sample $\boldsymbol{a}$ from $\mathbb{Z}_p^n$.
- Drawing random sample e from χ
- Outputting the pair $(\boldsymbol{a}, \boldsymbol{s} * \boldsymbol{a} + e \ mod \ p)$.

Definition - $LWE_{\boldsymbol{s},\chi,n,m}$ problem

Given m independent samples $(\boldsymbol{a}_i, b_i)$ drawn from $A_{\boldsymbol{s}\chi}$ using a uniformly random $\boldsymbol{s} \in \mathbb{Z}_p^n$, find $\boldsymbol{s}$.

Learning with errors

Definition - LWE distribution

Take prime $p \in \mathbb{Z}$ and some $n \in \mathbb{Z}$. For a vector $\boldsymbol{s} \in \mathbb{Z}_p^n$, called the secret, and some probability distribution χ on $\mathbb{Z}_p$, the LWE distribution $A_{\boldsymbol{s},\chi}$ over $\mathbb{Z}_p^n \times \mathbb{Z}_p$ is sampled by:

- Uniformly randomly drawing sample $\boldsymbol{a}$ from $\mathbb{Z}_p^n$.
- Drawing random sample e from χ
- Outputting the pair $(\boldsymbol{a}, \boldsymbol{s} * \boldsymbol{a} + e \ mod \ p)$.

Definition - $LWE_{\boldsymbol{s},\chi,n,m}$ problem

Given m independent samples $(\boldsymbol{a}_i, b_i)$ drawn from $A_{\boldsymbol{s}\chi}$ using a uniformly random $\boldsymbol{s} \in \mathbb{Z}_p^n$, find $\boldsymbol{s}$.

Note: This is the "search" version of LWE.

Learning with errors

Definition - $LWE_{\mathbf{s},\chi,n,m}$ problem

Given m independent samples $(\mathbf{a}_i, b_i)$ drawn from $A_{\mathbf{s}\chi}$ using a uniformly random $\mathbf{s} \in \mathbb{Z}_p^n$, find $\mathbf{s}$.

$$\begin{aligned}
\mathbf{s} * \mathbf{a}_1 &= b_1 + e_1 \ (mod\ p)\\
\mathbf{s} * \mathbf{a}_2 &= b_2 + e_2 \ (mod\ p)\\
\mathbf{s} * \mathbf{a}_3 &= b_3 + e_3 \ (mod\ p)\\
&\dots\\
\mathbf{s} * \mathbf{a}_m &= b_m + e_m \ (mod\ p)
\end{aligned}$$

Contents

Overview of hardness

We need to confirm that LWE is hard.

Overview of hardness

We need to confirm that LWE is hard.
As proving that a problem is hard is generally very difficult, a less labourful approach might be to prove that the problem at hand is reducable to a known hard problem in a reasonable amount of time.

Overview of hardness

We need to confirm that LWE is hard.
As proving that a problem is hard is generally very difficult, a less labourful approach might be to prove that the problem at hand is reducable to a known hard problem in a reasonable amount of time.

Overview of hardness

Theorem (hardness of LWE) (Informal)

[a] Let n, p be integers and χ an error distribution so that certain criterias are met. If there exists an efficient algorithm that solves $LWE_{p,\chi}$ then there exists an efficient quantum algorithm that approximates the decision version of the shortest vector problem (GapSVP) in the worst case.

[a] [Regev 2009]

Overview of hardness

Theorem (hardness of LWE) (Informal)

[a] Let n, p be integers and χ an error distribution so that certain criterias are met. If there exists an efficient algorithm that solves $LWE_{p,\chi}$ then there exists an efficient quantum algorithm that approximates the decision version of the shortest vector problem (GapSVP) in the worst case.

[a] [Regev 2009]

Sketch of hardness proof:

$LWE \underset{classical\ reduction}{\Longrightarrow} BDD_\gamma$

$BDD_\gamma \underset{quantum\ reduction}{\Longrightarrow} GapSVP_\gamma$

BDD

Definition - (BDD_γ (Bounded Distance Decoding problem))

Given a basis B of an n-dimensional lattice L, some function γ and a target point $t \in \mathbb{R}^n$ with the guarantee that $dist(t, L) < d = \frac{\lambda_1(L)}{2\gamma(n)}$, find the unique lattice vector $v \in L$ such that $||t - v|| < d$.

