

# Lecture 12

## The CKKS Scheme: Approximate Arithmetic

SEC 595: Encrypted Computing

### Roadmap

BFV (Lecture 10) and BGV (Lecture 11) keep the RLWE noise below a threshold so that decryption is exact. CKKS (Cheon, Kim, Kim and Song, 2017) instead treats the noise as fixed-point rounding error and keeps it. Section 1 states the idea. Section 2 shows how a vector of numbers becomes a polynomial: the numbers are stored as the values of the polynomial at the roots of  $X^n+1$ . Sections 3 and 4 give the scheme: keys, encryption, decryption, addition, multiplication and relinearization. Section 5 introduces RESCALE, which undoes the squaring of the scale, and Section 6 builds the modulus chain on it. Section 7 measures precision and Section 8 security. Section 9 runs the scheme end to end on a toy instance, and Section 10 places the three schemes side by side.

## 1. Noise as Rounding Error

### 1.1 The Idea

In every RLWE scheme decryption produces  $c_0 + c_1SK = (\text{scaled message}) + v$  with  $v$  small and unknown. BFV removes  $v$  by rounding, BGV by reduction modulo  $t$ . CKKS observes that a scaled message plus a small perturbation is exactly what fixed-point arithmetic stores anyway, and declares the noise part of the answer. The price is that correctness holds only approximately:

$$\text{Decrypt}(\text{Encrypt}(\mu)) \approx \mu, \quad \text{Decrypt}(ct_1 * ct_2) \approx \text{Decrypt}(ct_1) * \text{Decrypt}(ct_2).$$

The ring, the keys, relinearization and the modulus chain are inherited almost unchanged.

### 1.2 Fixed Point, Recalled

In Lecture 1 a real  $x$  is stored as  $\hat{x} = \lfloor \Delta x \rfloor$  with  $\Delta = 2^f$ . Addition keeps the scale,  $\hat{x} + \hat{y} \approx \Delta(x + y)$ ; multiplication squares it,  $\hat{x}\hat{y} \approx \Delta^2 xy$ , and the scale is restored by **rescaling**: dividing by  $\Delta$  and rounding. Two facts carry the whole design.

**Fact A.** A large  $\Delta$  makes a small error harmless: with error of size  $2^k$  at scale  $2^f$ , about  $f - k$  bits of the value survive.

**Fact B.** Multiplication squares the scale; something must undo it.

**Example 1** (The  $Q_{4.4}$  round trip). Take  $\Delta = 16$ ,  $x = 2.75$ ,  $y = 3.25$ .

- (a)  $\hat{x} = 16 \cdot 2.75 = 44$  and  $\hat{y} = 16 \cdot 3.25 = 52$ , both exact.
- (b)  $\hat{x}\hat{y} = 2288$ , at scale  $\Delta^2 = 256$ .
- (c)  $\lfloor 2288/16 \rfloor = 143$  at scale 16, and  $143/16 = 8.9375 = xy$ .

- (d) With errors  $e_x = 3$ ,  $e_y = -2$ :  $(44 + 3)(52 - 2) = 2288 + 3 \cdot 52 - 2 \cdot 44 - 6 = 2350$ , and  $\lfloor 2350/16 \rfloor = \lfloor 146.875 \rfloor = 147$ , i.e. 9.1875. The error is  $0.25 = 2^{-2}$ : two of the four fractional bits survive. Note that the terms linear in the errors (156 and  $-88$ ) dominate the quadratic term ( $-6$ ).

## 2. Encoding and Decoding

The user holds a vector of numbers; CKKS encrypts a polynomial. Encoding is the bridge, and it is chosen so that *one* polynomial product performs  $n/2$  numeric products at once.

### 2.1 The Wrong Way: Values in the Coefficients

Put the numbers into the coefficients. Addition works coefficient by coefficient, but multiplication convolves:

$$(1 + 2X)(3 + 4X) = 3 + \underbrace{(1 \cdot 4 + 2 \cdot 3)}_{\text{mixed}} X + 8X^2.$$

The products  $1 \cdot 3$  and  $2 \cdot 4$  appear only at the ends, and the middle coefficient mixes the two inputs. We need a representation in which polynomial multiplication acts entry by entry.

### 2.2 The Right Way: Values at Points

Evaluation at a fixed point  $w$  respects both operations:

$$(a + b)(w) = a(w) + b(w), \quad (ab)(w) = a(w)b(w).$$

So store each number as the *value of the polynomial at a point*. Working modulo a polynomial  $f$  that vanishes at all the points changes nothing there, because  $f(w) = 0$ ; it only keeps the degree small.

**Example 2** (Two values at two points). Use the points  $w = 1$  and  $w = -1$  and work modulo  $X^2 - 1$ , which vanishes at both.

- (a) Store the first value at  $w = 1$  and the second at  $w = -1$ . For  $(3, 5)$  we need  $a(X) = c_0 + c_1X$  with  $a(1) = c_0 + c_1 = 3$  and  $a(-1) = c_0 - c_1 = 5$ , so  $c_0 = 4$ ,  $c_1 = -1$  and  $a(X) = 4 - X$ : the line through  $(1, 3)$  and  $(-1, 5)$ . Likewise  $(2, 1)$  gives  $b(X) = \frac{3}{2} + \frac{1}{2}X$ . Finding the polynomial from its values is *interpolation*.
- (b)  $ab = 6 + \frac{1}{2}X - \frac{1}{2}X^2 \equiv \frac{11}{2} + \frac{1}{2}X \pmod{X^2 - 1}$ , using  $X^2 = 1$ .
- (c)  $(ab)(1) = 6 = 3 \cdot 2$  and  $(ab)(-1) = 5 = 5 \cdot 1$ : one polynomial product gave both numeric products (Figure 1).

### 2.3 The Points of CKKS: Slots

CKKS works modulo  $X^n + 1$ , so its points are the roots of  $X^n + 1$ : the odd powers  $\zeta^j$  of  $\zeta = e^{\pi i/n}$  (Lecture 4),  $n$  points evenly spaced on the unit circle. Indeed  $(\zeta^j)^n = e^{\pi i j} = (-1)^j$ , which is  $-1$  exactly when  $j$  is odd; the even powers are roots of  $X^n - 1$  instead. They are complex, so the stored values are complex.

A polynomial with real coefficients satisfies  $a(\overline{w}) = \overline{a(w)}$ : the value at a root fixes the value at its mirror image across the real axis. So only one root of each conjugate pair is free, which leaves  $n/2$  values. These are the **slots** (Figure 2).

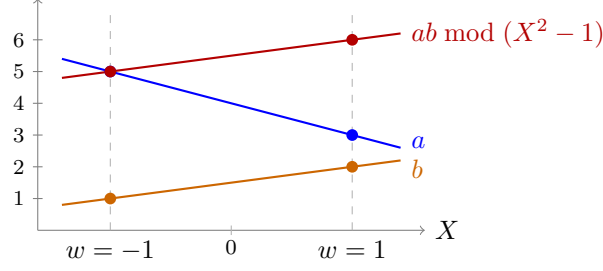


Figure 1: The values live at the points. At each point, the value of  $ab$  is the product of the values of  $a$  and  $b$ .

**Definition 1** (Canonical embedding).

$$\tau : \mathbb{R}[X]/(X^n + 1) \rightarrow \mathbb{C}^{n/2}, \quad \tau(a) = (a(\zeta), a(\zeta^5), a(\zeta^{25}), \dots).$$

$\tau$  is a bijection (both sides have real dimension  $n$ ), it is linear, and it is multiplicative, since each coordinate is an evaluation: ring multiplication is slotwise multiplication. At  $n = 2^{13}$  one ciphertext holds 4096 values. The roots  $\zeta^{5^j}$  are chosen so that  $X \mapsto X^5$  rotates the slots cyclically.

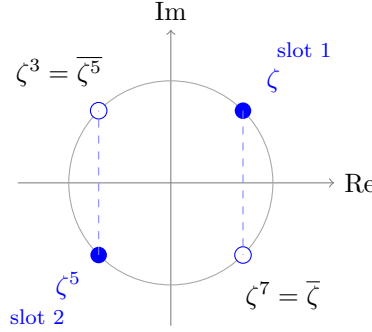


Figure 2:  $n = 4$ : the four roots of  $X^4 + 1$ . The filled roots carry the two slots; the hollow roots are their mirror images and carry the conjugate values.

