The CKKS Cryptosystem
Part 2: Algorithms

Introduction to Homomorphic Cryptosystems - Lecture 4

Notations and Abbreviations

In the following, elements $\in \mathrm{R}_{n,q_L}$ are treated as vectors ($\mathbb{Z}^n$). The coefficients of the polynomial are the vector elements

$$q_L \in \mathbb{N},\ n \in \{2^k \mid k \in \mathbb{N}\},$$
$$h, P \in \mathbb{Z},\ \sigma \in \mathbb{R}^+$$

$\mathbb{Z}_p :=$ $\mathbb{Z}$ modulo p

$\mathrm{R}_{n,q_L} := \mathbb{Z}_{q_L}[X]/(X^n + 1)$

$A * z$

$:=$ Matrixmultiplication of Matrix A with vector z

$$\mathbb{H}_n := \left\{\left(v_1, v_2, \dots, v_{\frac{n}{2}}, \overline{v_{\frac{n}{2}}}, \dots, \overline{v_2}, \overline{v_1}\right) \in \mathbb{C}^n\right\}$$

$\pi: \mathbb{H}_n \to \mathbb{C}^{\frac{n}{2}}$

$\langle v, w\rangle :=$ inner product of the vectors $v, w \in \mathbb{C}^n$

$$\langle v, w\rangle := \sum_{i=1}^{n} v_i \overline{w_i}$$

$v \odot w, v \oplus w :=$ coordinate wise $\cdot, +$

$v^{\perp} :=$ transpose of v

$\xi_n := e^{\frac{2\pi i}{n}}$

$V\left((x_1, \dots, x_n)\right) :=$ Vandermonde Matrix

$V_n := V\left((\xi_{2n}^1, \xi_{2n}^3, \dots, \xi_{2n}^{2n-1})\right)$

$V_n[i] := i$th column of V_n

round $:=$ random rounding

CKKS Algorithms Overview

Encoding

$$\text{Ecd}: \mathbb{C}^{\frac{n}{2}} \rightarrow \mathbb{Z}[X]/(X^n+1)$$

Decoding

$$\text{Dcd}: \mathbb{Z}[X]/(X^n+1) \rightarrow \mathbb{C}^{\frac{n}{2}}$$

plaintext message

encoded message

key related

ciphertext

R^2_{n,q_L} : Vector with two elements, which are polynomials (and therefore also vectors):

$$\begin{pmatrix} (a_1, a_2, a_3, \dots)^{\perp} \\ (b_1, b_2, b_3, \dots)^{\perp} \end{pmatrix}$$

public key

Encryption

$$\text{Enc}: \mathrm{R}^2_{n,q_L} \times \mathbb{Z}[X]/(X^n+1) \rightarrow \mathrm{R}^2_{n,q_L}$$

Decryption

$$\text{Dec}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}_{n,q_L}$$

secret key

Addition

$$\text{Add}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

Multiplication

$$\text{Mul}: \mathrm{R}^2_{n,P*q_L} \times \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

ENCODING

Encoding

$$\mathrm{Ecd}: \mathbb{C}^{\frac{n}{2}} \to \mathbb{Z}[X]/(X^n + 1)$$

Goal

We want to encode a vector of (complex-)numbers. The complex numbers represent the plaintext message, and the result of the encoding is a polynomial which will be used in the encryption algorithm.

Why polynomials?

Working with polynomials instead of plain vectors provide a good trade-off between security and efficiency.

Encoding

$$\mathrm{Ecd}: \mathbb{C}^{\frac{n}{2}} \to \mathbb{Z}[X]/(X^n+1)$$

Make v antisymmetrical

Steps

Input: $v \in \mathbb{C}^{\frac{n}{2}}$

1. $z \coloneqq \pi^{-1}(v) \in \mathbb{H}_n$
2. $p_i \coloneqq \frac{\langle z, V_n[i] \rangle}{\langle V_n[i], V_n[i] \rangle} \in \mathbb{R}$ for every $i \in \{1, \dots, n\}$
3. $m_i \coloneqq \mathrm{round}(p_i)$ for every $i \in \{1, \dots, n\}$
4. $\mathrm{Ecd}(v) \coloneqq m$

$m = (m_1, \dots, m_n)^{\perp}$

m are now the coefficients of a polynomial $\in \mathbb{Z}[X]/(X^n+1)$

Mathematical Explanation

m is a polynomial, which when evaluated for the value $\xi_{2n}^{1+2(i-1)}$ gives the i-th value from v.

