FHE for SIMD Arithmetic Logic Units with $O(1)$ Bootstrapping

Hongren Zheng

Tsinghua University

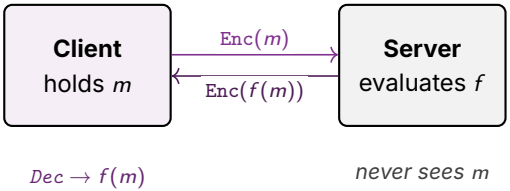
Based on Mingyu Gao, Hongren Zheng (CRYPTO'26)
and Collaboration with Yuchen Wei (ePrint/2026/1836)

Table of Contents

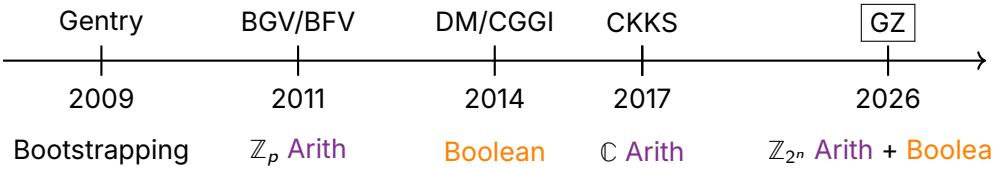
Fully Homomorphic Encryption

$$f(\mathtt{Enc}(m)) \quad = \quad \mathtt{Enc}(f(m))$$

- Computation on encrypted data
- Client: Encrypt data m
- Server: Evaluate function f
- If f is arbitrary, it is “fully”!

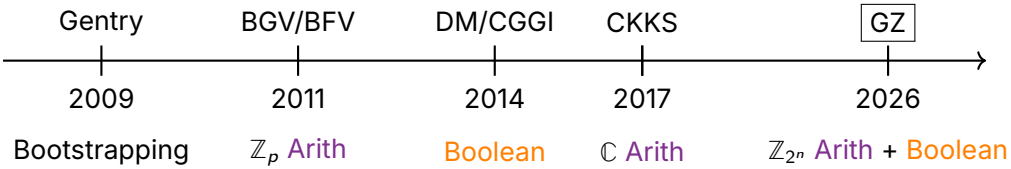


Timeline: the Holy Grail of Cryptography



- FHE was an open problem since 1970s
- 2009: Craig Gentry, the first FHE scheme in his PhD thesis
  - Proposed the “bootstrapping” technique
- Theoretically, FHE can compute everything since 2009
- But practical efficiency is still a huge concern
- The key problem: How to represent the application circuit f?

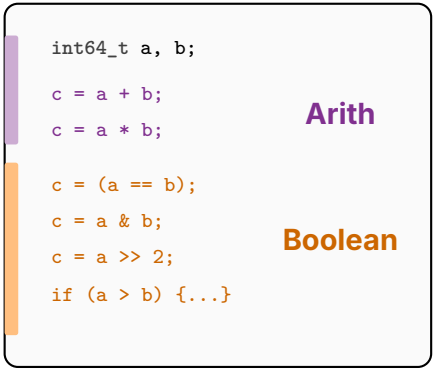
Timeline: the Holy Grail of Cryptography



- Existing schemes only excel in one type of circuit
- BGV/BFV: Typically $\mathbb{Z}_p$ arithmetic circuit
  - Add and Mult, p is prime
- DM/CGGI: bit-level operations, but $O(n^2)$ PBS for arithmetic Mult
  - Suitable for Boolean circuits, but expensive for arithmetic tasks
- CKKS: approximate $\mathbb{C}$ arithmetic, but no exact comparison
  - Suitable for AI/LLM, but not for general-purpose computing

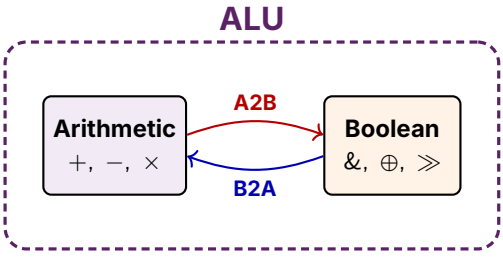
Real-World Programming Model

- The basic types for programmers
- Integers: `int64_t`
- Arithmetic: `+`, `-`, `*`
- Boolean: `==`, `&`, `>>`
- Cryptographical examples:
- ECDSA: $\mathbb{F}_p$ Arith + Scalar Mult
- ML-DSA: R_q Arith + HighBits
- Routine in plaintext programs
- Difficult under FHE