**Example 3** (The canonical embedding,  $n = 4$ ). The roots of  $X^4 + 1$  are  $\zeta = \frac{1+i}{\sqrt{2}}$ ,  $\zeta^3 = \frac{-1+i}{\sqrt{2}}$ ,  $\zeta^5 = \frac{-1-i}{\sqrt{2}}$  and  $\zeta^7 = \frac{1-i}{\sqrt{2}}$ . Take  $a(X) = 1 + X + X^3$ . Any real polynomial would do; this one is chosen because its values come out clean.

- (a) *Evaluate at every root.* For example, at  $\zeta$  the two powers are  $\zeta = \frac{1+i}{\sqrt{2}}$  and  $\zeta^3 = \frac{-1+i}{\sqrt{2}}$ ; their real parts cancel and their imaginary parts add, so  $\zeta + \zeta^3 = \frac{2i}{\sqrt{2}} = \sqrt{2}i$  and  $a(\zeta) = 1 + \zeta + \zeta^3 = 1 + \sqrt{2}i$ . In full:

| root $w$ | $\zeta$ (slot 1) | $\zeta^3$       | $\zeta^5$ (slot 2) | $\zeta^7$       |
|----------|------------------|-----------------|--------------------|-----------------|
| $a(w)$   | $1 + \sqrt{2}i$  | $1 + \sqrt{2}i$ | $1 - \sqrt{2}i$    | $1 - \sqrt{2}i$ |

- (b) *Mirror pairs.*  $\zeta^7 = \bar{\zeta}$  and  $a(\zeta^7) = \overline{a(\zeta)}$ ;  $\zeta^3 = \bar{\zeta^5}$  and  $a(\zeta^3) = \overline{a(\zeta^5)}$ . The hollow roots of Figure 2 carry no new information.
- (c) *Keep one root per pair.*  $\tau(a) = (a(\zeta), a(\zeta^5)) = (1 + \sqrt{2}i, 1 - \sqrt{2}i)$ : four real coefficients become two complex slots.
- (d) *Products are slotwise.* Let  $b(X) = X^2$ , so  $\tau(b) = (\zeta^2, \zeta^{10}) = (i, i)$ . Then  $ab = X^2 + X^3 + X^5 \equiv -X + X^2 + X^3$ , using  $X^4 = -1$ , and evaluating gives  $\tau(ab) = (-\sqrt{2} + i, \sqrt{2} + i) =$

$((1 + \sqrt{2}i)i, (1 - \sqrt{2}i)i)$ : each slot of  $a$  times the matching slot of  $b$ . ✓

Section 2.4 runs this map backwards: interpolating the slots  $(1 + \sqrt{2}i, 1 - \sqrt{2}i)$  returns  $a(X) = 1 + X + X^3$ .

## 2.4 ENCODE and DECODE

Going from slot values back to a polynomial is interpolation,  $\tau^{-1}$ . For  $n = 4$ , with  $(z_1, z_2)$  at  $(\zeta, \zeta^5)$  and the conjugates at the other two roots,

$$a_k = \frac{1}{2} \operatorname{Re}(z_1 \zeta^{-k} + z_2 \zeta^{-5k}), \quad k = 0, 1, 2, 3.$$

The coefficients are real but usually irrational. Fixed point (Lecture 1) repairs this: scale by  $\Delta$  and round.

**Definition 2** (ENCODE, DECODE). For a scale factor  $\Delta > 0$ ,

$$\text{ENCODE}(z, \Delta) = \lfloor \Delta \cdot \tau^{-1}(z) \rfloor \in R, \quad \text{DECODE}(a, \Delta) = \tau\left(\frac{a}{\Delta}\right) \in \mathbb{C}^{n/2},$$

with rounding applied coefficientwise.

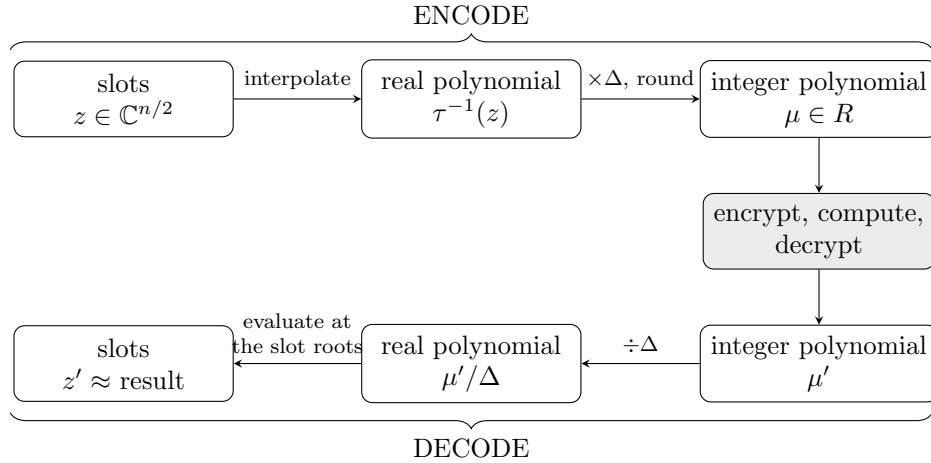


Figure 3: ENCODE is interpolation followed by fixed point; DECODE undoes both. Encryption sees only the integer polynomial.

These are Lecture 1's  $\lfloor x \cdot 2^f \rfloor$  and  $x/2^f$  after a change of basis that makes multiplication slotwise. Rounding moves each coefficient by at most  $1/2$ ; at a root of modulus 1 that moves the value by at most  $n/2$ , and after the division by  $\Delta$  by at most  $n/(2\Delta)$ .  $\Delta$  wants to be large for precision; the modulus budget (Section 6) wants it small. That tension is the design of the scheme.

**Example 4** (Encoding a real vector,  $n = 4$ ). Example 3 went from a polynomial to its slots; this example goes back. Use the same roots  $\zeta = \frac{1+i}{\sqrt{2}}$ ,  $\zeta^3 = \frac{-1+i}{\sqrt{2}}$ ,  $\zeta^5 = \frac{-1-i}{\sqrt{2}}$ ,  $\zeta^7 = \frac{1-i}{\sqrt{2}}$ , and recall  $\zeta^2 = i$ ,  $\zeta^4 = -1$ ,  $\zeta^8 = 1$ . Encode the slots  $z = (2, 1)$  with  $\Delta = 100$ .

- (a) *Place the values at every root.* Slot 1 is  $\zeta$  and slot 2 is  $\zeta^5$ . The mirror roots receive the conjugates (as in Example 3(b)); here the values are real, so they are unchanged:

| root $w$      | $\zeta$ (slot 1) | $\zeta^3$     | $\zeta^5$ (slot 2) | $\zeta^7$     |
|---------------|------------------|---------------|--------------------|---------------|
| wanted $a(w)$ | 2                | $\bar{1} = 1$ | 1                  | $\bar{2} = 2$ |

- (b) *Interpolate.* Use  $a_k = \frac{1}{2}\text{Re}(z_1\zeta^{-k} + z_2\zeta^{-5k})$  with  $z_1 = 2$ ,  $z_2 = 1$ . Negative powers are reduced with  $\zeta^8 = 1$ :  $\zeta^{-1} = \zeta^7$ ,  $\zeta^{-2} = \zeta^6 = \zeta^4\zeta^2 = -i$ ,  $\zeta^{-3} = \zeta^5$ ,  $\zeta^{-5} = \zeta^3$ ,  $\zeta^{-10} = \zeta^6 = -i$ ,  $\zeta^{-15} = \zeta^1$ . Then

$$\begin{aligned}
k = 0: \quad & 2 \cdot 1 + 1 = 3, & a_0 &= \frac{3}{2}, \\
k = 1: \quad & 2\zeta^7 + \zeta^3 = \frac{(2-2i)+(-1+i)}{\sqrt{2}} = \frac{1-i}{\sqrt{2}}, & a_1 &= \frac{1}{2} \cdot \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{4} \approx 0.3536, \\
k = 2: \quad & 2(-i) + (-i) = -3i, & a_2 &= 0, \\
k = 3: \quad & 2\zeta^5 + \zeta^1 = \frac{(-2-2i)+(1+i)}{\sqrt{2}} = \frac{-1-i}{\sqrt{2}}, & a_3 &= -\frac{\sqrt{2}}{4} \approx -0.3536.
\end{aligned}$$

So  $\tau^{-1}(z) = \frac{3}{2} + \frac{\sqrt{2}}{4}(X - X^3) \approx 1.5 + 0.3536X - 0.3536X^3$ .

