

The CKKS Scheme

Lecture 12 — Approximate Arithmetic

SEC 595 — Encrypted Computing

- 1 Noise as Rounding Error
- 2 Encoding and Decoding
- 3 Keys, Encryption, Decryption
- 4 EvalAdd, EvalMult, Relinearization
- 5 RESCALE
- 6 Levels
- 7 Precision
- 8 Security
- 9 A Verified Instance
- 10 Where the Message Sits

BFV (Lecture 10) and BGV (Lecture 11) treat the RLWE error as a tax, kept below a threshold so decryption is **exact**.

CKKS (Cheon–Kim–Kim–Song, 2017)

The noise an RLWE scheme must add for security is the same kind of object as the rounding error of fixed-point arithmetic. Stop removing it: declare it part of the answer.

The price: only *approximate* correctness,

$$\text{Decrypt}(\text{Encrypt}(\mu)) \approx \mu,$$

$$\text{Decrypt}(ct_1 * ct_2) \approx \text{Decrypt}(ct_1) * \text{Decrypt}(ct_2).$$

Ring, keys, relinearization and modulus chain are inherited almost unchanged.

Lecture 1: a real x is stored as $\hat{x} = \lfloor \Delta x \rfloor$, with $\Delta = 2^f$.

addition: $\hat{x} + \hat{y} \approx \Delta(x + y)$ scale unchanged

multiplication: $\hat{x} \hat{y} \approx \Delta^2(xy)$ scale *squared*

To restore the scale, **rescale**: divide by Δ and round.

Two facts the scheme is built on

- A. A large Δ makes a small error harmless: with $\|e\| = 2^k$ and scale 2^f , about $f - k$ bits survive.
- B. Multiplication squares the scale; something must undo it.

Example — the $Q_{4.4}$ round trip

Work in $Q_{4.4}$, so $\Delta = 16$. Take $x = 2.75$, $y = 3.25$.

- a. Encode both. What are $\hat{x}$ and $\hat{y}$?
- b. Multiply. What is the scale of the product?
- c. Rescale. What integer do you get, and what real number does it represent?
- d. Now add errors $e_x = 3$ and $e_y = -2$ before multiplying. How many bits of the answer survive?

Why not put the numbers in the coefficients?

The user has a vector of numbers; CKKS encrypts a polynomial.

Coefficients add correctly, but they multiply by **convolution**:

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

Goal

A representation in which *one* polynomial product performs $n/2$ numeric products, entry by entry.

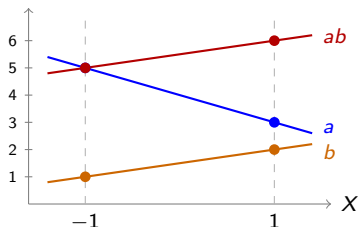
Store the numbers at points

Evaluation at a fixed point respects both operations:

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

Points $w = \pm 1$, modulo $X^2 - 1$ (first value at 1):

- $(3, 5) \mapsto a = 4 - X$
- $(2, 1) \mapsto b = \frac{3}{2} + \frac{1}{2}X$
- $ab \equiv \frac{11}{2} + \frac{1}{2}X$
- $(ab)(1) = 6, (ab)(-1) = 5$

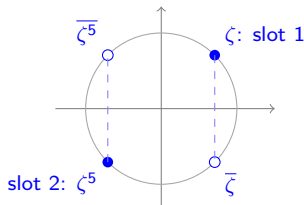


Working modulo a polynomial that vanishes at the points changes nothing there.

The points of CKKS: slots

Modulo $X^n + 1$ the points are its roots:
the odd powers of $\zeta = e^{\pi i/n}$, on the unit
circle.

Real coefficients: $a(\overline{w}) = \overline{a(w)}$. A root
fixes its mirror image, so only $n/2$ values
are free: the **slots**.