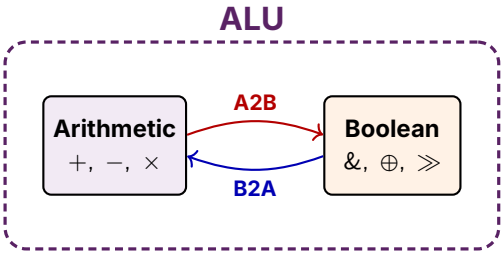
The Challenge: ALU in FHE

- ALU: Arithmetic Logic Unit
  - Terminology from hardware
  - CPU-like
  - Arithmetic mode
  - Boolean mode
  - Conversion between them

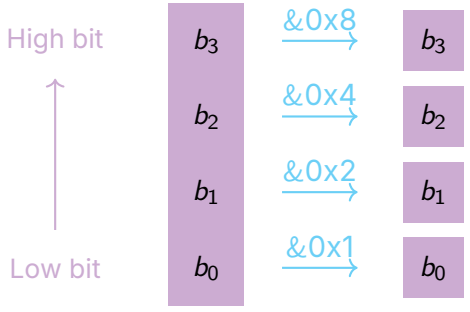


The Challenge: ALU in FHE

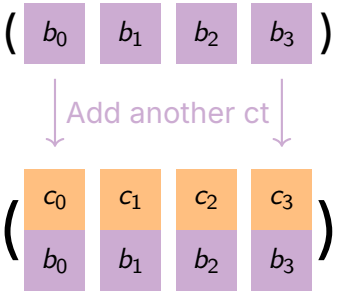
- ALU: Arithmetic Logic Unit
  - Terminology from hardware
  - CPU-like
  - Arithmetic mode
  - Boolean mode
  - Conversion between them
- Prior: TFHE or scheme-switching
- Problem: costly and inconvenient
- Recent: bootstrapping-intensive
- REFHE (EC26): $O(n)$ PBS
- CPL (EC26): $O(\log k)$ CKKS BTS
- Reason: They are *iterative*



Why iterative? Two Extremes of the Spectrum



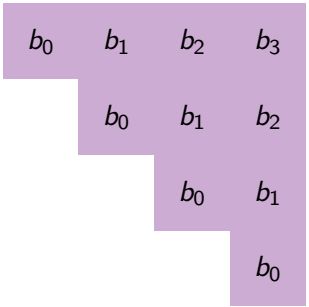
(a) "Whole" (BGV/BFV/CKKS)



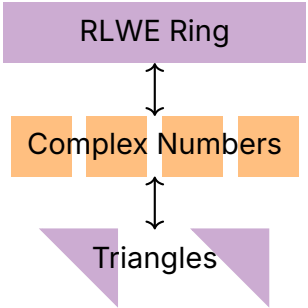
(b) "Bit Vector" (DM/CGGI)

- Encode message in "Whole": Hard to extract bits
- Encode message in "Bit Vector": Need to propagate carries
- Takeaway: Arithmetic and Boolean need incompatible encodings

Our Techniques (Visual)



(a) “Triangle” Encoding



(b) SIMD Homomorphism Chain

- We propose a new encoding method: *“Triangle” Encoding*
  - Balance between “Whole” and “Bit Vector”
- We enable SIMD in a new way: *Homomorphism Chain*

Main Results (Theory)

Scheme	A→A	B→B	A→B	B→A
TFHE	$O(n)$ or $O(n^2)$ PBS	$O(n)$ PBS	-	-
REFHE	Leveled w/ $O(n)$ PBS	$O(n)$ PBS	-	-
CPL	$O(\log k)$ BTS	$O(1)$ BTS	$O(\log k)$	Free
GZ	Leveled w/ $O(1)$ BTS	$O(1)$ BTS	$O(1)$ BTS	Lightweight

- We enable all mode operations in $O(1)$ bootstrapping
- A→B takes extra $O(\log n)$ *levels* (n is bit-width) for carry propagation
  - Prior "iterative" BTS $\longrightarrow$ "iterative" level consumption
- "Leveled" A→A: We enable 2 or 3 Mult before one bootstrapping

Main Results (Practice)

Ring Dim N	**BitWidth** n	**# Slots**	**A$\to$ A**	**A$\to$B**
2^{16}	64	1024	18s / 17ms	49s / 48ms

- Experiments measured under single-threaded CPU setting
- A$\to$A amounts to 2 CKKS BTS (A2A-I + A2A-e)
- A$\to$B amounts to 3 CKKS BTS (A2A-I + A2B-MV-FBT + LCtoC)
- Other bitwidth n satisfies $\#\text{slots} = N/n$

Section 1

Backgrounds: RLWE and its Homomorphic Operations

RLWE Polynomial Ring

$$\mathbb{Z}[X] \Big/ \left\langle X^N + 1 \right\rangle$$

- We call it RLWE ring
- $\mathbb{Z}[X]$: integer polynomials
  - Example: $f(X) = 3X^2 + 5$
- $X^N + 1$: power-of-two cyclotomic polynomial, where N is power-of-two
  - Example: $X^4 + 1$
- Multiplication in this ring is done modulo $X^N + 1$
  - Example: $X^4 \equiv -1$, so $X^5 \equiv -X$, etc.