- (c) *Check: evaluate at every root,* as in Example 3(a). Substitute each root  $w$  for  $X$  in  $\tau^{-1}(z) = \frac{3}{2} + \frac{\sqrt{2}}{4}X - \frac{\sqrt{2}}{4}X^3$ . Each root has the form  $\frac{\pm 1 \pm i}{\sqrt{2}}$ , so  $\frac{\sqrt{2}}{4} \cdot \frac{\pm 1 \pm i}{\sqrt{2}} = \frac{\pm 1 \pm i}{4}$ . Powers above 7 are reduced with  $\zeta^8 = 1$ .

$$\begin{aligned}
w = \zeta: \quad & w^3 = \zeta^3; & \frac{3}{2} + \frac{1+i}{4} - \frac{-1+i}{4} &= \frac{3}{2} + \frac{2}{4} = 2, \\
w = \zeta^3: \quad & w^3 = \zeta^9 = \zeta; & \frac{3}{2} + \frac{-1+i}{4} - \frac{1+i}{4} &= \frac{3}{2} - \frac{2}{4} = 1, \\
w = \zeta^5: \quad & w^3 = \zeta^{15} = \zeta^7; & \frac{3}{2} + \frac{-1-i}{4} - \frac{1-i}{4} &= \frac{3}{2} - \frac{2}{4} = 1, \\
w = \zeta^7: \quad & w^3 = \zeta^{21} = \zeta^5; & \frac{3}{2} + \frac{1-i}{4} - \frac{-1-i}{4} &= \frac{3}{2} + \frac{2}{4} = 2.
\end{aligned}$$

In each line the imaginary parts cancel. The values 2, 1, 1, 2 are the table of (a). ✓

- (d) *Scale and round.* Multiply each coefficient by  $\Delta = 100$  and round to the nearest integer:

$$100 \cdot \frac{3}{2} = 150, \quad 100 \cdot 0.35355 = 35.355 \rightarrow 35, \quad 100 \cdot 0 = 0, \quad 100 \cdot (-0.35355) = -35.355 \rightarrow -35,$$

so  $\mu = \text{ENCODE}(z, 100) = 150 + 35X - 35X^3 \in R$ .

- (e) *Decode: keep one root per pair and divide by  $\Delta$ ,* as in Example 3(c). Substitute the slot roots  $\zeta$  and  $\zeta^5$  into  $\mu(X) = 150 + 35X - 35X^3$ , using  $35 \cdot \frac{\pm 1 \pm i}{\sqrt{2}} = 24.749(\pm 1 \pm i)$ :

$$\begin{aligned}
\mu(\zeta) &= 150 + 24.749(1+i) - 24.749(-1+i) = 150 + 49.497 = 199.497, \\
\mu(\zeta^5) &= 150 + 24.749(-1-i) - 24.749(1-i) = 150 - 49.497 = 100.503.
\end{aligned}$$

Dividing by  $\Delta = 100$ :  $\text{DECODE}(\mu, 100) = (1.99497, 1.00503)$ . Each slot is off by 0.005, under the bound  $n/(2\Delta) = 0.02$ .

- (f) *A larger scale.* At  $\Delta = 1000$  the coefficients become 1500,  $353.55 \rightarrow 354$ ,  $0$ ,  $-353.55 \rightarrow -354$ , so  $\mu = 1500 + 354X - 354X^3$ . With  $354/\sqrt{2} = 250.316$ :

$$\mu(\zeta) = 1500 + 2(250.316) = 2000.632, \quad \mu(\zeta^5) = 1500 - 2(250.316) = 999.368,$$

which decode to (2.00063, 0.99937). The error is  $6.3 \times 10^{-4}$ : ten times the scale, about a tenth of the error.

(g) *Products are slotwise*, as in Example 3(d). Back at  $\Delta = 100$ , square  $\mu = 150 + 35X - 35X^3$  term by term:

$$\begin{aligned}\mu^2 &= 150^2 + 2(150)(35X) + 2(150)(-35X^3) + (35X)^2 + 2(35X)(-35X^3) + (-35X^3)^2 \\ &= 22500 + 10500X - 10500X^3 + 1225X^2 - 2450X^4 + 1225X^6.\end{aligned}$$

Reduce with  $X^4 = -1$  and  $X^6 = X^4X^2 = -X^2$ :  $-2450X^4 = +2450$  and  $1225X^6 = -1225X^2$ , so

$$\mu^2 \equiv (22500 + 2450) + 10500X + (1225 - 1225)X^2 - 10500X^3 = 24950 + 10500X - 10500X^3.$$

Its scale is  $\Delta^2 = 10^4$ . Decode as in (e), with  $10500/\sqrt{2} = 7424.621$ :

$$\begin{aligned}\mu^2(\zeta) &= 24950 + 2(7424.621) = 39799.24, \quad 39799.24/10^4 = 3.97992 = 1.99497^2, \\ \mu^2(\zeta^5) &= 24950 - 2(7424.621) = 10100.76, \quad 10100.76/10^4 = 1.01008 = 1.00503^2.\end{aligned}$$

Squaring the polynomial squared each slot,  $\approx (4, 1)$ , and the scale became  $\Delta^2$  (Section 5 removes the extra  $\Delta$ ).

### 3. Keys, Encryption, Decryption

#### 3.1 KeyGen and Encrypt

Plaintexts are  $\mathbb{C}^{n/2} \xrightarrow{\text{ENCODE}} R = \mathbb{Z}[X]/(X^n + 1)$  and ciphertexts lie in  $R_{q_\ell}^2$  at a level  $\ell$  (Section 6). There is **no plaintext modulus**  $t$ . The distributions are those of Lecture 6, §1:  $R_2$  ternary,  $\mathcal{X}$  a discrete Gaussian with bound  $\beta$ .

**Definition 3** (KeyGen, Encrypt). Sample  $\text{SK} \leftarrow R_2$ ,  $a \leftarrow R_{q_L}$ ,  $e \leftarrow \mathcal{X}$  and set

$$\text{PK} = ([-(a\text{SK} + e)]_{q_L}, a), \quad \text{so} \quad \text{PK}_0 + \text{PK}_1\text{SK} = -e.$$

To encrypt  $\mu \in R$ , sample  $u \leftarrow R_2$ ,  $e_1, e_2 \leftarrow \mathcal{X}$  and output

$$c_0 = [\text{PK}_0 u + e_1 + \mu]_{q_\ell}, \quad c_1 = [\text{PK}_1 u + e_2]_{q_\ell}.$$

Compare BFV's  $c_0 = \text{PK}_0 u + e_1 + \Delta M$  and BGV's  $c_0 = \text{PK}_0 u + t e_1 + M$ . The only change from BFV is that the message carries no  $\Delta$ : ENCODE has already applied it.

#### 3.2 Decrypt

**Definition 4** (Decrypt). Compute  $\hat{\mu} = [c_0 + c_1\text{SK}]_{q_\ell}$ , then output  $\text{DECODE}(\hat{\mu}, \Delta)$ .

There is no rounding, no division and no reduction modulo  $t$ : decryption is the bare evaluation  $\text{ct}(\text{SK})$ .

**Proposition 1** (Decryption identity). *For a fresh ciphertext,  $c_0 + c_1\text{SK} = \mu + v \pmod{q_\ell}$  with  $v = -eu + e_1 + e_2\text{SK}$ .*

*Proof.*  $c_0 + c_1\text{SK} = \mu + e_1 + e_2\text{SK} + u(\text{PK}_0 + \text{PK}_1\text{SK}) = \mu + e_1 + e_2\text{SK} - eu.$  □

The mask  $au\text{SK}$  cancels as in BFV, and  $v$  is BFV's noise, so  $\|v\| \leq n\beta + \beta + n\beta = 2n\beta + \beta$  (Lecture 10). Nothing removes it.

**Proposition 2** (Approximate correctness). *If  $\|\mu + v\| < q_\ell/2$ , decryption returns  $\mu + v$  exactly in  $R$ , and*

$$\text{DECODE}(\hat{\mu}, \Delta) = z + \frac{\tau(v)}{\Delta} + \frac{\tau(\varepsilon)}{\Delta},$$

where  $\varepsilon = \Delta\tau^{-1}(z) - \mu$  is the encoding error. Otherwise the output is unrelated to  $z$ .