DECODING

Decoding

$$\mathrm{Dcd}\colon \mathbb{Z}[X]/(X^n+1) \to \mathbb{C}^{\frac{n}{2}}$$

Goal

We want to decode a polynomial. The polynomial represents an encoded message, and the result of the decoding is a vector of (complex-)numbers. These numbers are the plaintext message and may be the result of computations on the encrypted data.

Decoding

$$\mathrm{Dcd}: \mathbb{Z}[X]/(X^n + 1) \to \mathbb{C}^{\frac{n}{2}}$$

Steps

Input: $m \in \mathbb{Z}[X]/(X^n + 1)$

1. $h_j := m(\xi_{2n}^{1+2(i-1)})$ for every $i \in \{1, \dots, n\}$
2. $v := \pi(h)$
3. $\mathrm{Dcd}(r) := v$

$h = (h_1, \dots, h_n)^{\perp} \in \mathbb{H}_n$

v are now the coefficients of our original vector $\in \mathbb{C}^{\frac{n}{2}}$

Mathematical Explanation

m is a polynomial, which when evaluated for the value $\xi_{2n}^{1+2(i-1)}$ gives the i-th value from v.

We now evaluate the polynomial to receive the original values.

Random Distributions

For the following algorithms, we define the following random distributions

$DG(\sigma^2)$

samples vectors $\in \mathbb{Z}^n$.
The coefficients are independently drawn from the discrete gaussian distribution of variance σ^2.

Examples

$n = 4, \sigma^2 = 16$

$$\begin{pmatrix} -2 \\ 4 \\ 0 \\ -4 \end{pmatrix}, \begin{pmatrix} -2 \\ 0 \\ 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 3 \\ -1 \\ 2 \end{pmatrix}$$

ZO

samples vectors $\in \{-1,1\}^n$.
The coefficients are independently drawn from $\{-1,1\}$ with probability $\frac{1}{2}$.

Examples

$n = 4$

$$\begin{pmatrix} -1 \\ -1 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$HWT(h)$

samples vectors $\in \{-1,0,1\}^n$.
The coefficients are drawn so that the hamming weight of the vector is exactly h.

Examples

$n = 4, h = 3$

$$\begin{pmatrix} -1 \\ -1 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ -1 \end{pmatrix}$$

KEY GENERATION

Key Generation

Goal

As CKKS is an asymmetric encryption scheme, we want to generate a public key (pk) and a secret key (sk).
Furthermore, a third 'evaluation key' (evk) is needed for the multiplication. We will understand its purpose later.

Steps

1. Draw randomly:
   $s \leftarrow HWT(h)$
   $e, e' \leftarrow DG(\sigma^2)$
   $a \leftarrow \mathrm{R}_{n,q_L}$ (uniformly distributed)
   $a' \leftarrow \mathrm{R}_{n,Pq_L}$ (uniformly distributed)

2. Generate the keys:
   $pk = (b, a) \coloneqq (-a \odot s \oplus e, a) \in \mathrm{R}^2_{n,q_L}$
   $sk = (1, s) \in \mathbb{Z} \times \{-1,0,1\}^n$
   $evk = (b', a')$
   $\coloneqq \left(\left((-a' \odot s) \oplus e' \oplus (P * s \odot s)\right), a'\right) \in \mathrm{R}^2_{n,P*q_L}$

The calculations for pk are done $\bmod q_L$.