From LWE to RLWE

compute $b = -as + m + e$ and output (b, a)

- LWE: Learning With Errors: Encrypt $m \in \mathbb{Z}_q$
- Secret $s \in \mathbb{Z}_q^n$, random $a \in \mathbb{Z}_q^n$, small noise $e \in \mathbb{Z}_q$

- $\mathcal{R} = \mathbb{Z}[X]/\langle X^N + 1 \rangle$, N is power-of-two, use $\mathcal{R}_Q$ to denote $\mathcal{R}/Q\mathcal{R}$
- RLWE: Ring Learning With Errors: Encrypt $m \in \mathcal{R}_Q$
- Secret $s \in \mathcal{R}_Q$, random $a \in \mathcal{R}_Q$, noise $e \in \mathcal{R}_Q$
- Main difference: LWE encrypts one number, RLWE encrypts a polynomial

Homomorphic Operations from RLWE

- Ciphertext addition: $(b, a) + (b', a') = (b + b', a + a')$
- Encrypted message becomes $\boxed{m + m'} \in \mathcal{R}_Q$

- Ciphertext tensor product: $(b, a) \otimes (b', a') = (bb', ba' + ab', aa')$
  - Decrypt using $(1, s, s^2)$
- Encrypted message becomes $\boxed{m \cdot m'} \in \mathcal{R}_Q$

- Through Add/Tensor Product in the outer ciphertext
- We have Add/Mult for the inner message m in $\mathcal{R}_Q$
- This is homomorphic!

Let's begin with goals

We have Add/Mult in $\mathcal{R}_Q = \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$

How can we achieve the following goals?

1. Arithmetic + Boolean
   - Arithmetic mode: $\mathcal{R}_Q \xleftarrow{?} \mathbb{Z}_{2^n}$ (Section 2)
   - Boolean mode: $\mathcal{R}_Q \xleftarrow{?} \{0,1\}^n$ (Section 5)
2. SIMD
   - Induce SIMD structure: $\mathcal{R}_Q \xleftarrow{?}$ many $\mathbb{Z}_{2^n}$ or $\{0,1\}^n$ slots (Section 3)
3. Efficient Bootstrapping
   - *Refreshing noise* inside each mode (Section 4)
   - *Conversion* between modes (Section 5)

Section 2

Triangle Encoding and its Arithmetic

“Flatten” integer into polynomial

$$\begin{aligned} \mathbb{Z} &\cong \mathbb{Z}[X]/\langle X-2\rangle \\ \mathbb{Z}_{2^n} &\cong \mathbb{Z}[X]/\langle X^n - X + 2, X - 2\rangle \end{aligned}$$

- For a message $m \in \mathbb{Z}_{2^n}$, its “flattened” polynomial is

$$[m]_t = b_0 + b_1 X + b_2 X^2 + \cdots + b_{n-1} X^{n-1}$$

  where b_i is the i-th bit of m (only n bits), and $t = X - 2$
- Example: $[5]_t(X) = 1 + 0 \cdot X + 1 \cdot X^2 = 1 + X^2$
- Let $X = 2$, we have $[5]_t(2) = 1 + 2^2 = 5$

The Modulo Structure

- Why these two polynomials $X^n - X + 2$ and $X - 2$?
- (Bit Decomposition) $2 = X \Rightarrow 3 = 1 + X \Rightarrow$ Bit decomposition
- (Modulo Structure)

$$\begin{aligned} \text{Arith mod } 2^n \Leftrightarrow 2^n \equiv 0 &\Leftrightarrow X^n \equiv 0 \\ &\Leftrightarrow X^n \equiv X - 2 \end{aligned}$$

- Any polynomial $f(X)$ can be written as the following

$$f(X) = [f(2)]_t + (X - 2) \cdot I(X)$$

  which resembles "$f(X) =$ remainder $+$ divisor $\cdot$ polynomial quotient"
- We use the notation $t = X - 2$

Arithmetic for flattened polynomials

$$[m_1]_t + [m_2]_t = [m_1 + m_2]_t + tI$$
$$[m_1]_t \cdot [m_2]_t = [m_1 m_2]_t + tI$$