*Proof.* Over  $\mathbb{Z}$ ,  $c_0 + c_1\text{SK} = \mu + v + q_\ell k$ . If  $\|\mu + v\| < q_\ell/2$  the centred reduction removes  $q_\ell k$  exactly; then  $\mu = \Delta\tau^{-1}(z) - \varepsilon$  and  $\mathbb{R}$ -linearity of  $\tau$  give the formula. If not, a nonzero multiple of  $q_\ell$  remains, which is enormous after division by  $\Delta$ .  $\square$

Correctness is only about wraparound: there is no threshold below which the answer becomes exact. Since  $\mu \approx \Delta\tau^{-1}(z)$ , the condition bounds the data: roughly  $\|z\| < q_\ell/(2\Delta)$ . At  $q_0 = 2^{60}$ ,  $\Delta = 2^{40}$  this allows values up to about  $2^{20}$ , and overflow is silent.

## 4. EvalAdd, EvalMult, Relinearization

### 4.1 EvalAdd

**Definition 5** (EvalAdd). For ciphertexts at the same level  $\ell$  and the same scale  $\Delta$ ,

$$\text{EvalAdd}(\text{ct}^{(1)}, \text{ct}^{(2)}) = ([c_0^{(1)} + c_0^{(2)}]_{q_\ell}, [c_1^{(1)} + c_1^{(2)}]_{q_\ell}).$$

The sum decrypts to  $\mu_1 + \mu_2 + (v_1 + v_2)$ , so  $\|v_{\text{add}}\| \leq \|v_1\| + \|v_2\|$ , the scale is unchanged and no level is consumed. The new condition is scale alignment:  $\Delta z + \Delta' z'$  decodes to nothing meaningful unless  $\Delta = \Delta'$ .

### 4.2 EvalMult and the Squared Scale

**Definition 6** (EvalMult).

$$\text{EvalMult}(\text{ct}^{(1)}, \text{ct}^{(2)}) = (c_0^{(1)}c_0^{(2)}, c_0^{(1)}c_1^{(2)} + c_1^{(1)}c_0^{(2)}, c_1^{(1)}c_1^{(2)}) \bmod q_\ell.$$

This is the product in  $R_{q_\ell}[S]$  (Lecture 4), identical in BFV, BGV and CKKS. What differs is the plaintext arithmetic. With scaled values  $\Delta x + e_x$  and  $\Delta y + e_y$ ,

$$(\Delta x + e_x)(\Delta y + e_y) = \underbrace{\Delta^2 xy}_{\text{payload}} + \Delta(xe_y + ye_x) + e_x e_y,$$

while in BGV  $(te_1 + M_1)(te_2 + M_2) = t^2 e_1 e_2 + t(e_1 M_2 + e_2 M_1) + M_1 M_2$ . The same three terms appear with the roles of payload and noise exchanged. The scale becomes  $\Delta^2$ . A plaintext–ciphertext product  $(\mu' c_0, \mu' c_1)$  needs no relinearization but also squares the scale.

**Proposition 3** (Multiplicative noise growth). *If  $\text{ct}^{(i)}(\text{SK}) = \mu_i + v_i$ , the product decrypts to  $\mu_1 \mu_2 + v_\times$  with  $v_\times = \mu_1 v_2 + \mu_2 v_1 + v_1 v_2$  and*

$$\|v_\times\| \leq n(\|\mu_1\|\|v_2\| + \|\mu_2\|\|v_1\|) + n\|v_1\|\|v_2\|.$$

*Proof.* Expand  $(\mu_1 + v_1)(\mu_2 + v_2)$  and apply  $\|ab\| \leq n\|a\|\|b\|$  (Lecture 3) to each term.  $\square$

In BGV the dangerous term is  $t^2 e_1 e_2$ , quadratic in the noise. In CKKS  $\mu_i \approx \Delta z_i$  is far larger than  $v_i$ , so the *linear* terms dominate. Relative to  $\Delta^2$ , a multiplication costs a bit or two of precision and does not compound.

### 4.3 Relinearization

**Definition 7** (EK and Relinearize). Publish  $EK = ([-(aSK + e) + SK^2]_{q_\ell}, a)$ , so  $EK_0 + EK_1SK = SK^2 - e$ . For  $c^* = (c_0^*, c_1^*, c_2^*)$  set

$$\hat{c}_0 = c_0^* + EK_0 c_2^*, \quad \hat{c}_1 = c_1^* + EK_1 c_2^*.$$

Then  $\hat{c}_0 + \hat{c}_1 SK = c_0^* + c_1^* SK + c_2^* SK^2 - c_2^* e$ . This is BFV's construction unchanged. BGV needed a factor  $t$  in EK to protect the residue modulo  $t$ ; CKKS has no residue to protect, so  $-c_2^* e$  is only more noise. With gadget base  $w$  and  $\ell_w = \lceil \log_w q_\ell \rceil$  (Lecture 4) the added noise is

$$\|v_{\text{Relinearize}}\| \leq n \ell_w (w - 1) \beta.$$

## 5. Rescale

### 5.1 The Operation

**Definition 8** (RESCALE). For  $ct \in R_{q_\ell}^2$  at scale  $\Delta^2$ ,

$$\text{RESCALE}(ct, \Delta) = \lfloor \Delta^{-1} \cdot ct \rfloor \in R_{q_{\ell-1}}^2, \quad q_{\ell-1} = q_\ell / \Delta,$$

coefficientwise; the result has scale  $\Delta$ .

**Proposition 4.** If  $ct(SK) = \Delta^2 \mu + v \pmod{q_\ell}$ , then  $ct'(SK) = \Delta \mu + \Delta^{-1} v + r \pmod{q_{\ell-1}}$  with  $\|r\| \leq \frac{1}{2}(1 + n\|SK\|)$ .

*Proof.* Write  $c'_i = \Delta^{-1} c_i + r_i$  with  $\|r_i\| \leq \frac{1}{2}$ . Over  $\mathbb{Z}$ ,  $c_0 + c_1 SK = \Delta^2 \mu + v + q_\ell k$ , so  $c'_0 + c'_1 SK = \Delta \mu + \Delta^{-1} v + q_{\ell-1} k + (r_0 + r_1 SK)$ , and  $\|r_0 + r_1 SK\| \leq \frac{1}{2} + \frac{1}{2} n\|SK\|$ .  $\square$

The payload is the same at scale  $\Delta$ , the noise is divided by  $\Delta$ , and rounding adds at most  $\frac{1}{2}(1 + n\|SK\|)$ .

### 5.2 MODSWITCH Under Another Name

|            | BGV MODSWITCH               | CKKS RESCALE                          |
|------------|-----------------------------|---------------------------------------|
| Formula    | $\lfloor (q'/q) ct \rfloor$ | $\lfloor (q'/q) ct \rfloor$           |
| Rounded    | within the class mod $t$    | to the nearest integer                |
| Purpose    | shrink the noise            | restore $\Delta^2 \rightarrow \Delta$ |
| Constraint | $q \equiv q' \pmod{t}$      | none                                  |
| Message    | exactly preserved           | divided by $q/q'$                     |

BGV needs the congruence because it reduces the plaintext modulo  $t$ , and the term  $(q - q')k$  would disturb the residue (Lecture 11). CKKS never reduces the plaintext modulo anything, so no congruence is needed and each rounding correction is at most  $1/2$ . Same formula, different disease: in BGV the *noise* explodes; in CKKS the *scale* does.

### 5.3 What RESCALE Does, and Does Not, Do

It does not improve precision: payload and noise are divided alike, so  $\|v\|/\text{scale}$  is unchanged up to the rounding term. What it buys is the growth rate of the modulus. Without RESCALE,  $d$  multiplications leave scale  $\Delta^{2^d}$ , so  $\log q > 2^d \log \Delta$ . With it, the scale returns to  $\Delta$  each time, so  $\log q > L \log \Delta$ . Exponential becomes linear: that is Fact B.