GapSVP

Definition - ($GapSVP_\gamma$ (Gap Shortest Vector Problem))

Given a basis B of an n-dimensional lattice L, a function γ, a number $d > 0$ and the guarantee that either $\lambda_1(L) \leq d$ or $\lambda_1(L) > \gamma(n) * d$, determine which is the case.

GapSVP

Definition - ($GapSVP_\gamma$ (Gap Shortest Vector Problem))

Given a basis B of an n-dimensional lattice L, a function γ, a number $d > 0$ and the guarantee that either $\lambda_1(L) \leq d$ or $\lambda_1(L) > \gamma(n) * d$, determine which is the case.

Note: is it known that *GapSVP* is hard.

Contents

Why do we need to switch to a ring?

Cryptosystems based on LWE tend to require about n samples from LWE-dist for the public key, $\implies$ key lengths $\approx \mathcal{O}(n^2)$.

$$
\begin{aligned}
\boldsymbol{s} * \boldsymbol{a}_1 &= b_1 + e_1 \ (mod\ p) \\
\boldsymbol{s} * \boldsymbol{a}_2 &= b_2 + e_2 \ (mod\ p) \\
\boldsymbol{s} * \boldsymbol{a}_3 &= b_3 + e_3 \ (mod\ p) \\
&\ldots \\
\boldsymbol{s} * \boldsymbol{a}_m &= b_m + e_m \ (mod\ p)
\end{aligned}
$$

Why do we need to switch to a ring?

Cryptosystems based on LWE tend to require about n samples from the LWE-distribution for the public key, $\implies$ key lengths $\approx \mathcal{O}(n^2)$.

$$\begin{aligned} \boldsymbol{s} * \boldsymbol{a}_1 &= b_1 + e_1 \ (mod \ \ p) \\ \boldsymbol{s} * \boldsymbol{a}_2 &= b_2 + e_2 \ (mod \ \ p) \\ \boldsymbol{s} * \boldsymbol{a}_3 &= b_3 + e_3 \ (mod \ \ p) \\ &\dots \\ \boldsymbol{s} * \boldsymbol{a}_m &= b_m + e_m \ (mod \ \ p) \end{aligned}$$

(Compare to needing n equations to solve a linear system with Gauss elimination)

Translation to Ring LWE

What if the public key was shorter but had some structure so that the same amount of samples could be constructed from it?

Translation to Ring LWE

What if the public key was shorter but had some structure so that the same amount of samples could be constructed from it?

$\implies$ Use a ring!

RLWE

definition - (RLWE distribution)

For $\boldsymbol{s} \in R_q$, called the secret, the <u>RLWE distribution</u> $A_{\boldsymbol{s},\chi}$ is sampled by choosing $a \in R_q$ uniformly random, choosing $\boldsymbol{e} \in R_q$ according to χ and outputting

$$(\boldsymbol{a}, \boldsymbol{s} * \boldsymbol{a} + \boldsymbol{e} \ mod q)$$

RLWE

definition - (RLWE distribution)

For $\boldsymbol{s} \in R_q$, called the secret, the RLWE distribution $A_{\boldsymbol{s},\chi}$ is sampled by choosing $a \in R_q$ uniformly random, choosing $\boldsymbol{e} \in R_q$ according to χ and outputting

$$(\boldsymbol{a}, \boldsymbol{s} * \boldsymbol{a} + \boldsymbol{e} \ mod q)$$

definition - (Search $RLWE_{n,q,\chi,m}$)

Given m independent samples from $A_{s,\chi}$, find s.

Contents

Applications

Using *LWE* as a basis for cryptographic schemes is thought to have two main benefits:

- Believed to be suitable for post-quantum cryptography.

Applications

Using *LWE* as a basis for cryptographic schemes is thought to have two main benefits:

- Believed to be suitable for post-quantum cryptography.
- Enables homomorphic encryption (HE).
  - A (potential) gamechanger when it comes to privacy.

Homomorphic encryption

- Idea: Encryption such that certain calculations can be made on the encrypted data <u>without</u> decrypting it.
- Example:

Seminar_talk_matcryp/homomorph.png

HE example - Approximative Eigenvector Method

Consider: If μ_1, μ_2 are the eigenvalues w.r.t $\boldsymbol{s}$ of C_1, C_2 respectively with the same eigenvector $\boldsymbol{s}$. Then we have that the eigenvalue of $C_1 + C_2$ is $\mu_1 + \mu_2$ w.r.t $\boldsymbol{s}$ and that the eigenvalue of $C_1 * C_2$ is $\mu_1\mu_2$ w.r.t $\boldsymbol{s}$.