- Arithmetic is good, but the extra $tI(X)$ term is hard to deal with...
- What if we divide by $t = X - 2$ in $\mathbb{Q}[X]/\langle X^n - X + 2\rangle$?
- We directly get integer polynomial $I(X)$
- $\frac{[m]_t}{t}$ looks like a triangle! The k-th coefficient contains the lower $k+1$ bits.

$$\frac{[m]_t}{t} = \left(\frac{m}{2^n} - \sum_{k=0}^{n-1} \frac{1}{2^{k+1}} \underset{0 \le i \le k}{[m]} X^k \right)$$

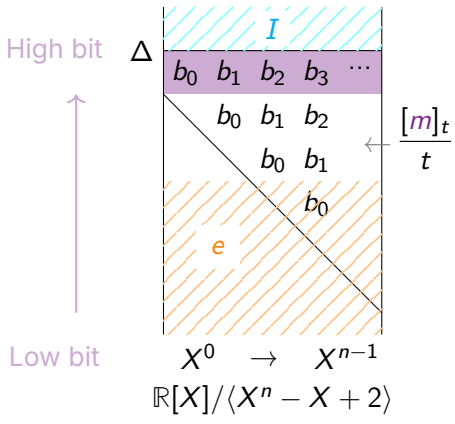
where $\underset{0 \le i \le k}{[m]} = \sum_{i=0}^{k} b_i 2^i$ is the lower $k+1$ bits of m.

Introducing the Triangle Encoding

$$[m]_t \quad \in \quad \mathbb{Z}[X]/\langle X^n - X + 2, X - 2\rangle \quad \xrightarrow{\text{Embed}} \quad \mathbb{R}[X]/\langle X^n - X + 2\rangle$$

$$\Delta\left(I + \frac{[m]_t}{t}\right) + e$$

- Divide by GBFV-style $t = X - 2$
- Scale by CKKS-style Δ
- Upper: Carries/Overflow I
- Middle: Message m in *triangle* form
- Lower: Noise e
- Fig: each column is a coefficient



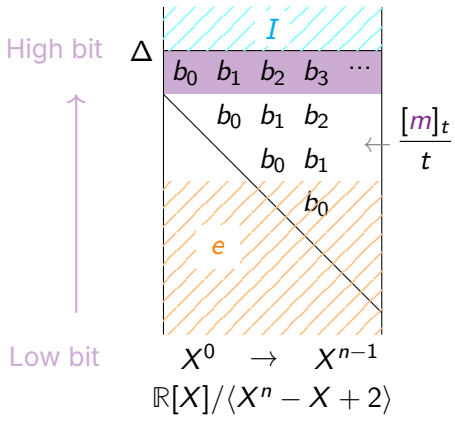
Introducing the Triangle Encoding

$$[m]_t \quad \in \quad \mathbb{Z}[X]/\langle X^n - X + 2, X - 2\rangle \quad \xrightarrow{\text{Embed}} \quad \mathbb{R}[X]/\langle X^n - X + 2\rangle$$

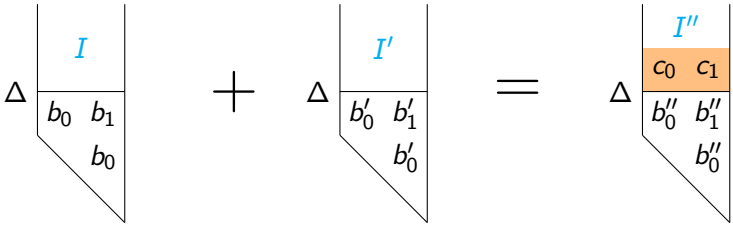
$$\Delta\left(I + \frac{[m]_t}{t}\right) + e$$

We have the following properties

1. The top row of the triangle has all bits
2. The noise can contaminate the triangle
3. Exactly decode as long as $\|te\| < \Delta/2$



Understanding Overflow I in Triangle Encoding



- In the coefficient for X^k, the $(k+1)$ bits are added accordingly
- The carries are absorbed into overflow I
- Similar "absorption" happens for multiplication

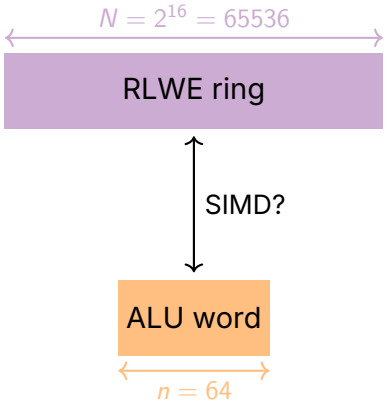
We achieve Add/Mult for $\mathbb{Z}_{2^n}$ without manual carry propagation!