**Example 5** (A rescale trace). At level 1,  $q_1 = 100000190000$ , a ciphertext at scale  $\Delta^2 = 10^8$  is (this is the relinearized product of Section 9)

$$\begin{aligned}\hat{c}_0 &= -6564956194 + 25551957201X + 21920360407X^2 + 28669366452X^3, \\ \hat{c}_1 &= -9461472835 + 40573592885X - 25296998807X^2 - 43550170004X^3,\end{aligned}$$

with  $\Delta = 10^4$ ,  $SK = 1 - X + X^3$  and noise  $\|v\| = 117420$ . Dividing by  $\Delta$  and rounding, with corrections  $c' - \Delta^{-1}c$ :

|       | $\Delta^{-1}\hat{c}_0$ | correction | $\Delta^{-1}\hat{c}_1$ | correction |
|-------|------------------------|------------|------------------------|------------|
| 1     | -656495.6194           | -0.3806    | -946147.2835           | +0.2835    |
| $X$   | 2555195.7201           | +0.2799    | 4057359.2885           | -0.2885    |
| $X^2$ | 2192036.0407           | -0.0407    | -2529699.8807          | -0.1193    |
| $X^3$ | 2866936.6452           | +0.3548    | -4355017.0004          | +0.0004    |

Every correction is at most  $1/2$  in size. The new modulus is  $q_0 = q_1/\Delta = 10000019$  and

$$\begin{aligned}c'_0 &= -656496 + 2555196X + 2192036X^2 + 2866937X^3, \\ c'_1 &= -946147 + 4057359X - 2529700X^2 - 4355017X^3.\end{aligned}$$

Evaluating at  $SK$  modulo  $q_0$  gives  $-15000 + 88383X - 40006X^2 + 95473X^3$ . Against the rounded payload  $\lfloor \mu^{(1)}\mu^{(2)}/\Delta \rfloor = -15001 + 88388X - 40001X^2 + 95460X^3$  (the two ties  $88388.5$  and  $95460.5$  rounded down), the noise is

$$v' = 1 - 5X - 5X^2 + 13X^3, \quad \|v'\| = 13.$$

Before the switch  $\|v\|/\Delta = 117420/10^4 = 11.74$ ; the extra  $1.26$  is rounding, within the bound  $\frac{1}{2}(1 + n\|SK\|) = \frac{1}{2}(1 + 4) = 2.5$  of Proposition 4.

## 6. Levels

### 6.1 The Modulus Chain

**Definition 9** (Modulus chain). Fix a base modulus  $q_0 \gg \Delta$  and a depth  $L$ , and set  $q_\ell = q_0 \cdot \Delta^\ell$  for  $0 \leq \ell \leq L$ ,  $q = q_L$ . A ciphertext with modulus  $q_\ell$  is at **level**  $\ell$ .

| Operation   | in                 | out                  | costs             |
|-------------|--------------------|----------------------|-------------------|
| EvalAdd     | $(\ell, \Delta)$   | $(\ell, \Delta)$     | nothing           |
| EvalMult    | $(\ell, \Delta)$   | $(\ell, \Delta^2)$   | a third component |
| Relinearize | $(\ell, \Delta^2)$ | $(\ell, \Delta^2)$   | a little noise    |
| RESCALE     | $(\ell, \Delta^2)$ | $(\ell - 1, \Delta)$ | <b>one level</b>  |

Level is computational capability. At level 0 only addition and decryption remain: there is no room for a squared scale.  $q_0$  is chosen  $\gg \Delta$  so that the final payload still fits (Proposition 2).

### 6.2 The Depth Budget

Taking logarithms of the chain,

$$\log q > L \cdot \log \Delta,$$

with slack  $\log q_0$ . For fixed security a larger  $\log q$  forces a larger  $n$  (Lecture 17), and every operation becomes slower: depth, precision and speed are one resource. Circuit shape matters:  $(xy)(zw)$  is a balanced tree of depth 2 and needs  $q_0\Delta^2$ , while  $((xy)z)w$  is a chain of depth 3 and needs  $q_0\Delta^3$ , for the same three multiplications. Compiling for CKKS means minimising depth, not operation count.

### 6.3 Song's Parameter Set

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| Ring dimension        | $n = 2^{13}$                                          |
| Base modulus          | $\log_2 q_0 = 60$                                     |
| Scale                 | $\log_2 \Delta = 40 = \log_2 q_1 = \log_2 q_2$        |
| Special prime         | $\log_2 q_3 = 60$                                     |
| Total                 | $60 + 40 + 40 + 60 = 200 \leq 218$ (security ceiling) |
| Depth                 | $L = 2$                                               |
| Precision             | roughly 30 bits                                       |
| Data range at level 0 | $q_0/\Delta = 2^{20}$                                 |

$q_3$  is not a level: the keys live at  $q_0 q_1 q_2 q_3$  so that key-switching noise can be divided away. In practice  $q_\ell = q_0 q_1 \cdots q_\ell$  in RNS (Lecture 4) with primes  $q_i \approx \Delta$ . RESCALE then divides by  $q_\ell$ , not  $\Delta$ , so the scale drifts:  $\Delta' = \Delta^2/q_\ell \approx \Delta$ , and libraries track the scale per ciphertext.

## 7. Precision

**Proposition 5** (Precision). *A ciphertext at scale  $\Delta$  with noise  $v$  decodes to slots correct to about  $\log_2 \Delta - \log_2 \|v\|$  bits.*

*Sketch.* By Proposition 2 the slot error is  $\tau(v)/\Delta$  plus the smaller encoding term, of size about  $\|v\|/\Delta$ ; take  $-\log_2$ .  $\square$

Song reports  $\log_2 \|v\| \approx 10$ – $15$  for basic operations, so  $\log_2 \Delta = 40$  gives 25–30 bits. The error analysis of a CKKS circuit is the error analysis of the same circuit in floating point.

**Example 6** (Precision calculations). (a) At  $\Delta = 2^{40}$  and  $\|v\| = 2^{12}$ :  $40 - 12 = 28$  bits, i.e.  $28 \log_{10} 2 \approx 8.4$  decimal digits.  
 (b) At  $\Delta = 10^4$  with  $\|v'\| = 13$ : predicted slot error  $13/10^4 = 1.3 \times 10^{-3}$ .  
 (c) After one multiplication,  $\|v_\times\| \approx n\|\mu\|\|v\|$  with  $\|\mu\| \approx \Delta\|z\|$ , at scale  $\Delta^2$ , so

$$\log_2 \frac{\Delta^2}{n\|\mu\|\|v\|} \approx \log_2 \frac{\Delta}{n\|z\|\|v\|} \text{ bits.}$$

(d) With  $\|v\| \approx 2^{10}$  and  $n\|z\| \approx 2^{16}$ , 30 bits require  $\log_2 \Delta = 30 + 16 + 10 = 56$ . In practice one takes  $\Delta = 2^{40}$  and accepts precision in the twenties.

## 8. Security

### 8.1 IND-CPA and IND-CPA<sup>+</sup>

PK is an RLWE sample, so IND-CPA (Lecture 8, §1.5) follows from decision RLWE (Lecture 6, §2.2) exactly as for BFV and BGV. A stronger model captures a client that decrypts and publishes the result.

**Definition 10** (IND-CPA<sup>+</sup>, Li–Micciancio 2021). The adversary may adaptively (1) request encryptions of message pairs  $(m_0, m_1)$ , receiving  $\text{Encrypt}(m_b)$ ; (2) request homomorphic evaluation of any circuit on its ciphertexts; (3) request decryption of an evaluated ciphertext, provided the result is the same under  $b = 0$  and  $b = 1$ . The scheme is secure if  $|\Pr[b' = b] - \frac{1}{2}|$  is negligible.

For an exactly correct scheme (BFV, BGV) the decryption oracle returns a value the adversary already knows, so it can be simulated and  $\text{IND-CPA}^+$  coincides with  $\text{IND-CPA}$ .

## 8.2 The Li–Micciancio Attack

In CKKS a decryption query on  $(c_0, c_1)$  returns  $\hat{\mu} = [c_0 + c_1 \text{SK}]_q = \mu + v$ , the payload *plus the noise*. Hence

$$\text{SK} = (\hat{\mu} - c_0) \cdot c_1^{-1} \quad \text{in } R_q,$$

computed by the extended Euclidean algorithm (Lecture 1); a random  $c_1$  is invertible with high probability. In BFV and BGV the oracle returns  $M$ , and  $v$  remains unknown. One published decryption recovered the key against every major CKKS library.

**Example 7** (Key recovery in miniature). Work in  $R_{17} = \mathbb{Z}_{17}[X]/(X^2 + 1)$ . Since  $4^2 = 16 \equiv -1$ ,  $X^2 + 1 = (X - 4)(X + 4)$  and the ring is not a field. The adversary sees  $c_0 = 6 + 6X$ ,  $c_1 = 2 + X$ ,  $\hat{\mu} = 7 + 2X$ .