HE example - Approximative Eigenvector Method

Note: If μ_1, μ_2 are the eigenvalues w.r.t $\boldsymbol{s}$ of C_1, C_2 respectively with the same eigenvector $\boldsymbol{s}$. Then we have that the eigenvalue of $C_1 + C_2$ is $\mu_1 + \mu_2$ w.r.t $\boldsymbol{s}$ and that the eigenvalue of $C_1 * C_2$ is $\mu_1\mu_2$ w.r.t $\boldsymbol{s}$.

Idea:Let μ be the message, $\boldsymbol{s}$ the secret key and C the ciphertext. Such construction seems to be homomorphic under addition and multiplication.

HE example - Approximative Eigenvector Method

Key generation:

- Draw m samples of length from $A_{s,\chi}$.

$$\boldsymbol{b} = B * \boldsymbol{t} + \boldsymbol{e}$$

$,B \in \mathbb{Z}_q^{mxn}$, $\boldsymbol{e} \in \chi^m$

HE example - Approximative Eigenvector Method

Key generation:

- Draw m samples of length from $A_{s,\chi}$.

$$\boldsymbol{b} = B * \boldsymbol{t} + \boldsymbol{e}$$

$,B \in \mathbb{Z}_q^{mxn}$, $\boldsymbol{e} \in \chi^m$

- Output $\boldsymbol{s} = (1, -t_1, ..., -t_m)$ as the secret key.

HE example - Approximative Eigenvector Method

<u>Key generation:</u>

- Draw m samples of length from $A_{s,\chi}$.

$$\boldsymbol{b} = B * \boldsymbol{t} + \boldsymbol{e}$$

  $, B \in \mathbb{Z}_q^{mxn}$, $\boldsymbol{e} \in \chi^m$

- Output $\boldsymbol{s} = (1, -t_1, ..., -t_m)$ as the secret key.
- Output $A = [\boldsymbol{b} \; B]$ as the public key

HE example - Approximative Eigenvector Method

Key generation:

- Draw m samples of length from $A_{s,\chi}$.

$$\boldsymbol{b} = B * \boldsymbol{t} + \boldsymbol{e}$$

$,B \in \mathbb{Z}_q^{mxn}$, $\boldsymbol{e} \in \chi^m$

- Output $\boldsymbol{s} = (1, -t_1, ..., -t_m)$ as the secret key.
- Output $A = [\boldsymbol{b}\ B]$ as the public key
- Note that $A * \boldsymbol{s} = \boldsymbol{e}$

HE example - Approximative Eigenvector Method

For these help operators, let $\boldsymbol{a}$ be an k-dimensional vector over $\mathbb{Z}_p$, take $l = \lfloor log_2(p) \rfloor + 1$ and $N = k * l$.

<u>$powersOf2$:</u>

- $powersOf2(\boldsymbol{a}) := (a_1, 2a_1, ..., 2^{l-1}, ..., a_k, ..., 2^{l-1})$

HE example - Approximative Eigenvector Method

For these help operators, let $\boldsymbol{a}$ be an k-dimensional vector over $\mathbb{Z}_p$, take $l = \lfloor log_2(p) \rfloor + 1$ and $N = k * l$.

<u>$powersOf2$:</u>

- $powersOf2(\boldsymbol{a}) := (a_1, 2a_1, ..., 2^{l-1}, ..., a_k, ..., 2^{l-1})$

<u>$bitDecomp$:</u>

- $bitDecomp(\boldsymbol{a}) := (a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, ..., a_{k,l-1})$
  ,with $a_{i,j}$ being the j-th bit of a_i's bit representation (LSB $\rightarrow$ MSB).

HE example - Approximative Eigenvector Method

For these help operators, let $\boldsymbol{a}$ be an k-dimensional vector over $\mathbb{Z}_p$, take $l = \lfloor log_2(p) \rfloor + 1$ and $N = k * l$.

$\underline{powersOf2:}$

- $powersOf2(\boldsymbol{a}) := (a_1, 2a_1, ..., 2^{l-1}, ..., a_k, ..., 2^{l-1})$

$\underline{bitDecomp:}$

- $bitDecomp(\boldsymbol{a}) := (a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, ..., a_{k,l-1})$
  ,with $a_{i,j}$ being the j-th bit of a_i's bit representation (LSB $\rightarrow$ MSB).
- $bitDecomp^{-1}(\boldsymbol{a}) := (\sum_{j=0}^{l-1} s^j * a_{1,j}, ..., \sum_{j=0}^{l-1} s^j * a_{k,j})$

HE example - Approximative Eigenvector Method

For these help operators, let $\boldsymbol{a}$ be an k-dimensional vector over $\mathbb{Z}_p$, take $l = \lfloor log_2(p) \rfloor + 1$ and $N = k * l$.