Canonical embedding

$$\tau(a) = (a(\zeta), a(\zeta^5), a(\zeta^{25}), \dots) \in \mathbb{C}^{n/2}$$

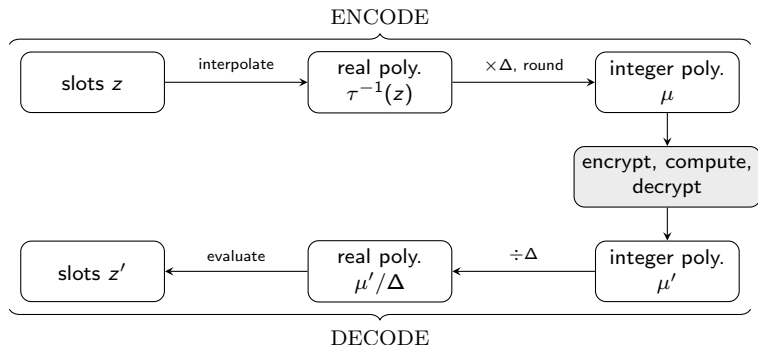
A bijection, linear and multiplicative: ring products are slotwise products.
At $n = 2^{13}$: 4096 slots. $X \mapsto X^5$ rotates them.

Example — the canonical embedding

$n = 4$: the roots of $X^4 + 1$ are $\zeta = \frac{1+i}{\sqrt{2}}$, $\zeta^3 = \frac{-1+i}{\sqrt{2}}$, $\zeta^5 = \frac{-1-i}{\sqrt{2}}$, $\zeta^7 = \frac{1-i}{\sqrt{2}}$.
Let $a(X) = 1 + X + X^3$: any real polynomial would do; this one has clean values.

- a. Evaluate a at all four roots.
- b. Which values are mirror images of each other?
- c. Write $\tau(a)$.
- d. Let $b(X) = X^2$. Compute $\tau(b)$ and $ab \bmod (X^4 + 1)$, and check that $\tau(ab)$ is the slotwise product.

Here τ runs forward, polynomial $\rightarrow$ slots. ENCODE runs it backwards: interpolating $(1 + \sqrt{2}i, 1 - \sqrt{2}i)$ returns a .



$$\text{ENCODE}(z, \Delta) = \lfloor \Delta \cdot \tau^{-1}(z) \rfloor, \quad \text{DECODE}(a, \Delta) = \tau(a/\Delta).$$

Lecture 1's fixed point, after a change of basis. For $n = 4$:

$$a_k = \frac{1}{2} \text{Re}(z_1 \zeta^{-k} + z_2 \zeta^{-5k}).$$

Example — encoding a real vector

Take $n = 4$, $\Delta = 100$, $z = (2, 1)$. Same roots as before. Reverse the canonical embedding:

- a. Place the wanted values at all four roots.
- b. Interpolate with $a_k = \frac{1}{2}\text{Re}(z_1\zeta^{-k} + z_2\zeta^{-5k})$.
- c. Check: evaluate $\tau^{-1}(z)$ at every root.
- d. Scale by Δ and round to get μ .
- e. Decode μ . How far is each slot from z ?
- f. Compute $\mu^2 \bmod (X^4 + 1)$ and decode it at scale Δ^2 .

The encoding error

Rounding moves each coefficient by at most $1/2$, so each slot by at most

$$\frac{n}{2\Delta}.$$

Encoding $z = (2, 1)$ with $n = 4$:

Δ	$\mu(X)$	slot error
10^2	$150 + 35X - 35X^3$	5.0×10^{-3}
10^3	$1500 + 354X - 354X^3$	6.3×10^{-4}

Δ wants to be *large* (precision); the modulus budget wants it *small*. That tension is the design of the scheme.

$\mathbb{C}^{n/2} \xrightarrow{\text{ENCODE}} R = \mathbb{Z}[X]/(X^n + 1)$; ciphertexts in $R_{q_\ell}^2$. **There is no plaintext modulus t .**

Definition

$\text{SK}, u \leftarrow R_2$; $a \leftarrow R_{q_L}$; $e, e_1, e_2 \leftarrow \mathcal{X}$.