- (a)  $c_1(4) = 6$  and  $c_1(-4) = -2 \equiv 15$  are nonzero, so  $c_1$  is invertible. Writing  $(2 + X)(a + bX) = (2a - b) + (a + 2b)X = 1$  gives  $a = -2b$  and  $-5b = 1$ , so  $b = -5^{-1} = -7 \equiv 10$  and  $a = -20 \equiv 14$ :  $c_1^{-1} = 14 + 10X$ . Check:  $(2 + X)(14 + 10X) = 18 + 34X \equiv 1$ .
- (b)  $\hat{\mu} - c_0 = 1 - 4X \equiv 1 + 13X$ , and  $\text{SK} = (1 + 13X)(14 + 10X) = (14 - 130) + (10 + 182)X = -116 + 192X \equiv 3 + 5X$ .
- (c) Check:  $c_1 \text{SK} = (2 + X)(3 + 5X) = 1 + 13X$ , and  $c_0 + c_1 \text{SK} = 7 + 19X \equiv 7 + 2X = \hat{\mu}$ .

## 8.3 The Fix: Noise Flooding

Publish  $\hat{\mu} + e_{\text{flood}}$  with fresh  $e_{\text{flood}}$  large enough that the output is statistically independent of  $v$ . A statistical distance of  $2^{-\lambda}$  needs  $\|e_{\text{flood}}\| \approx 2^\lambda \|v\|$ , costing about  $\lambda$  bits of modulus and of precision. Every current library implements a variant. A client that decrypts locally and keeps the answer is unaffected; a client that returns the decryption to the server is completely broken without it.

## 9. A Verified Instance

This section runs every algorithm once on numbers small enough to check by hand. Throughout, polynomials are written as coefficient vectors (const,  $X$ ,  $X^2$ ,  $X^3$ ), products are taken in  $\mathbb{Z}[X]/(X^4 + 1)$  (so  $X^4 = -1$ ), and  $[\cdot]_q$  is the centred representative in  $(-q/2, q/2]$ : add or subtract  $q$  until the coefficient lies in that range.

### 9.1 Setup and Keys

*Parameters.*  $n = 4$ ,  $\Delta = 10^4$ , one level ( $L = 1$ ),  $q_0 = 10000019$  (prime) and  $q_1 = q_0 \Delta = 100000190000$ , so  $q_1/2 = 50000095000$ . Small polynomials are ternary ( $\beta = 1$ ), and the fresh noise bound is  $2n\beta + \beta = 9$ .

*Step 1: sample.*  $\text{SK} = 1 - X + X^3$ ,  $e = 1 - X^2 + X^3$ , and  $a$  uniform in  $R_{q_1}$ :

|                   | 1           | $X$         | $X^2$       | $X^3$       |
|-------------------|-------------|-------------|-------------|-------------|
| $a = \text{PK}_1$ | 48772087225 | 71480653741 | 58975695645 | 69681953716 |

*Step 2: compute  $a \cdot \text{SK}$ .* With  $X^4 = -1$ , the constant term collects  $a_0 s_0$  and, with a minus sign, the products whose degrees add to 4:

$$(a\text{SK})_0 = a_0 s_0 - a_1 s_3 - a_2 s_2 - a_3 s_1 = 48772087225 - 71480653741 - 0 + 69681953716 = 46973387200.$$

The other coefficients are computed the same way:

$$a\text{SK} = (46973387200, -36267129129, -82186911812, 59478345296).$$

*Step 3: form  $\text{PK}_0 = [-(a\text{SK} + e)]_{q_1}$ .* Adding  $e = (1, 0, -1, 1)$  and negating gives

$$-(a\text{SK} + e) = (-46973387201, 36267129129, 82186911813, -59478345297).$$

The last two entries lie outside  $(-q_1/2, q_1/2]$ , so

$$82186911813 - q_1 = -17813278187, \quad -59478345297 + q_1 = 40521844703,$$

and

|               | 1            | $X$         | $X^2$        | $X^3$       |
|---------------|--------------|-------------|--------------|-------------|
| $\text{PK}_0$ | -46973387201 | 36267129129 | -17813278187 | 40521844703 |

*Check.*  $[\text{PK}_0 + \text{PK}_1 \text{SK}]_{q_1} = (-1, 0, 1, -1) = -e. \checkmark$

## 9.2 Encoding and Encryption

*Step 1: encode.* The payloads are  $z^{(1)} = (1 + 4i, 5 - 2i)$  and  $z^{(2)} = (2 + i, 1 - 3i)$ . Interpolate as in Example 4(b),  $a_k = \frac{1}{2}\text{Re}(z_1 \zeta^{-k} + z_2 \zeta^{-5k})$ ; for instance, for  $z^{(1)}$  and  $k = 1$ ,

$$(1 + 4i)\zeta^7 + (5 - 2i)\zeta^3 = \frac{(1+4i)(1-i)+(5-2i)(-1+i)}{\sqrt{2}} = \frac{(5+3i)+(-3+7i)}{\sqrt{2}} = \frac{2+10i}{\sqrt{2}} = 1.41421 + 7.07107i,$$

so  $a_1 = \frac{1}{2} \cdot 1.41421 = 0.70711$ . In full, then scaling by  $\Delta = 10^4$  and rounding:

|     | $z^{(1)} = (1 + 4i, 5 - 2i)$       |         |              |               | $z^{(2)} = (2 + i, 1 - 3i)$        |         |              |               |
|-----|------------------------------------|---------|--------------|---------------|------------------------------------|---------|--------------|---------------|
| $k$ | $z_1 \zeta^{-k} + z_2 \zeta^{-5k}$ | $a_k$   | $\Delta a_k$ | $\mu_k^{(1)}$ | $z_1 \zeta^{-k} + z_2 \zeta^{-5k}$ | $a_k$   | $\Delta a_k$ | $\mu_k^{(2)}$ |
| 0   | $6 + 2i$                           | 3       | 30000        | 30000         | $3 - 2i$                           | 1.5     | 15000        | 15000         |
| 1   | $1.41421 + 7.07107i$               | 0.70711 | 7071.068     | 7071          | $3.53553 + 2.12132i$               | 1.76777 | 17677.670    | 17678         |
| 2   | $2 - 6i$                           | 1       | 10000        | 10000         | $-2 - 3i$                          | -1      | -10000       | -10000        |
| 3   | $7.07107 - 1.41421i$               | 3.53553 | 35355.339    | 35355         | $2.12132 - 3.53553i$               | 1.06066 | 10606.602    | 10607         |

So  $\mu^{(1)} = 30000 + 7071X + 10000X^2 + 35355X^3$  and  $\mu^{(2)} = 15000 + 17678X - 10000X^2 + 10607X^3$ . Decoding them back gives  $(1.00002 + 3.99997i, 4.99998 - 1.99997i)$  and  $(1.999995 + 1.000052i, 1.000005 - 3.000052i)$ : round-trip errors  $3.5 \times 10^{-5}$  and  $5.2 \times 10^{-5}$ , under  $n/(2\Delta) = 2 \times 10^{-4}$ .

*Step 2: encrypt  $\mu^{(1)}$*  with  $u^{(1)} = X$ ,  $e_1^{(1)} = -1 + X^2 + X^3$ ,  $e_2^{(1)} = 1 + X$ . Multiplying by  $X$  shifts every coefficient up one place, and the top coefficient wraps to the constant with a minus sign:

$$\text{PK}_0 X = (-40521844703, -46973387201, 36267129129, -17813278187),$$

$$\text{PK}_1 X = (-69681953716, 48772087225, 71480653741, 58975695645).$$

Then  $c_0 = [\text{PK}_0 u + e_1 + \mu]_{q_1}$ , coefficient by coefficient:

$$\begin{aligned} 1 : & -40521844703 - 1 + 30000 = -40521814704, \\ X : & -46973387201 + 0 + 7071 = -46973380130, \\ X^2 : & 36267129129 + 1 + 10000 = 36267139130, \\ X^3 : & -17813278187 + 1 + 35355 = -17813242831, \end{aligned}$$

all already in range. And  $c_1 = [\text{PK}_1 u + e_2]_{q_1}$ :

$$\text{PK}_1 u + e_2 = (-69681953715, 48772087226, 71480653741, 58975695645);$$

adding  $q_1$  to the first entry and subtracting it from the last two gives

$$c_1^{(1)} = (30318236285, 48772087226, -28519536259, -41024494355).$$

*Step 3: encrypt  $\mu^{(2)}$  with  $u^{(2)} = -1 - X + X^2$ ,  $e_1^{(2)} = X^2 + X^3$ ,  $e_2^{(2)} = 1 - X + X^2$ . Now  $\text{PK}_0 u = -\text{PK}_0 - \text{PK}_0 X + \text{PK}_0 X^2$ , three shifted copies:*

$$\begin{aligned} \text{PK}_0 u &= (105308510091, -29815586631, -65427238143, 13558562613), \\ \text{PK}_1 u &= (-38065829154, -189934694682, -81684262161, -57176995620). \end{aligned}$$