<u>$powersOf2$:</u>

- $powersOf2(\boldsymbol{a}) := (a_1, 2a_1, ..., 2^{l-1}, ..., a_k, ..., 2^{l-1})$

<u>$bitDecomp$:</u>

- $bitDecomp(\boldsymbol{a}) := (a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, ..., a_{k,l-1})$,with $a_{i,j}$ being the j-th bit of a_i's bit representation (LSB $\rightarrow$ MSB).
- $bitDecomp^{-1}(\boldsymbol{a}) := (\sum_{j=0}^{l-1} s^j * a_{1,j}, ..., \sum_{j=0}^{l-1} s^j * a_{k,j})$

Note: $bitDecomp^{-1}$ is defined even for non-binary $\boldsymbol{a}$

HE example - Approximative Eigenvector Method

For these operators, let $\boldsymbol{a}$ be an k-dimensional vector over $\mathbb{Z}_p$, take $l = \lfloor log_2(p) \rfloor$ and $N = k * l$.

<u>*powersOf 2* :</u>

- $powersOf2(\boldsymbol{a}) := (a_1, 2a_1, ..., 2^{l-1}, ..., a_k, ..., 2^{l-1})$

<u>*bitDecomp* :</u>

- $bitDecomp(\boldsymbol{a}) := (a_{1,0}, ..., a_{1,l-1}, ..., a_{k,0}, ..., a_{k,l-1})$
  ,with $a_{i,j}$ being the j-th bit of a_i's bit representation (LSB $\rightarrow$ MSB).
- $bitDecomp^{-1}(\boldsymbol{a}) := (\sum_{j=0}^{l-1} s^j * a_{1,j}, ..., \sum_{j=0}^{l-1} s^j * a_{k,j})$

<u>*Flatten* :</u>

- $Flatten(\boldsymbol{a}) := bitDecomp(bitDecomp^{-1}(\boldsymbol{a}))$

HE example - Approximative Eigenvector Method

Enc:

- Take message $\mu \in \mathbb{Z}_p$, define $\boldsymbol{v} = powersOf2(\boldsymbol{s})$ and generate random $R \in \mathbb{Z}_2^{Nxm}$.

HE example - Approximative Eigenvector Method

<u>*Enc*:</u>

- Take message $\mu \in \mathbb{Z}_p$, define $\boldsymbol{v} = powersOf2(\boldsymbol{s})$ and generate random $R \in \mathbb{Z}_2^{Nxm}$.
- $C := Flatten(\mu * I_n + bitDecomp(R * A)) \in \mathbb{Z}_p^{NxN}$, where I_n is the identity matrix of size nxn.

$$\begin{aligned} \implies C * \boldsymbol{v} &= \mu * \boldsymbol{v} + bitDecomp(R * A) * \boldsymbol{v} \\ &= \mu * \boldsymbol{v} + R * A * s = \mu * \boldsymbol{v} + R * \boldsymbol{e} \\ &= \mu * \boldsymbol{v} + small \end{aligned}$$

HE example - Approximative Eigenvector Method

Dec:

- Observe that the first l coefficients of $\boldsymbol{v}$ are $1, 2, ..., 2^{l-1}$.

HE example - Approximative Eigenvector Method

Dec:

- Observe that the first l coefficients of $\boldsymbol{v}$ are $1, 2, ..., 2^{l-1}$.
- Among these coefficients, let $v_i = 2^i$ be in $(p/4, p/2]$. Let C_i be the i-th row of C.

HE example - Approximative Eigenvector Method

Dec:

- Observe that the first l coefficients of $\boldsymbol{v}$ are $1, 2, ..., 2^{l-1}$.
- Among these coefficients, let $v_i = 2^i$ be in $(p/4, p/2]$. Let C_i be the i-th row of C.
- Compute $x_i \leftarrow C_i * \boldsymbol{v} = \mu * v_i + R_i * e_i$.