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

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

$$\text{BFV: } c_0 = \text{PK}_0 u + e_1 + \Delta M \quad \text{BGV: } c_0 = \text{PK}_0 u + t e_1 + M$$

The only change from BFV: no Δ on the message — ENCODE already applied it.

Definition

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

No rounding, no division, no reduction mod t : Decrypt is the bare evaluation $\text{ct}(\text{SK})$.

Proposition

For a fresh ciphertext,

$$c_0 + c_1\text{SK} = \mu + v \pmod{q_\ell}, \quad v = -eu + e_1 + e_2\text{SK}.$$

The mask $au\text{SK}$ cancels as in BFV. The noise is BFV's, so $\|v\| \leq 2n\beta + \beta$ — and nothing removes it.

Theorem

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 .

Correctness is only about *wraparound*: there is no threshold below which the answer becomes exact.

Data range: $\|z\| < q_\ell/(2\Delta)$. At $q_0 = 2^{60}$, $\Delta = 2^{40}$, values up to about 2^{20} . Overflow is silent.

Definition

For ciphertexts at the same level ℓ and the same scale Δ ,

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

It decrypts to $\mu_1 + \mu_2 + (v_1 + v_2)$: $\|v_{\text{add}}\| \leq \|v_1\| + \|v_2\|$, scale unchanged, no level consumed.

New condition: $\Delta z + \Delta' z'$ decodes to nothing meaningful unless $\Delta = \Delta'$.
BGV needed level alignment; CKKS also needs *scale* alignment.

Definition

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

The tensor product of Lecture 4, identical in BFV, BGV and CKKS.

What differs is the plaintext arithmetic:

$$(\Delta\mu_1 + e_1)(\Delta\mu_2 + e_2) = \underbrace{\Delta^2\mu_1\mu_2}_{\text{payload}} + \Delta(\mu_1e_2 + \mu_2e_1) + e_1e_2.$$

$$\text{BGV: } (te_1 + \mu_1)(te_2 + \mu_2) = t^2e_1e_2 + t(e_1\mu_2 + e_2\mu_1) + \underbrace{\mu_1\mu_2}_{\text{payload}}.$$

The scale becomes Δ^2 . A plaintext–ciphertext product $(\mu'c_0, \mu'c_1)$ needs no relinearization but also squares the scale.

Proposition

If $\text{ct}^{(i)}(\text{SK}) = \mu_i + v_i$, the product decrypts to $\mu_1\mu_2 + v_\times$ with

$$v_\times = \mu_1 v_2 + \mu_2 v_1 + v_1 v_2,$$

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

- BGV: the dangerous term is $t^2 e_1 e_2$ — quadratic in the noise.
- CKKS: $\mu_i = \Delta z_i$ is huge beside v_i , so the *linear* terms dominate.

Relative to Δ^2 , a multiplication costs a bit or two of precision and does not compound.

EK and Relinearize

$$\begin{aligned} \text{EK} &= ([-(a\text{SK} + e) + \text{SK}^2]_{q_\ell}, a), & \text{EK}_0 + \text{EK}_1\text{SK} &= \text{SK}^2 - e, \\ \hat{c}_0 &= c_0^* + \text{EK}_0 c_2^*, & \hat{c}_1 &= c_1^* + \text{EK}_1 c_2^*. \end{aligned}$$

BFV's construction, unchanged. BGV needed a factor t in EK to protect the residue mod t ; CKKS has no residue to protect.

With gadget base w and $\ell_w = \lceil \log_w q_\ell \rceil$ (Lecture 4), the added noise is additive:

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

Definition

For $ct \in R_{q_\ell}^2$ at scale Δ^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 Δ .

$$ct'(\text{SK}) \approx \Delta^{-1}(\Delta^2 \mu + v) = \Delta \mu + \Delta^{-1} v$$

Same payload at scale Δ ; noise divided by Δ . The rounding adds at most $\frac{1}{2}(1 + n\|\text{SK}\|)$.