Adding  $e_1 + \mu^{(2)}$  and  $e_2$ :

$$\begin{aligned} \text{PK}_0 u + e_1 + \mu^{(2)} &= (105308525091, -29815568953, -65427248142, 13558573221), \\ \text{PK}_1 u + e_2 &= (-38065829153, -189934694683, -81684262160, -57176995620). \end{aligned}$$

Subtracting  $q_1$  from the first entry and adding it to the third gives  $c_0^{(2)}$ ; adding  $2q_1$  to the second entry and  $q_1$  to the last two gives  $c_1^{(2)}$ :

|             | 1            | $X$          | $X^2$        | $X^3$        |
|-------------|--------------|--------------|--------------|--------------|
| $c_0^{(1)}$ | -40521814704 | -46973380130 | 36267139130  | -17813242831 |
| $c_1^{(1)}$ | 30318236285  | 48772087226  | -28519536259 | -41024494355 |
| $c_0^{(2)}$ | 5308335091   | -29815568953 | 34572941858  | 13558573221  |
| $c_1^{(2)}$ | -38065829153 | 10065685317  | 18315927840  | 42823194380  |

*Step 4: decrypt and read the noise.* For  $\text{ct}^{(1)}$ ,

$$c_1 \text{SK} = (-59478345296, 46973387200, -36267129130, 17813278189)$$

and

$$c_0 + c_1 \text{SK} = (-100000160000, 7070, 10000, 35358) \xrightarrow{[\cdot]_{q_1}} (30000, 7070, 10000, 35358),$$

adding  $q_1$  to the constant. Subtracting  $\mu^{(1)}$  leaves  $v_1 = (0, -1, 0, 3) = -X + 3X^3$ . For  $\text{ct}^{(2)}$ ,

$$c_1 \text{SK} = (-5308320090, 29815586630, -34572951857, -13558562613),$$

and  $c_0 + c_1 \text{SK} = (15001, 17677, -9999, 10608)$  is already in range; subtracting  $\mu^{(2)}$  leaves  $v_2 = (1, -1, 1, 1)$ . Computing  $-eu + e_1 + e_2 \text{SK}$  directly gives the same  $v_1$  and  $v_2$ , of norms 3 and 1, both under 9.

*Step 5: decode.*

$$\begin{aligned} &\text{DECODE}(30000 + 7070X + 10000X^2 + 35358X^3, 10^4) \\ &= (0.999736 + 4.000113i, 5.000264 - 2.000113i), \end{aligned}$$

slot errors  $2.9 \times 10^{-4}$ . Proposition 5 predicts  $\log_2 10^4 - \log_2 3 = 11.7$  bits, an error near  $2^{-11.7} = 3.0 \times 10^{-4}$ .

### 9.3 Addition and Multiplication

*Add.* Add the ciphertexts part by part and reduce ( $\rightarrow$  marks  $[\cdot]_{q_1}$ :  $+q_1$  on the second entry of the first line,  $-q_1$  on its third, and  $-q_1$  on the second entry of the third line):

$$\begin{aligned} c_0^{(1)} + c_0^{(2)} &= (-35213479613, -76788949083, 70840080988, -4254669610) \\ &\rightarrow (-35213479613, 23211240917, -29160109012, -4254669610), \\ c_1^{(1)} + c_1^{(2)} &= (-7747592868, 58837772543, -10203608419, 1798700025) \\ &\rightarrow (-7747592868, -41162417457, -10203608419, 1798700025). \end{aligned}$$

Decrypting gives  $(45001, 24747, 1, 45966)$ . Subtracting  $\mu^{(1)} + \mu^{(2)} = (45000, 24749, 0, 45962)$  leaves  $v = (1, -2, 1, 4) = v_1 + v_2$ , of norm  $4 = 3 + 1$ . It decodes to  $(2.99969 + 5.000264i, 6.00051 - 5.000064i)$ ; the true sum is  $(3 + 5i, 6 - 5i)$ .

*Multiply, step 1: the tensor product.* Multiply the ciphertexts as polynomials in  $S$  and reduce modulo  $q_1$ :

|                                                           | 1            | $X$         | $X^2$       | $X^3$        |
|-----------------------------------------------------------|--------------|-------------|-------------|--------------|
| $d_0 = [c_0^{(1)} c_0^{(2)}]_{q_1}$                       | 31144251183  | -3475974650 | 24553573339 | -6188396635  |
| $d_1 = [c_0^{(1)} c_1^{(2)} + c_1^{(1)} c_0^{(2)}]_{q_1}$ | 9036789235   | 8506184152  | 25710648358 | -39264480952 |
| $d_2 = [c_1^{(1)} c_1^{(2)}]_{q_1}$                       | -42641930390 | 43837455387 | 43097488569 | 11519972352  |

*Step 2: decrypt with  $(1, \text{SK}, \text{SK}^2)$ .* First, using  $X^4 = -1$  and  $X^6 = -X^2$ ,

$$\text{SK}^2 = (1 - X + X^3)^2 = 1 - 2X + X^2 + 2X^3 - 2X^4 + X^6 = 3 - 2X + 2X^3.$$

Then

$$\begin{aligned} [d_1 \text{SK}]_{q_1} &= (-38733875869, -26241253441, -43531244842, 44061849925), \\ [d_2 \text{SK}^2]_{q_1} &= (7439622760, 30601059803, 18577610229, -36918730862), \end{aligned}$$

and adding  $d_0$  and reducing,

$$[d_0 + d_1 \text{SK} + d_2 \text{SK}^2]_{q_1} = (-150001926, 883831712, -400061274, 954722428).$$

*Step 3: compare with the product of the payloads.* The constant term of  $\mu^{(1)} \mu^{(2)}$  is

$$\begin{aligned} &30000 \cdot 15000 - (7071 \cdot 10607 + 10000 \cdot (-10000) + 35355 \cdot 17678) \\ &= 450000000 - 600007787 = -150007787; \end{aligned}$$

in full,

$$\mu^{(1)} \mu^{(2)} = -150007787 + 883885000X - 400009347X^2 + 954605000X^3.$$

Subtracting it from Step 2 leaves the noise

$$v_{\times} = 5861 - 53288X - 51927X^2 + 117428X^3, \quad \|v_{\times}\| = 117428.$$

At scale  $\Delta^2 = 10^8$  the result decodes to  $(-2.001292 + 8.999928i, -0.998746 - 17.001154i)$ ; the true product is  $z^{(1)} \odot z^{(2)} = (-2 + 9i, -1 - 17i)$ . The product is already correct, at the wrong scale.

**Example 8** (Decomposing the multiplication noise). By Proposition 3,  $v_{\times} = \mu^{(1)} v_2 + \mu^{(2)} v_1 + v_1 v_2$ . In  $\mathbb{Z}[X]/(X^4 + 1)$ :

$$\begin{aligned} \mu^{(1)} v_2 &= 48284 - 68284X - 2426X^2 + 62426X^3, & \|\cdot\| &= 68284, \\ \mu^{(2)} v_1 &= -42427 + 15000X - 49499X^2 + 55000X^3, & \|\cdot\| &= 55000, \\ v_1 v_2 &= 4 - 4X - 2X^2 + 2X^3, & \|\cdot\| &= 4. \end{aligned}$$

The constant terms add to  $48284 - 42427 + 4 = 5861$ , and likewise for the other coefficients, recovering  $v_\times$ . The bound is  $4(35355 \cdot 1 + 17678 \cdot 3) + 4 \cdot 3 \cdot 1 = 353568$ , against the observed 117428. The quadratic term contributes 4; the linear terms, proportional to the payload, contribute everything else. In BGV (Lecture 11) the payload  $M$  is small and the noise carries the factor  $t$ , so  $t^2 e_1 e_2$  dominates; in CKKS the payload carries the large factor  $\Delta$ , so the linear terms dominate.