HE example - Approximative Eigenvector Method

Dec:

- Observe that the first l coefficients of $\boldsymbol{v}$ are $1, 2, ..., 2^{l-1}$.
- Among these coefficients, let $v_i = 2^i$ be in $(p/4, p/2]$. Let C_i be the i-th row of C.
- Compute $x_i \leftarrow C_i * \boldsymbol{v} = \mu * v_i + R_i * e_i$.
- Output $\mu' = \lfloor \frac{x_i}{v_i} \rceil$.

HE example - Approximative Eigenvector Method

Dec:

- Observe that the first l coefficients of $\boldsymbol{v}$ are $1, 2, ..., 2^{l-1}$.
- Among these coefficients, let $v_i = 2^i$ be in $(p/4, p/2]$. Let C_i be the i-th row of C.
- Compute $x_i \leftarrow C_i * \boldsymbol{v} = \mu * v_i + R_i * e_i$.
- Output $\mu' = \lfloor \frac{x_i}{v_i} \rceil$.

Note: Under certain assumptions that $\boldsymbol{e}$ is small, we can be sure that $\mu' = \mu$.

HE example - Approximative Eigenvector Method

Add:

- $Add(C_1, C_2) := Flatten(C_1 + C_2) = bitDecomp(bitDecomp^{-1}(C_1 + C_2))$

HE example - Approximative Eigenvector Method

Add:

- $Add(C_1, C_2) := Flatten(C_1 + C_2) = bitDecomp(bitDecomp^{-1}(C_1 + C_2))$

Mult:

- $Mult(C_1, C_2) := Flatten(C_1 * C_2) = bitDecomp(bitDecomp^{-1}(C_1 * C_2))$

HE example - Approximative Eigenvector Method

<u>*Add*</u>:

- $Add(C_1, C_2) := Flatten(C_1 + C_2) = bitDecomp(bitDecomp^{-1}(C_1 + C_2))$

<u>*Mult*</u>:

- $Mult(C_1, C_2) := Flatten(C_1 * C_2) = bitDecomp(bitDecomp^{-1}(C_1 * C_2))$
- Note that:

$$
\begin{aligned}
Mult(C_1, C_2) * \mathbf{v} &= C_1 * C_2 * \mathbf{v} = \\
&= C_1 * (\mu_2 * \mathbf{v} + \mathbf{e_2}) + \mu_2 * (\mu_1 * \mathbf{v} + \mathbf{e_1}) + C_1 * \mathbf{e_2} \\
&= \mu_1 * \mu_2 * \mathbf{v} + \mu_2 * \mathbf{e_1} + C_1 * \mathbf{e_2} \\
&= \mu_1 * \mu_2 * \mathbf{v} + small
\end{aligned}
$$

HE example- Approximative Eigenvector Method

Remarks:

- Now we have an encryption scheme and operators that (presumably) act homomorphically (Hurray!)

HE example- Approximative Eigenvector Method

Remarks:

- Now we have an encryption scheme and operators that (presumably) act homomorphically (Hurray!)
- Note that the decryption is dependent on that the error is somewhat small.

HE example- Approximative Eigenvector Method

Remarks:

- Now we have an encryption scheme and operators that (presumably) act homomorphically (Hurray!)
- Note that the decryption is dependent on that the error is somewhat small.
- Since the "final" error is increased after each use of an operator, the error distribution in the beginning needs to be dependent on the number of operations in the computation.

HE example- Approximative Eigenvector Method

Remarks:

- Now we have an encryption scheme and operators that (presumably) act homomorphically (Hurray!)
- Note that the decryption is dependent on that the error is somewhat small.
- Since the “final” error is increased after each use of an operator, the error distribution in the beginning needs to be dependent on the number of operations in the computation.
- Idea: all algorithms can be built with NAND gates.

HE example- Approximative Eigenvector Method

Remarks:

- This scheme is based on *LWE* and not *RLWE*. (Can be translated)

HE example- Approximative Eigenvector Method

Remarks:

- This scheme is based on *LWE* and not *RLWE*. (Can be translated)
- There exist more effective schemes based on *RLWE*, both with respect to speed and dependence on the number of operations.

References

- **Regev (2010)**, *The Learning with Errors Problem*
  Invited survey for 2010 IEEE 25th Annual Conference on Computational Complexity
  https://cims.nyu.edu/ regev/papers/lwesurvey.pdf
- **Gentry, Sahai, Waters (2013)**: *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based.*
  Annual Cryptology Conference CRYPTO 2013: Advances in Cryptology – CRYPTO 2013 pp 75-92
  https://eprint.iacr.org/2013/340.pdf