# CKKS Error Estimation and Parameters

## 1 Preliminaries

Let $\chi$ is a discrete Gaussian, usually of standard deviation $\sigma = 3.2$. We denote by $\mathcal{ZO}(\rho)$ the distribution where 0 is sampled with probability $\rho$, and $\pm 1$ are sampled with probability $\rho/2$. We denote the secret key distribution by $S$. This is usually the unifrom ternary distribution.

We use the notation $R_q$ to denote the ring $\mathbb{Z}[x]/(\Phi_N, q)$, where $\phi_N = x^N + 1$. $[\cdot]_q$ denotes modular reduction $\mod q$ (usually centered around 0).

## 2 The CKKS scheme [1]

### 2.1 Decrypting a fresh ciphertext

Let `ct` be a fresh ciphertext encrypted under the public key `pk`, where we have $\texttt{pk} = ([-as + e]_q, a)$. Then, decrypting `ct` yields

$$\begin{aligned}\texttt{Decrypt}(\texttt{ct}, \texttt{sk}) &= c_0 + sc_1 \pmod q \\ &= m + p_0 v + e_1 + svp_1 + se_2 \\ &= m + ve + e_1 + se_2.\end{aligned}$$

Recall $e, e_0, e_1 \leftarrow \chi$. The ephemeral key $v$ here is drawn from the same distribution as the secret key $S$, but sometimes it can be sampled from a slightly different distribution. This can for example be the distribution $\mathcal{ZO}(\rho)$.

### 2.2 Decrypting a multiplication

Here, I will skip the `Rescale` step. This is because the formulas are simpler without it, and the last `Rescale` operation is (perhaps unsurprisingly) simply a rescaling of the noise. Note that I am also keeping the modular reductions implicit. This also simplifies the equations.

So, let $\texttt{ct} = (c_0, c_1)$ and $\texttt{ct}' = (c_0', c_1')$ be two ciphertexts such that they decrypt as follows.

$$\begin{aligned}c_0 + sc_1 &= m + e \\ c_0' + sc_1' &= m' + e'.\end{aligned}$$

Then, the output of `Pre-Mult` is

$$(d_0, d_1, d_2) = (c_0 c_0', c_0 c_1' + c_0' c_1, c_1 c_1').$$

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = (1, s)$.

$\texttt{PublicKeyGen}(\texttt{sk})$: For $\texttt{sk} = (1, s)$, sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\texttt{pk} = ([-as + e]_q, a)$.

$\texttt{EvaluationKeyGen}(\texttt{sk}, w)$: Set $s = \texttt{sk}$. Sample $a' \leftarrow R_Q$, $(Q = Pq)$ uniformly at random and $e' \leftarrow \chi$. Output $\texttt{evk} = \left([-a's + e' + Ps^2]_Q, a'\right)$.

$\texttt{Encrypt}(\texttt{pk}, m)$: For the message $m \in R$. Let $\texttt{pk} = (p_0, p_1)$, sample $v \leftarrow S$ and $e_1, e_2 \leftarrow \chi$. Output $\texttt{ct} = ([m + p_0 v + e_1]_q, [p_1 v + e_2]_q)$.

$\texttt{Decrypt}(\texttt{sk}, \texttt{ct})$: Let $\texttt{ct} = (c_0, c_1)$. Output $m' = [c_0 + c_1 s]_q$.

$\texttt{Add}(\texttt{ct}_0, \texttt{ct}_1)$: Output $\texttt{ct} = ([\texttt{ct}_0[0] + \texttt{ct}_1[0]]_q, [\texttt{ct}_0[1] + \texttt{ct}_1[1]]_q)$.

$\texttt{Pre-Multiply}(\texttt{ct}_0, \texttt{ct}_1)$: Set

$$\begin{aligned} d_0 &= [\texttt{ct}_0[0]\texttt{ct}_1[0]]_q \\ d_1 &= [\texttt{ct}_0[0]\texttt{ct}_1[1] + \texttt{ct}_0[1]\texttt{ct}_1[0]]_q \\ d_2 &= [\texttt{ct}_0[1]\texttt{ct}_1[1]]_q \end{aligned}$$

Output $\texttt{ct} = (d_0, d_1, d_2)$.