Section 3

The SIMD Homomorphism Chain

SIMD for efficiency due to RLWE dimension

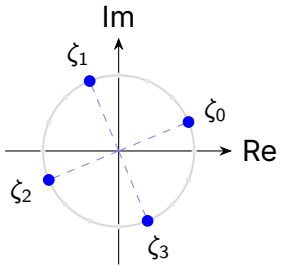
- RLWE: ring dim N is large
- ALU word: bit-width n is small
- Single-Instruction Multiple-Data
  - Many words in one ciphertext
  - Crucial for efficiency!



RLWE ring to complex slots

$$\begin{aligned} \sigma: \quad \mathbb{R}[X] \Big/ \left\langle X^N + 1 \right\rangle &\cong \mathbb{C}^{N/2} \\ f(X) &\mapsto (f(\zeta_0), f(\zeta_1), \ldots, f(\zeta_{N/2-1})) \end{aligned}$$

- Pick $N/2$ complex roots ζ_j for $X^N + 1$
- (Others are conjugate)
- RLWE ring connects to complex vector

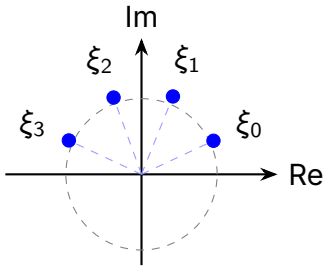


Selected roots for $X^8 + 1$

REFHE ring to complex slots

$$\begin{aligned} \tau: \quad \mathbb{R}[X]/\langle X^n - X + 2\rangle &\cong \mathbb{C}^{n/2} \\ f(X) &\mapsto (f(\xi_0), f(\xi_1), \ldots, f(\xi_{n/2-1})) \end{aligned}$$

- n is a power-of-two
- $X^n - X + 2$ is irreducible over $\mathbb{Q}$
- We call it REFHE ring
- Pick $n/2$ complex roots ξ_j for $X^n - X + 2$
- (Slightly off-the-circle)
- REFHE ring connects to complex vector



Selected roots for $X^8 - X + 2$

Recall CKKS: Naive version

$$\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle \quad \underset{\longleftarrow}{\text{Approximate}} \quad \mathbb{R}[X]\Big/\left\langle X^N+1\right\rangle \quad \cong \quad \mathbb{C}^{N/2}$$

- For a complex vector $\vec{z} = (z_0, z_1, \ldots, z_{N/2-1}) \in \mathbb{C}^{N/2}$
- Encode it into polynomial $m(X)$ in $\mathbb{R}[X]/\langle X^N+1\rangle$ through the inverse canonical embedding $\sigma^{-1}(\vec{z})$ and scaling by Δ
- Then round $m(X)$ to the nearest integer polynomial in $\mathbb{Z}[X]/\langle X^N+1\rangle$
- We call $\mathbb{C}^{N/2}$ slots and the polynomial as the coefficients
- Add/Mult in $\mathcal{R}_Q$ correspond to *approximate* Add/Mult in $\mathbb{C}^{N/2}$

Main technique: integer arithmetic from complex slots

$$\mathbb{Z}_{2^n} \quad \cong \quad \mathbb{Z}[X]/\langle X^n - X + 2, X - 2\rangle \quad \overset{\text{Embed}}{\longrightarrow} \quad \mathbb{R}[X]/\langle X^n - X + 2\rangle \quad \cong \quad \mathbb{C}^{n/2}$$

- We embed $\mathbb{Z}_{2^n}$ into $\mathbb{R}[X]/\langle X^n - X + 2\rangle$ using triangle encoding
- The "Embed" here is a homomorphism,
  as we have dealt with $\langle X - 2\rangle$ properly using the triangle form
- Then use the REFHE ring mapping $\tau(\cdot)$ to get $n/2$ complex slots

The SIMD Homomorphism Chain

$$\boxed{\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle}$$

$$\uparrow$$

$$\mathbb{R}[X]/\langle X^N+1\rangle \quad \cong \quad \mathbb{C}^{N/2} \quad = \quad \left(\mathbb{C}^{n/2}\right)^{N/n} \quad \cong \quad \left(\mathbb{R}[X]/\langle X^n-X+2\rangle\right)^{N/n}$$

$$\uparrow \Delta$$

$$\boxed{\mathbb{Z}_{2^n}^{N/n}} \quad \cong \quad \left(\mathbb{Z}[X]/\langle X^n-X+2, X-2\rangle\right)^{N/n}$$

N/n number of $\mathbb{Z}_{2^n}$ slots from the usual RLWE ring!