	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'

No congruence is needed because CKKS never reduces the plaintext modulo anything: there is no residue to disturb.

Same formula, different disease: BGV's *noise* explodes; CKKS's *scale* does.

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:

- Without RESCALE: d multiplications leave scale Δ^{2^d} , so $\log q > 2^d \log \Delta$.
- With RESCALE: the scale returns to Δ each time, so $\log q > L \log \Delta$.

Exponential $\longrightarrow$ linear. That is Fact B.

Example — a rescale trace

Level 1, $q_1 = 100000190000$, scale $\Delta^2 = 10^8$:

$$\hat{c}_0 = -6564956194 + 25551957201X + 21920360407X^2 + 28669366452X^3,$$

$$\hat{c}_1 = -9461472835 + 40573592885X - 25296998807X^2 - 43550170004X^3.$$

With $\Delta = 10^4$ and $SK = 1 - X + X^3$:

- a. Rescale. What are c'_0 , c'_1 and the new modulus?
- b. Tabulate the rounding corrections $c' - \Delta^{-1}c$. How large can they be?
- c. Evaluate at SK modulo q_0 . Compare $\|v'\|$ with $\|v\|/\Delta$ before the switch.

Definition

Fix a base modulus $q_0 \gg \Delta$ and a depth L :

$$q_\ell = q_0 \cdot \Delta^\ell \quad (0 \leq \ell \leq L), \quad q = q_L.$$

A ciphertext with modulus q_ℓ is at **level** ℓ .

Operation	in	out	costs
EvalAdd	(ℓ, Δ)	(ℓ, Δ)	nothing
EvalMult	(ℓ, Δ)	(ℓ, Δ^2)	a third component
Relinearize	(ℓ, Δ^2)	(ℓ, Δ^2)	a little noise
RESCALE	(ℓ, Δ^2)	$(\ell - 1, \Delta)$	one level

Level is computational capability. At level 0 only addition and decryption remain: there is no room for a squared scale.

$$\boxed{\log q > L \cdot \log \Delta} \quad (\text{slack } \log q_0)$$

For fixed security, a larger $\log q$ forces a larger n , and every operation gets slower. Depth, precision and speed are one resource.

	depth	modulus
$(xy)(zw)$, balanced tree	2	$q_0 \Delta^2$
$((xy)z)w$, chain	3	$q_0 \Delta^3$

Same three multiplications. Compiling for CKKS means minimising depth, not operation count.

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: keys live at $q_0 q_1 q_2 q_3$ so 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 divides by q_ℓ , so the scale drifts: $\Delta' = \Delta^2/q_\ell \approx \Delta$.

Proposition

A ciphertext at scale Δ with noise v decodes to slots correct to about

$$\log_2 \Delta - \log_2 \|v\| \text{ bits.}$$

The slot error is about $\|v\|/\Delta$, by the correctness theorem.

Song: $\log_2 \|v\| \approx 10\text{--}15$ for basic operations, so $\log_2 \Delta = 40$ gives 25–30 bits.

Error analysis of a CKKS circuit is the error analysis of the same circuit in floating point.

Example — a precision calculation

- a. At $\Delta = 2^{40}$ and $\|v\| = 2^{12}$, how many bits survive? How many decimal digits?
- b. A ciphertext at scale $\Delta = 10^4$ has $\|v'\| = 13$. Predict the slot error.
- c. Using $\|v_{\times}\| \approx n\|\mu\|\|v\|$, show that one multiplication leaves about $\log_2 (\Delta / (n\|z\|\|v\|))$ bits.
- d. With $\|v\| \approx 2^{10}$ and $n\|z\| \approx 2^{16}$, what Δ gives 30 bits after one multiplication?