$\texttt{Relinearize}(\texttt{ct}, \texttt{evk})$: Let $\texttt{ct}[0] = d_0$, $\texttt{ct}[1] = d_1$ and $\texttt{ct}[2] = d_2$. Let $\texttt{evk}[0] = -a's + e' + Ps^2$ and $\texttt{evk}[1] = a'$. Set

$$\begin{aligned} c_0' &= [d_0 + \lfloor P^{-1} \cdot d_2 \cdot (-a's + e' + Ps^2) \rceil]_q \\ c_1' &= [d_1 + \lfloor P^{-1} \cdot d_2 \cdot a' \rceil]_q \end{aligned}$$

Output $\texttt{ct}' = (c_0', c_1')$.

$\texttt{Rescale}(\texttt{ct}, q')$ : Let $\texttt{ct} = (c_0, c_1)$. Set $c_0' = \left[\left\lfloor \frac{q'}{q} c_0 \right\rceil\right]_{q'}$ and $c_1' = \left[\left\lfloor \frac{q'}{q} c_1 \right\rceil\right]_{q'}$. Output $\texttt{ct} = (c_0', c_1')$.

Note that this decrypts as

$$\begin{aligned} d_0 + sd_1 + s^2 d_2 &= (c_0 + c_1 s)(c_0' + sc_1') \\ &= mm' + E, \end{aligned}$$

for some error $E$. Recall that the evaluation key is $\texttt{evk} = \left([-a's + e' + Ps^2]_{Pq}, a'\right)$. Then, the output of $\texttt{Relinearize}$ is

$$\begin{aligned} C_0 &= d_0 + \lfloor P^{-1} \cdot d_2 \cdot (-a's + e' + Ps^2) \rceil \\ &= d_0 + P^{-1} \cdot d_2 \cdot (-a's + e' + Ps^2) + \epsilon_0 \\ C_1 &= d_1 + \lfloor P^{-1} \cdot d_2 \cdot a' \rceil \\ &= d_1 + P^{-1} \cdot d_2 \cdot a' + \epsilon_1, \end{aligned}$$

where $\epsilon_i$ are rounding errors. Decrypting this yields

$$\begin{aligned} C_0 + sC_1 &= d_0 + P^{-1}d_2(-a's + e' + Ps^2) + \epsilon_0 + sd_1 + sP^{-1}d_2a' + s\epsilon_1 \\ &= d_0 + sd_1 + s^2d_2 + (\epsilon_0 + \epsilon_2 s) + P^{-1}d_2e' \\ &= mm' + E + (\epsilon_0 + \epsilon_1 s) + P^{-1}d_2e'. \end{aligned}$$

From the previous subsection, we have that

$$\begin{aligned} \texttt{Decrypt}(\texttt{ct}, \texttt{sk}) &= c_0 + sc_1 \pmod{Q} \\ &= m + p_0c + e_1 + svp_1 + se_2 \\ &= m + ve + e_1 + se_2. \end{aligned}$$

By abuse of notation[1], we have that

$$E = (m + ve + e_1 + se_2)(m' + v'e' + e_1' + se_2'),$$

so we are indeed left with an $s^2$ term.

# 3 CKKS Error Estimation

## 3.1 Fresh ciphertext

The number of error bits in fresh ciphertext $c_1$ encrypting message $m_1$ using error distribution $N(0, \sigma^2 I_N)$ and secret distribution $\{-1, 0, 1\}$ is given by (Using Central Limit Theory (CLT) and results from [2]).

$$\epsilon_1 = \frac{1}{2}\log N(\rho^2_{fresh} + \frac{1}{12}) + logH_c(\alpha, N), \tag{1}$$

where

$$\begin{aligned} \rho^2_{fresh} &= (\frac{4}{3}N + 1)\sigma^2 \\ H_c(\alpha, N) &= (-\log(1 - (1-\alpha)^{\frac{2}{N}}))^{\frac{1}{2}}. \end{aligned}$$

$\alpha$ represents the failure probability or error tolerance.