## 9.4 Relinearize, Then Rescale

*Relinearize.* With  $w = 2$ ,  $\ell_w = \lceil \log_2 100000190000 \rceil = 37$ :  $d_2$  is split into 37 binary digit polynomials, each multiplied by its evaluation key and added to  $(d_0, d_1)$ . The result is the two-part ciphertext  $(\hat{c}_0, \hat{c}_1)$  of Example 5. Decrypting it with SK alone,

$$[\hat{c}_0 + \hat{c}_1 \text{SK}]_{q_1} = (-150001918, 883831728, -400061281, 954722420),$$

and subtracting  $\mu^{(1)}\mu^{(2)}$  leaves the noise  $(5869, -53272, -51934, 117420)$ , of norm 117420. Coefficient by coefficient it moved by 8, 16,  $-7$ ,  $-8$  from  $v_\times$ : at most 16, against the bound  $n\ell_w(w-1)\beta = 4 \cdot 37 \cdot 1 \cdot 1 = 148$ .

*Rescale.* Example 5 divides every coefficient of  $(\hat{c}_0, \hat{c}_1)$  by  $\Delta$  and rounds, giving  $(c'_0, c'_1)$  modulo  $q_0$ . Decrypting,  $[c'_0 + c'_1 \text{SK}]_{q_0} = (-15000, 88383, -40006, 95473)$ , of norm  $95473 < q_0/2$ , with noise  $\|v'\| = 13$ . Decoding at scale  $10^4$ :

$$(-2.001339 + 8.999982i, -0.998661 - 17.001182i),$$

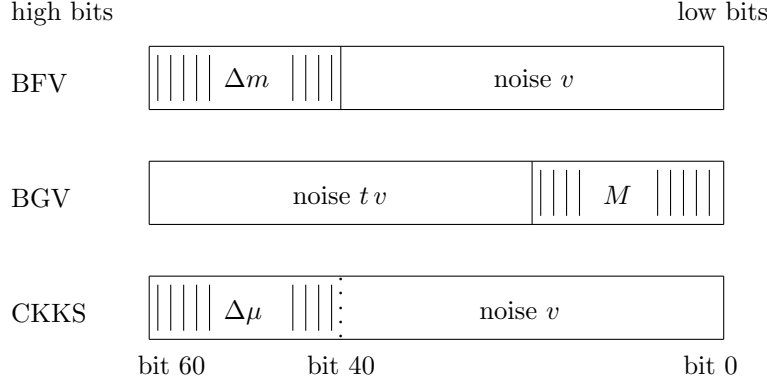
slot errors  $1.34 \times 10^{-3}$  and  $1.79 \times 10^{-3}$ . The prediction is  $\log_2 10^4 - \log_2 13 = 13.29 - 3.70 = 9.59$  bits; observed  $-\log_2(1.34 \times 10^{-3}) = 9.54$  and  $-\log_2(1.79 \times 10^{-3}) = 9.13$ .

| Stage             | modulus | scale  | $\ v\ $ |
|-------------------|---------|--------|---------|
| fresh             | $q_1$   | $10^4$ | 3, 1    |
| after EvalAdd     | $q_1$   | $10^4$ | 4       |
| after EvalMult    | $q_1$   | $10^8$ | 117428  |
| after Relinearize | $q_1$   | $10^8$ | 117420  |
| after RESCALE     | $q_0$   | $10^4$ | 13      |

In the last step modulus, scale and noise all fall by a factor  $\Delta$ : precision is unchanged, and the next multiplication starts from scale  $10^4$ , not  $10^8$ .

## 10. Where the Message Sits

Draw one ciphertext coefficient as a 60-bit word, with  $t = 2^{20}$  and  $\Delta = 2^{40}$ ; hatching marks the message.



BFV places the message in the top twenty bits and BGV in the bottom twenty; each draws a boundary inside the word and defends it. CKKS places  $\Delta\mu$  at the top with no boundary below (dotted): payload and noise form one block and cannot be separated after encryption.

|                      | <b>BFV</b>       | <b>BGV</b>              | <b>CKKS</b>                             |
|----------------------|------------------|-------------------------|-----------------------------------------|
| Plaintext            | exact, in $R_t$  | exact, in $R_t$         | approximate, in $\mathbb{C}^{n/2}$      |
| $c_0 + c_1\text{SK}$ | $\Delta M + v$   | $M + t v$               | $\mu + v$                               |
| $\Delta$ applied at  | encryption       | —                       | <i>encoding</i>                         |
| Noise removed by     | rounding         | mod $t$                 | <i>not removed</i>                      |
| Correct when         | $\ v\  < q/(2t)$ | $\ v\  < q_\ell/(2t)$   | never exactly; $\ \mu + v\  < q_\ell/2$ |
| Level control        | in EvalMult      | MODSWITCH               | RESCALE                                 |
| Chain constraint     | —                | $p_i \equiv 1 \pmod{t}$ | none                                    |

In CKKS the noise is never removed; it costs  $\log_2\|v\|$  bits of precision.

## Summary

1. CKKS treats the RLWE noise as fixed-point rounding error:  $\text{ENCODE}(z, \Delta) = \lfloor \Delta \tau^{-1}(z) \rfloor$  via the canonical embedding, with  $n/2$  complex slots and encoding error at most  $n/(2\Delta)$  per slot.
2. KeyGen and Encrypt are BFV's without  $\Delta$  on the message and without  $t$ ; Decrypt is  $[c_0 + c_1\text{SK}]_{q_\ell} = \mu + v$ , correct only up to the noise and only while  $\|\mu + v\| < q_\ell/2$ .
3. EvalMult squares the scale; its noise is dominated by the linear terms  $\mu_1 v_2 + \mu_2 v_1$ . Relinearize is BFV's.
4. RESCALE =  $\lfloor \Delta^{-1} \text{ct} \rfloor$  drops one level of  $q_\ell = q_0 \Delta^\ell$ ; it is BGV's MODSWITCH formula without the congruence. The budget is  $\log q > L \log \Delta$ , and precision is about  $\log_2 \Delta - \log_2 \|v\|$  bits.
5. Vanilla CKKS fails IND-CPA<sup>+</sup>: one published decryption gives  $\text{SK} = (\hat{\mu} - c_0)c_1^{-1}$ . The fix is noise flooding.

## Exercises

1. Repeat Example 1(d) in  $Q_{4,12}$  ( $\Delta = 2^{12}$ ) with the same errors  $e_x = 3$ ,  $e_y = -2$ . How many fractional bits survive? State the rule relating this to  $f$  and to the size of the errors.
2. At  $n = 4$ ,  $\Delta = 10^2$ , encode  $z' = (2, 3)$  and multiply the result by  $\mu = 300 + 71X + 100X^2 + 354X^3$ , which encodes  $(1 + 4i, 5 - 2i)$  at the same  $\Delta$ , in  $\mathbb{Z}[X]/(X^4 + 1)$ . Decode at scale  $\Delta^2$

and check that you obtain approximately  $(2(1 + 4i), 3(5 - 2i))$ . Which property of  $\tau$  makes this work?

3. In the instance of Section 9, encrypt the constant  $\mu = 10000$  with  $u = 1$ ,  $e_1 = e_2 = 0$ . Compute  $[c_0 + c_1\text{SK}]_{q_1}$  and the noise  $v$  two ways. Why is  $v \neq 0$ ?
4. Take  $c = -6564956194$  from Example 5. Round  $\Delta^{-1}c$  instead to the nearest integer congruent to  $c$  modulo 8, as BGV with  $t = 8$  would. How large is the correction now, and why does CKKS not need this?
5. Song evaluates  $F(x) = \pi x^3 + 0.4x + 1$  on an encrypted  $x$ : square  $x$  and rescale; compute  $\pi x$  by a plaintext–ciphertext product and rescale; multiply the two and rescale; add the remaining terms. Starting from  $(\text{level}, \text{scale}) = (2, \Delta)$ , list the pair for every intermediate ciphertext. Where must two ciphertexts at different levels be aligned before adding?
6. For Song’s parameter set, how many further 40-bit levels fit under the ceiling of 218 bits? A user needs 40 bits of precision; give two ways to obtain it and the cost of each.
7. In Example 7, suppose the oracle returned only  $[\hat{\mu}]_4$ , as an exact scheme with  $t = 4$  would. Explain why the equation for SK no longer determines it.