Expert slide: The punchline for this chain

This is CKKS

$$\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle \xleftarrow{\text{Approximate}} \mathbb{R}[X]\Big/\left\langle X^N+1\right\rangle \cong \mathbb{C}^{N/2}$$

This is (roughly) BGV/BFV/GBFV/CLPX

$$\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle \xleftarrow{\text{Contain}} \mathbb{Z}[X]\Big/\left\langle X^N+1, X^k-b\right\rangle \cong \mathbb{Z}_p^{N/k}$$

This is REFHE

$$\mathbb{Z}[X]\Big/\left\langle X^N-X+2, Q\right\rangle \xleftarrow{\text{Contain}} \mathbb{Z}[X]/\langle X^n-X+2, X-2\rangle \cong \mathbb{Z}_{2^n}$$

Previous works focus on one isomorphism
We have a *chain* of isomorphisms!

A concrete example of homomorphism chain

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\sigma^{-1}} \mathbb{C}^2 \xleftarrow{\tau} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\Delta(0.7 - 0.cX \\ -0.eX^2 - 0.fX^3)$$

- Notation: $0.c$ is the hexadecimal representation; Ignore $0x$ for simplicity
- Example: $N = n = 4$ and $m = 0xf$ (Hex representation of 15)
- Its flattened polynomial is $[0xf]_t = 1 + X + X^2 + X^3$
- Its triangle form is $\frac{[0xf]_t}{t} = 0.7 - 0.cX - 0.eX^2 - 0.fX^3$

A concrete example of homomorphism chain

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\sigma^{-1}} \mathbb{C}^2 \xleftarrow{\tau} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\begin{matrix}\Delta(-0.858ea - 0.580ea\sqrt{-1}) \\ \Delta(-0.02715 - 2.87bc5\sqrt{-1})\end{matrix} \qquad \begin{matrix}\Delta(0.7 - 0.cX \\ -0.eX^2 - 0.fX^3)\end{matrix}$$

- Notation: $0.c$ is the hexadecimal representation; Ignore $0x$ for simplicity
- Example: $N = n = 4$ and $m = 0xf$ (Hex representation of 15)
- Its flattened polynomial is $[0xf]_t = 1 + X + X^2 + X^3$
- Its triangle form is $\frac{[0xf]_t}{t} = 0.7 - 0.cX - 0.eX^2 - 0.fX^3$
- We then use τ to get the complex slots,

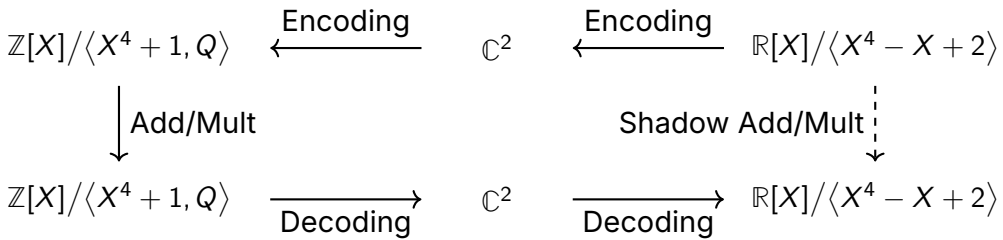
A concrete example of homomorphism chain

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\sigma^{-1}} \mathbb{C}^2 \xleftarrow{\tau} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\begin{array}{ccc} \Delta(-0.44 + 0.97852X & \Delta(-0.858ea - 0.580ea\sqrt{-1}) & \Delta(0.7 - 0.cX \\ -1.6fe5X^2 + 0.f43b7X^3) & \Delta(-0.02715 - 2.87bc5\sqrt{-1}) & -0.eX^2 - 0.fX^3) \end{array}$$

- Notation: $0.c$ is the hexadecimal representation; Ignore $0x$ for simplicity
- Example: $N = n = 4$ and $m = 0xf$ (Hex representation of 15)
- Its flattened polynomial is $[0xf]_t = 1 + X + X^2 + X^3$
- Its triangle form is $\frac{[0xf]_t}{t} = 0.7 - 0.cX - 0.eX^2 - 0.fX^3$
- We then use τ to get the complex slots, and σ^{-1} to get the RLWE ring

Encoding? Computation?