Similarly, ciphertext $c_2$ encrypting message $m_2$ using error distribution $N(0, \sigma'^2 I_N)$ and same secret distribution can be written as

$$\epsilon_2 = \frac{1}{2}\log N(\rho'^2_{fresh} + \frac{1}{12}) + logH_c(\alpha, N), \tag{2}$$

where

$$\rho'^2_{fresh} = (\frac{4}{3}N + 1)\sigma'^2.$$

[1] I am reusing the notation $e'$

### 3.2 Addition

Adding these two ciphertext results into ciphertext with error bits

$$\epsilon_1 + \epsilon_2 = \frac{1}{2}\log N(\rho''^2_{fresh} + \frac{1}{6}) + log2H_c(\alpha, N) \tag{3}$$

with new error $N(0, \rho''^2_{fresh} I_N)$, where

$$\rho''^2_{fresh} = (\frac{4}{3}N + 1)(\sigma^2 + \sigma'^2)$$

### 3.3 Multiplication by constant

: Multiplying the ciphertext with a constant $\lambda$ results in a ciphertext with new error $N(0, \rho^2_{mulconst} I_N)$ where

$$\rho^2_{mulconst} = ||\lambda||_2^2(\frac{4}{3}N + 1)\sigma^2 \tag{4}$$

### 3.4 Multiplication

: Multiplication of two ciphertext results into ciphertext with error of the following form

$$B_{final\ error} = \Delta^{-1}(B_{mult} + B_{ks}) + B_{round} \tag{5}$$

$$\begin{aligned} B_{mult} &= N(0, N\rho^2_{fresh}\rho'^2_{fresh} I_N) \\ B_{ks} &= N(0, \eta^2_{ks} I_N) \\ B_{round} &= N(0, \eta^2_{round} I_N) \end{aligned}$$

where

$$\begin{aligned} \rho^2_{fresh} &= (\frac{4}{3}N + 1)\sigma^2 \\ \rho'^2_{fresh} &= (\frac{4}{3}N + 1)\sigma'^2 \\ \eta^2_{ks} &= \frac{1}{12}p^{-2}q_l^2 N\sigma^2 + 1_{p\nmid q_l}(\frac{N}{18} + \frac{1}{12}) \\ &= \frac{1}{12}N\sigma^2 \qquad [usually\ \ p^{-2}q_l^2 \approx 1] \\ \eta^2_{round} &= \frac{N}{18} + \frac{1}{12} \end{aligned}$$

The final error of multiplication of two ciphertext can be written down as

$$\begin{aligned} B_{final\ error} &= \Delta^{-1}(N(0, N\rho^2_{fresh}\rho'^2_{fresh}I_N) + N(0, \eta^2_{ks}I_N)) + N(0, \eta^2_{round}I_N) \\ &= N(0, \Delta^{-2}N(\rho^2_{fresh}\rho'^2_{fresh} + \frac{1}{12}\sigma^2)I_N) + N(0, (\frac{N}{18} + \frac{1}{12})I_N) \\ &= N\bigg(0, \Big(\Delta^{-2}N(\rho^2_{fresh}\rho'^2_{fresh} + \frac{1}{12}\sigma^2) + \frac{N}{18} + \frac{1}{12}\Big)I_N\bigg) \\ &= N(0, \rho^2_{mult\ error}I_N) \end{aligned} \tag{6}$$

where

$$\rho^2_{mult\ error} = \Delta^{-2}N(\rho^2_{fresh}\rho'^2_{fresh} + \frac{1}{12}\sigma^2) + \frac{N}{18} + \frac{1}{12}$$

# 4 CKKS parameter

Here is the list of parameter values suggested by the homomorphic encryption standards to retain different security levels while the secret key is sampled from ternary distribution (i.e., each coefficient is chosen uniformly from $\{-1, 0, 1\}$).