For evk we use $\bmod Pq_L$

ENCRYPTION & DECRYPTION

Definition

Goal

We want to encrypt (and decrypt) the encoded message (the ciphertext) using our public key (secret key).

Furthermore, the scheme should rely on the hardness of the LWE problem to guarantee a certain level of security. (More on the security later!)

Encryption

$\text{Enc}: \mathrm{R}^2_{n,q_L} \times \mathbb{Z}[X]/(X^n+1) \to \mathrm{R}^2_{n,q_L}$

$pk = (b, a)$ m $x = (x_1, x_2)$

Definition

Draw randomly: $v \leftarrow ZO$ and $e_0, e_1 \leftarrow DG(\sigma^2)$

$\text{Enc}_{pk}(m) \coloneqq (v \odot b \oplus m \oplus e_0, v \odot a \oplus e_1) \bmod q_L = x$

Decryption

$\text{Dec}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \to \mathrm{R}_{n,q_L}$

$sk = (1, s)$ $x = (x_1, x_2)$ m

Definition

$\text{Dec}_{sk}(x) \coloneqq x_1 \oplus x_2 \odot s \bmod q_L = m$

Correctness and Error

$$pk = (b, a) \coloneqq (-a \odot s \oplus e, a)$$
$$sk = (1, s)$$
$$\mathrm{Enc}_{pk}(m) \coloneqq (v \odot b \oplus m \oplus e_0, v \odot a \oplus e_1) \bmod q_L$$
$$\mathrm{Dec}_{sk}(x) \coloneqq x_1 \oplus x_2 \odot s \bmod q_L$$

Applying decryption to an encrypted message does indeed result in the original message:

$$\mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}(m)\right) = \mathrm{Dec}_{sk}\left((x_1, x_2)\right) = x_1 \oplus x_2 \odot s$$
$$= (v \odot b \oplus m \oplus e_0) \oplus (v \odot a \oplus e_1) \odot s$$
$$= v \odot (-a \odot s \oplus e) \oplus m \oplus e_0 \oplus (v \odot a \oplus e_1) \odot s$$
$$= (-v \odot a \odot s) \oplus (v \odot e) \oplus m \oplus e_0 \oplus (v \odot a \odot s) \oplus (e_1 \odot s)$$
$$= m \oplus (v \odot e) \oplus e_0 \oplus (e_1 \odot s)$$
$$= m \oplus k$$
$$\approx m$$

$$k = (v \odot e) \oplus e_0 \oplus (e_1 \odot s) \ll m$$

But we can only recover the result, if the error is small.

Security

We can see the LWE problem in the CKKS encryption:

$$pk = (b, a) \coloneqq (-a \odot s \oplus e, a)$$
$$sk = (1, s)$$
$$\mathrm{Enc}_{pk}(m) \coloneqq (v \odot b \oplus m \oplus e_0, v \odot a \oplus e_1) \bmod q_L$$
$$\mathrm{Dec}_{sk}(x) \coloneqq x_1 \oplus x_2 \odot s \bmod q_L$$

The public key masks the secret key by adding an error…

… and so does the encryption.
The message is hidden in the first part of the ciphertext, and an attacker cannot solve for m, as he only gets errored information.

The difference to our first definition of LWE is that we now work with polynomials.
We call this version of LWE Ring-Learning-With-Errors (RLWE).

ADDITION

Definition and Correctness

Goal

We want to add two ciphertexts while retaining the homomorphic property.

Definition

$$\text{Add}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \to \mathrm{R}^2_{n,q_L}$$

$x = (x_1, x_2)$ $y = (y_1, y_2)$ $z = (z_1, z_2)$

$$\text{Add}(x, y) \coloneqq x \oplus y = (x_1 + y_1, x_2 + y_2) \bmod q_L$$

Correctness

$$\text{Dec}_{sk}\left(\text{Add}\left(\text{Enc}_{pk}(m), \text{Enc}_{pk}(m')\right)\right) = \text{Dec}_{sk}\left((x_1, x_2) \oplus (y_1, y_2)\right) =$$
$$= (x_1 \oplus y_1) \oplus (x_2 \oplus y_2) \odot s = (x_1 \oplus x_2 \odot s) \oplus (y_1 \oplus y_2 \odot s)$$
$$= (m \oplus k) \oplus (m' \oplus k')$$
$$= m \oplus m' \oplus k \oplus k'$$
$$\approx m \oplus m'$$