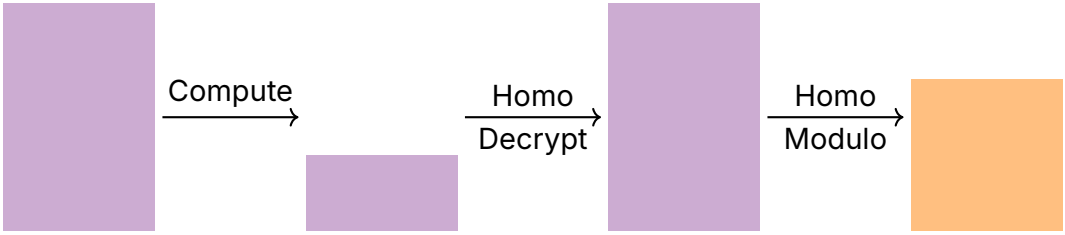


- People confused about encoding (offline) and computation (online)
- All **computation** happens in the RLWE ring $\mathbb{Z}[X]/\langle X^N+1, Q\rangle$
- Via the *chain*, we have *shadow* Add/Mult in the REFHE ring
- We do not homomorphically compute τ/σ, unless we are *bootstrapping*...

Section 4

Bootstrappings for Arithmetic Mode

Crash Course on Bootstrapping



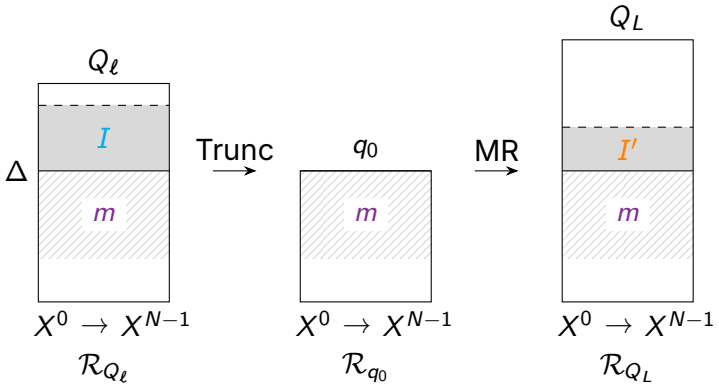
- Homomorphic computation consumes some "budget"
  - Technically, noise budget and level budget
- Recall RLWE decrypt: Inner product with $(1, s)$ then modulo-by-q_0
  - $\langle(-as + m + e, a), (1, s)\rangle = m + e + q_0 I \equiv m + e \pmod{q_0}$ in ring $\mathcal{R}_{q_0}$
- Bootstrapping (Gentry): *Homomorphically* evaluate the decryption circuit

Recall CKKS: Bootstrapping

$$\mathcal{R}_{q_0} \xrightarrow[\text{ModRaise}]{\text{Homo Decrypt}} \mathcal{R}_Q \xrightarrow[\text{CoeffsToSlots}]{} \mathbb{C}^{N/2} \xrightarrow[\text{EvalSine}]{\text{Homo Modulo}} \mathbb{C}^{N/2} \xrightarrow[\text{SlotsToCoeffs}]{} \mathcal{R}_{Q_\ell}$$

- ModRaise: Homomorphic Decryption: Introduce $q_0 I$ in poly coefficients
- CoeffsToSlots: Move $q_0 I$ into $\mathbb{C}$ slots
- EvalSine: Use $\sin(\cdot)$ to approximate $[\cdot]_{q_0}$: Clean $q_0 I$ and keep Δm in slots
- SlotsToCoeffs: Move Δm back to coeffs

The “Truncate and ModRaise” operation



- For $\Delta(I + m)$ in coefficient encoding where $q_0 = \Delta$ and $\|m\| < 1/2$
- Truncate through scaling/modulus switching [KN25]
- Then use CKKS ModRaise to get another $\Delta I'$

The “Truncate and ModRaise” operation (Example)

$$\Delta I + \Delta m \pmod{Q_\ell} \xrightarrow{\text{Truncate}} \Delta m \pmod{q_0} \xrightarrow{\text{ModRaise}} \Delta I' + \Delta m \pmod{Q}$$

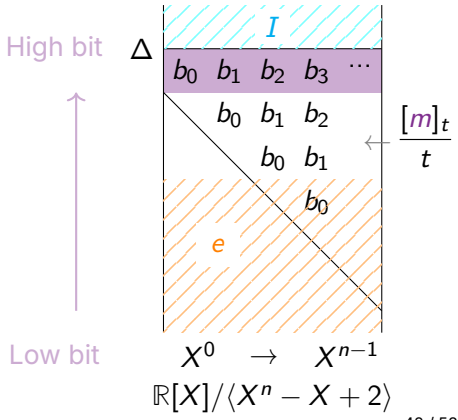
- Input: 0x12348899 $\pmod{2^{32}}$
- Truncate: 0x8899 $\pmod{2^{16}}$
- ModRaise: 0x568899 $\pmod{2^{32}}$
- The extra 56 is the new overflow I'
  - I′ is much smaller and is related to secret key distribution (e.g., encapsulated key)
- Essentially, this is a *noisy* modulo-by-2^{16} operation in coefficients!