| N | security level | $\log q_{max}$ | uSVP | dec | dual |
|---|---|---|---|---|---|
| 1024 | 128 | 27 | 131.6 | 160.2 | 138.7 |
| | 192 | 19 | 193.0 | 259.5 | 207.7 |
| | 256 | 14 | 265.6 | 406.4 | 293.8 |
| 2048 | 128 | 54 | 129.7 | 144.4 | 134.2 |
| | 192 | 37 | 197.5 | 233.0 | 207.8 |
| | 256 | 29 | 259.1 | 321.7 | 273.5 |
| 4096 | 128 | 109 | 128.1 | 134.9 | 129.9 |
| | 192 | 75 | 194.7 | 212.2 | 198.5 |
| | 256 | 58 | 260.4 | 292.6 | 270.1 |
| 8192 | 128 | 218 | 128.5 | 131.5 | 129.2 |
| | 192 | 152 | 192.2 | 200.4 | 194.6 |
| | 256 | 118 | 256.7 | 273.0 | 260.6 |
| 16384 | 128 | 438 | 128.1 | 129.9 | 129.0 |
| | 192 | 305 | 192.1 | 196.2 | 193.2 |
| | 256 | 237 | 256.9 | 264.2 | 259.8 |
| 32768 | 128 | 881 | 128.5 | 129.1 | 128.5 |
| | 192 | 611 | 192.7 | 194.2 | 193.7 |
| | 256 | 476 | 256.4 | 260.2 | 258.2 |

Table 1: The differnt parameters shown in the table represents the following: $N$ is the ring dimension, security level is the bit security provided by the parameters equivalent to that of AES bit security. $\log q$ is the number of bits in the modulus $q$. $uSVP$ represents the bit security against unique shortest vector attack, *dec* represents the bit security against decoding attack and *dual* represnts the bit security against dual attack.

In CKKS encryption scheme to get the bit security level as that of values mentioned in the Table 1 the modulus value have to follow the mentioned number of bits. Which means modulus value that is used for the evaluation key have to follow the mentioned number of bits shown in the table 1[3]. For example to obtain 128-bit security when $N$ is set to 10 the value of $PQ$ should be $\leq \log_2 q_{max} = 27$-bits. $Q$ is usually set as $Q_L = q_0 \cdot \Delta^L$ (usually $q_0$ is larger than $\Delta$ for correct decryption [3] and lies in range $\Delta < q_0 < \Delta^2$, [4]), $L$ represents the multiplicative depth of the circuit. $P$ is usually selected larger than the ciphertext modulus $q_0$, so that the key switching-error is bounded by $P^{-1} \cdot q \cdot B_{ks} + B_{rs} \approx B_{rs}$ smaller than 15 P (However I am not sure, guessed from the paper [5, 3]). For example in lattigo to obtain 128-bit security following paramters values have been mentioned.

$$
\begin{aligned}
\log_2 P &= 244 \\
\log_2 Q &= 1276 \quad (236 + 40 * 26) \\
\log_2 N &= 16 \\
\log_2 Slots &= 15 \\
\sigma &= 3.2 \\
\log_2 QP &= 1520 \\
levels &= 26 \\
scale &= 2^{40} \\
H(192; 32)
\end{aligned}
$$

## References


[1] Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23, Springer (2017) 409–437

[2] Costache, A., Curtis, B.R., Hales, E., Murphy, S., Ogilvie, T., Player, R.: On the precision loss in approximate homomorphic encryption. Cryptology ePrint Archive (2022)

[3] Kim, A., Papadimitriou, A., Polyakov, Y.: Approximate homomorphic encryption with reduced approximation error. In: Topics in Cryptology–CT-RSA 2022: Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1–2, 2022, Proceedings, Springer (2022) 120–144

[4] Cheon, J.H., Son, Y., Yhee, D.: Practical fhe parameters against lattice attacks. Cryptology ePrint Archive (2021)

[5] Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings, Springer (2020) 364–390