$k = (v \odot e) \oplus e_0 \oplus (e_1 \odot s) \ll m$

CKKS Algorithms Overview – Decoding Problem

Encoding

$$\text{Ecd}: \mathbb{C}^{\frac{n}{2}} \rightarrow \mathbb{Z}[X]/(X^n+1)$$

Decoding

$$\text{Dcd}: \mathbb{Z}[X]/(X^n+1) \rightarrow \mathbb{C}^{\frac{n}{2}}$$

Encryption

$$\text{Enc}: \mathrm{R}^2_{n,q_L} \times \mathbb{Z}[X]/(X^n+1) \rightarrow \mathrm{R}^2_{n,q_L}$$

Decryption

$$\text{Dec}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}_{n,q_L}$$

Addition

$$\text{Add}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

Multiplication

$$\text{Mul}: \mathrm{R}^2_{n,Pq_L} \times \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

CKKS Algorithms Overview – Decoding Problem

Encoding

$$\text{Ecd}: \mathbb{C}^{\frac{n}{2}} \rightarrow \mathbb{Z}[X]/(X^n+1)$$

Decoding

$$\text{Dcd}: \mathbb{Z}[X]/(X^n+1) \rightarrow \mathbb{C}^{\frac{n}{2}}$$

The input for the decoder is a polynomial which coefficients are $\in \mathbb{Z}$, but the decryption yields a polynomial with coefficients $\in \mathbb{Z}_{q_L}$.
So, if we come close to our modulus during calculations, the decoder might decode our value to an unexpected value.

Encryption

$$\text{Enc}: \mathrm{R}^2_{n,q_L} \times \mathbb{Z}[X]/(X^n+1) \rightarrow \mathrm{R}^2_{n,q_L}$$

Decryption

$$\text{Dec}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}_{n,q_L}$$

Addition

$$\text{Add}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

Multiplication

$$\text{Mul}: \mathrm{R}^2_{n,Pq_L} \times \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

MULTIPLICATION

Definition

Goal

We want to multiply two ciphertexts while retaining the homomorphic property.

Definition

$$\text{Mul}: \mathrm{R}^2_{n,P*q_L} \times \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^2_{n,q_L}$$

$evk = (b', a')$ $x = (x_1, x_2)$ $y = (y_1, y_2)$ $z = (z_1, z_2)$

$$\text{Mul}_{evk}(x, y) \coloneqq (x_1 \odot y_1, x_1 \odot y_2 \oplus y_1 \odot x_2) \oplus (P^{-1} \odot x_2 \odot y_2 \odot b', P^{-1} \odot x_2 \odot y_2 \odot a') \mod q_L$$

How do you get this definition and is this correct?

The naïve Approach

Goal

We want to multiply two ciphertexts while retaining the homomorphic property.

Definition

$$\mathrm{Mul}_{naive}: \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \to \mathrm{R}^2_{n,q_L}$$

$x = (x_1, x_2)$ $y = (y_1, y_2)$ $z = (z_1, z_2)$

$$\mathrm{Mul}_{naive}(x, y) \coloneqq x \odot y \mod q_L$$

Correctness?

$$\mathrm{Dec}_{sk}\left(\mathrm{Mul}_{naive}\left(\mathrm{Enc}_{pk}(m), \mathrm{Enc}_{pk}(m')\right)\right) = \cdots =$$

$$= mm' + a^2 s^2 vv' + a^2 svv' - amsv' - am'sv + [\ldots]$$

This solution does not yield a result from which we can recover the result of the operation.
There are other terms in this expression which also include the messages or parts of the keys.