PK is an RLWE sample; IND-CPA (Lecture 8, §1.5) follows from decision RLWE as for BFV and BGV.

IND-CPA⁺ (Li-Micciancio, 2021)

The adversary may adaptively

- 1 request encryptions of message pairs, receiving $\text{Encrypt}(m_b)$;
- 2 request evaluation of any circuit on its ciphertexts;
- 3 request *decryption* of an evaluated ciphertext, provided the result is the same under $b = 0$ and $b = 1$.

Secure if $|\Pr[b' = b] - \frac{1}{2}|$ is negligible.

This models a client that decrypts and publishes the result. For an exactly correct scheme (BFV, BGV) the decryption oracle is simulable, so $\text{IND-CPA}^+ = \text{IND-CPA}$.

The Li–Micciancio attack

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**. So

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

computed by the extended Euclidean algorithm (Lecture 1).

In BFV and BGV the oracle returns M , and v stays unknown. CKKS hands over $c_0 + c_1 \text{SK}$ itself.

One published decryption recovered the key against every major CKKS library.

Example — key recovery in miniature

Work in $R_{17} = \mathbb{Z}_{17}[X]/(X^2 + 1)$. Since $4^2 \equiv -1$, this ring is *not* a field.

The adversary sees

$$c_0 = 6 + 6X, \quad c_1 = 2 + X, \quad \hat{\mu} = 7 + 2X.$$

- a. Check that c_1 is invertible, and compute c_1^{-1} .
- b. Recover SK.
- c. Verify your answer against the ciphertext.

The fix: noise flooding

Publish $\hat{\mu} + e_{\text{flood}}$, with fresh e_{flood} large enough that the output is statistically independent of v .

- Statistical distance $2^{-\lambda}$ needs $\|e_{\text{flood}}\| \approx 2^\lambda \|v\|$.
- Cost: about λ bits of modulus and of precision.
- Every current library implements a variant.

Scope: a client that decrypts locally and keeps the answer is unaffected. A client that returns the decryption to the server is completely broken.

$$n = 4, \quad \Delta = 10^4, \quad L = 1, \quad q_0 = 10000019 \text{ (prime),}$$

$$q_1 = q_0 \Delta = 100000190000.$$

Small polynomials ternary ($\beta = 1$, fresh bound $2n\beta + \beta = 9$);

$$\text{SK} = 1 - X + X^3, \quad e = 1 - X^2 + X^3.$$

	1	X	X^2	X^3
$a = \text{PK}_1$	48772087225	71480653741	58975695645	69681953716
PK_0	-46973387201	36267129129	-17813278187	40521844703

$$\text{PK}_0 + \text{PK}_1 \text{SK} = -1 + X^2 - X^3 = -e$$

Level-0 data range $q_0/\Delta \approx 10^3$; wraparound budget $q_0/2 = 5000009$.

$$z^{(1)} = (1 + 4i, 5 - 2i), \quad z^{(2)} = (2 + i, 1 - 3i)$$

$$\Delta\tau^{-1}(z^{(1)}) = (30000, 7071.068, 10000, 35355.339),$$

$$\mu^{(1)} = 30000 + 7071X + 10000X^2 + 35355X^3,$$

$$\Delta\tau^{-1}(z^{(2)}) = (15000, 17677.670, -10000, 10606.602),$$

$$\mu^{(2)} = 15000 + 17678X - 10000X^2 + 10607X^3.$$

Round-trip slot errors 3.5×10^{-5} and 5.2×10^{-5} , under $n/(2\Delta) = 2 \times 10^{-4}$.

	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

$$v_1 = -X + 3X^3, \quad \|v_1\| = 3, \quad v_2 = 1 - X + X^2 + X^3, \quad \|v_2\| = 1$$

$\text{DECODE}(\text{Decrypt}(\text{ct}^{(1)})) =$
 $(0.999736 + 4.000113i, 5.000264 - 2.000113i)$: slot errors 2.9×10^{-4} .