A New Approach

Start from the back
We would like to achieve this:

$$\mathrm{Dec}_{sk}\left(\mathrm{Mul}\left(\mathrm{Enc}_{pk}(m), \mathrm{Enc}_{pk}(m')\right)\right) = \mathrm{Dec}_{sk}(x) \odot \mathrm{Dec}_{sk}(y)$$

$\mathrm{Enc}_{pk}(m) = x = (x_1, x_2)$
$\mathrm{Enc}_{pk}(m') = y = (y_1, y_2)$

Now evaluate the right side:

$$\mathrm{Dec}_{sk}(x) \odot \mathrm{Dec}_{sk}(y) = (x_1 \oplus x_2 \odot s) \odot (y_1 \oplus y_2 \odot s)$$
$$= (x_1 \odot y_1) \oplus \big((x_1 \odot y_2 \oplus y_1 \odot x_2) \odot s\big) \oplus (x_2 \odot y_2 \odot s^2)$$
$$= d_1 \oplus d_2 \odot s \oplus d_3 \odot s^2$$

$d_1 = (x_1 \odot y_1)$
$d_2 = (x_1 \odot y_2 \oplus y_1 \odot x_2)$
$d_3 = (x_2 \odot y_2)$

We now define:

$$\mathrm{Mul}'(x, y) := z' = (d_1, d_2, d_3)$$
$$\mathrm{DecMul}_{sk}(z') = d_1 \oplus d_2 \odot s \oplus d_3 \odot s^2$$

This works, but the problem is: **the size of the ciphertext grew!**

$$\mathrm{Mul}' : \mathrm{R}^2_{n,q_L} \times \mathrm{R}^2_{n,q_L} \rightarrow \mathrm{R}^3_{n,q_L}$$

We need a way to eliminate one of the three polynomials

Relinearization

$$\mathrm{Mul}'(x,y) \coloneqq z' = (d_1, d_2, d_3)$$
$$d_1 = (x_1 \odot y_1)$$
$$d_2 = (x_1 \odot y_2 \oplus y_1 \odot x_2)$$
$$d_3 = (x_2 \odot y_2)$$

Goal

We need an operation that converts our ciphertext z' to the correct shape (*relinearizing* it). We search for:

$$\mathrm{Relin}(z') = (d_1', d_2') \text{ , such that}$$
$$\mathrm{Dec}_{sk}\big((d_1', d_2')\big) = d_1' \oplus d_2' \odot s = d_1 \oplus d_2 \odot s \oplus d_3 \odot s^2$$
$$= \mathrm{Dec}_{sk}(x) \odot \mathrm{Dec}_{sk}(y)$$

This would allow us to perform relinearization after every multiplication and we would preserve the shape of the ciphertext.

Idea

$$\mathrm{Relin}(z') = (d_1', d_2') = (d_1, d_2) \oplus k$$
$$\mathrm{Dec}_{sk}(k) = d_3 \odot s^2$$

$k = (k_1, k_2) \in \mathrm{R}_{n,q_L}^2$

But what to choose for k ?

With this, our requirements for Relin are fulfilled:

$$\mathrm{Dec}_{sk}\big((d_1', d_2')\big) = (d_1 \oplus k_1) \oplus (d_2 \oplus k_2) \odot s$$
$$= (d_1 \oplus d_2 \odot s) \oplus (k_1 \oplus k_2 \odot s)$$
$$= (d_1 \oplus d_2 \odot s) \oplus \mathrm{Dec}_{sk}(k)$$
$$= d_1 \oplus d_2 \odot s \oplus d_3 \odot s^2$$

Relinearization

Goal

We need to choose $k \in \mathrm{R}^2_{n,q_L}$, such that $\mathrm{Dec}_{sk}(k) = d_3 \odot s^2$

Idea

k must include some information about the secret s. So we provide an evaluation key, which can be used to compute k during multiplication.

Evaluation Key

First try:

$$evk' := \left((-a' \odot s) \oplus e' \oplus s^2, a'\right) \in \mathrm{R}^2_{n,q_L}$$

Why?

$$\mathrm{Dec}_{sk}(evk) \approx s^2$$

We try to use this and define $k' := d_3 * evk' = \left(d_3 \odot \left((-a' \odot s) \oplus e' \oplus s^2\right), d_3 \odot a'\right)$, so we have:

$\mathrm{Dec}_{sk}(k') = \cdots = d_3 \odot s^2 \oplus d_3 \odot e'$

d_3 is too big, so we cannot omit $d_3 \odot e'$ in this case!

Relinearization

Almost there

We need to modify the evaluation key

$$evk' = \big((-a' \odot s) \oplus e' \oplus s^2, a'\big) \in \mathrm{R}^2_{n,q_L}$$

a little bit.

We introduce a large integer P and calculate the key $\bmod P * q_L$:

$$evk \coloneqq \big((-a' \odot s) \oplus e' \oplus P * s^2, a'\big) \in \mathrm{R}^2_{n,P*q_L}$$

This allows us to reduce the noise included by the multiplication with d_3:

$$k \coloneqq P^{-1} * d_3 * evk = P^{-1}\big(d_3 \odot \big((-a' \odot s) \oplus e' \oplus P * s^2\big), d_3 \odot a'\big)$$

We can decrypt k analogue to k' from the previous slide an see the effect:

$$\mathrm{Dec}_{sk}(k) = \cdots = P^{-1}\big((d_3 \odot P * s^2) \oplus d_3 \odot e'\big) = d_3 \odot s^2 \oplus P^{-1} * d_3 \odot e'$$

As P is a large integer, we can omit $P^{-1} * d_3 \odot e'$ this time and we *finally* reached our goal:

$$\mathrm{Dec}_{sk}(k) \approx d_3 \odot s^2$$

We saw this definition already during key generation

Thist is done $\bmod q_L$

Multiplication – Bringing it all together

As we now have k, we can properly define Relin

$$\mathrm{Relin}_{evk}(z') \coloneqq (d_1, d_2) \oplus P^{-1}\left(d_3 \odot \left((-a' \odot s) \oplus e' \oplus P * s^2\right), d_3 \odot a'\right)$$

Reminder:
$\mathrm{Relin}(z') = (d_1', d_2') = (d_1, d_2) \oplus k$
$\mathrm{Dec}_{sk}(k) = d_3 \odot s^2$

And we can see, that our original definition of the multiplication came from

$$\mathrm{Mul}_{evk}(x, y) = \mathrm{Relin}_{evk}\left(\mathrm{Mul}'(x, y)\right)$$
$$= (x_1 \odot y_1, x_1 \odot y_2 \oplus y_1 \odot x_2) \oplus (P^{-1} \odot x_2 \odot y_2 \odot b', P^{-1} \odot x_2 \odot y_2 \odot a')$$

$\mathrm{Mul}'(x, y) \coloneqq z' = (d_1, d_2, d_3)$
$d_1 = (x_1 \odot y_1)$
$d_2 = (x_1 \odot y_2 \oplus y_1 \odot x_2)$
$d_3 = (x_2 \odot y_2)$

$$evk = (b', a')$$
$$\coloneqq \left(\left((-a' \odot s) \oplus e' \oplus (P * s \odot s)\right), a'\right) \in \mathrm{R}^2_{n, P*q_L}$$

Correctness

As we started the journey with $\mathrm{Dec}_{sk}\left(\mathrm{Mul}\left(\mathrm{Enc}_{pk}(m), \mathrm{Enc}_{pk}(m')\right)\right) = \mathrm{Dec}_{sk}(x) \odot \mathrm{Dec}_{sk}(y)$, the definition is obviously correct.

Summary – What did we learn today?

CKKS Algorithms

Encoding, Decoding

Key Generation

Encryption, Decryption

Addition, Multiplication

CKKS Pitfalls

There are several things to consider when using the cryptosystem

Every algorithm adds an error

Performing calculations can break the boundaries.