Predicted: $\log_2 10^4 - \log_2 3 = 11.7$ bits, an error near 3.0×10^{-4} .

Add. $v_1 + v_2 = 1 - 2X + X^2 + 4X^3$, norm $4 = 3 + 1$. Decodes to $(2.99969 + 5.000264i, 6.00051 - 5.000064i)$; true $(3 + 5i, 6 - 5i)$.

Multiply.

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

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

At scale $\Delta^2 = 10^8$ this decodes to

$$(-2.001292 + 8.999928i, -0.998746 - 17.001154i);$$

$$\text{true } z^{(1)} \odot z^{(2)} = (-2 + 9i, -1 - 17i).$$

The product is already correct — at the wrong scale.

Example — decompose the multiplication noise

On the instance, $v_{\times} = \mu^{(1)}v_2 + \mu^{(2)}v_1 + v_1v_2$.

- a. Compute the three products in $\mathbb{Z}[X]/(X^4 + 1)$ and check they sum to $v_{\times}$.
- b. What is the norm of each? Which dominates?
- c. Compare with the bound $n(\|\mu_1\| \|v_2\| + \|\mu_2\| \|v_1\|) + n\|v_1\| \|v_2\|$.
- d. In BGV (Lecture 11) the quadratic term dominated. Explain the reversal.

$w = 2$, $\ell_w = 37$: after Relinearize, $\|v\| = 117420$, a change of 16 against the bound $n\ell_w(w-1)\beta = 148$.

Rescaling by Δ (every correction $\leq 1/2$) gives, in R_{q_0} ,

$$c'_0 = -656496 + 2555196X + 2192036X^2 + 2866937X^3,$$

$$c'_1 = -946147 + 4057359X - 2529700X^2 - 4355017X^3,$$

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

against $117420/10^4 = 11.74$: the extra is rounding, within $\frac{1}{2}(1 + n\|SK\|) = 2.5$.

DECODE = $(-2.001339 + 8.999982i, -0.998661 - 17.001182i)$, slot errors 1.34×10^{-3} and 1.79×10^{-3} .

Predicted: $13.29 - 3.70 = 9.59$ bits; observed 9.54 and 9.13.

The whole instance

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 Δ : precision is unchanged, and the next multiplication starts from 10^4 , not 10^8 .

Which bits are the message?

high bits

low bits

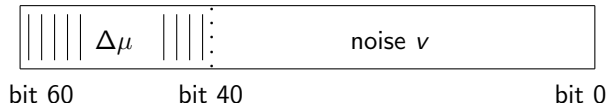
BFV



BGV



CKKS



A 60-bit coefficient with $t = 2^{20}$, $\Delta = 2^{40}$; hatching marks the message. BFV: top twenty bits. BGV: bottom twenty. CKKS: top, with no boundary below — payload and noise are one block.

	BFV	BGV	CKKS
Plaintext	exact, in R_t	exact, in R_t	approximate, in $\mathbb{C}^{n/2}$
$c_0 + c_1 SK$	$\Delta M + v$	$M + t v$	$\mu + v$
Δ 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

BFV and BGV draw a boundary inside the word and defend it. CKKS draws none: the noise simply costs $\log_2 \|v\|$ bits of precision.

- 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, $n/2$ complex slots, encoding error $\leq n/(2\Delta)$.
- ② KeyGen and Encrypt are BFV's without Δ or t ; Decrypt is $[c_0 + c_1 \text{SK}]_{q_\ell} = \mu + v$, correct only up to the noise.
- ③ EvalMult squares the scale; 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 $q_\ell = q_0 \Delta^\ell$; budget $\log q > L \log \Delta$; precision $\approx \log_2 \Delta - \log_2 \|v\|$ bits.
- ⑤ Vanilla CKKS fails IND-CPA⁺: one published decryption gives $\text{SK} = (\hat{\mu} - c_0) c_1^{-1}$. Fix: noise flooding.