Recall the triangle encoding

$$\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle \xleftarrow{\text{Approximate}} \mathbb{C}^{N/2} \xleftarrow{\tau} \mathbb{R}[X]/\langle X^n - X + 2\rangle \xleftarrow{\text{Embed}} \mathbb{Z}_{2^n}$$

$$\Delta\left(I + \frac{[m]_t}{t}\right) + e$$

- What if we can reset I at once?
- In previous slides, we see that the polynomial in $\mathcal{R}_Q$ is quite unregular...
- We can use CoeffsToSlots-ish process to move the triangle to $\mathcal{R}_Q$ and truncate



The A2A-I Bootstrapping

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\text{Homo } \sigma} \mathbb{C}^2 \xleftarrow{\text{Homo } \tau^{-1}} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\cdots \qquad \cdots \qquad \Delta I(X) + \Delta(0.7 - 0.cX - 0.eX^2 - 0.fX^3)$$

The A2A-I Bootstrapping

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \quad \xleftarrow{\text{Homo } \sigma} \quad \begin{matrix}\mathbb{C}^2 \\ \Delta(I_0+0.7+0i) \\ \Delta(I_1-0.c+0i) \\ \Delta(I_2-0.e+0i) \\ \Delta(I_3-0.f+0i)\end{matrix} \quad \xleftarrow{\text{Homo } \tau^{-1}} \quad \mathbb{R}[X]/\langle X^4-X+2\rangle$$

$$\cdots \qquad\qquad \cdots$$

- Apply REFHE-to-$\mathbb{C}$,

The A2A-I Bootstrapping

$$\mathbb{Z}[X]/\langle X^4+1, q_0\rangle \xleftarrow{\text{Homo } \sigma} \mathbb{C}^2 \xleftarrow{\text{Homo } \tau^{-1}} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\Delta I(X) + \Delta(+0.7 - 0.cX - 0.eX^2 - 0.fX^3) \quad \cdots \quad \cdots$$

- Apply REFHE-to-$\mathbb{C}$, then apply $\mathbb{C}$-to-$\mathcal{R}$ (aka SlotsToCoeffs in CKKS)

The A2A-I Bootstrapping

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\text{Homo } \sigma} \mathbb{C}^2 \xleftarrow{\text{Homo } \tau^{-1}} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\Delta I'(X) + \Delta(+0.7 - 0.cX - 0.eX^2 - 0.fX^3) \quad \cdots \quad \cdots$$

- Apply REFHE-to-$\mathbb{C}$, then apply $\mathbb{C}$-to-$\mathcal{R}$ (aka SlotsToCoeffs in CKKS)
- Then we reset $I(X)$ to $I'(X)$ through "Truncate and ModRaise".

The A2A-I Bootstrapping

$$\mathbb{Z}[X]/\langle X^4+1, Q\rangle \xleftarrow{\text{Homo } \sigma} \mathbb{C}^2 \xleftarrow{\text{Homo } \tau^{-1}} \mathbb{R}[X]/\langle X^4 - X + 2\rangle$$

$$\cdots \qquad \cdots \qquad \Delta I'(X) + \Delta(0.7 - 0.cX - 0.eX^2 - 0.fX^3)$$

- Apply REFHE-to-$\mathbb{C}$, then apply $\mathbb{C}$-to-$\mathcal{R}$ (aka SlotsToCoeffs in CKKS)
- Then we reset $I(X)$ to $I'(X)$ through "Truncate and ModRaise".
- Apply $\mathcal{R}$-to-$\mathbb{C}$ and $\mathbb{C}$-to-REFHE to get the triangle back in REFHE ring

The A2A Refreshing

- A2A-e: Bootstrap the noise (like META-BTS, EvalRound, etc.).
  - Multiply by t and truncate away $tI + [m]_t$

$$\Delta(\cancel{tI + [m]_t}) + te$$

  - Bootstrap the noise te, multiply by t^{-1} and subtract from original ct
- A2A-I and A2A-e each amounts to roughly 1 CKKS bootstrapping

A SIMD Arithmetic FHE scheme with $O(1)$ bootstrapping for $\mathbb{Z}_{2^n}$

Next step: Boolean! We also achieve $O(1)$ BTS

Section 5

Booleans and $O(1)$ Bootstrapping

The discrete CKKS framework: Booleans

$$\mathbb{Z}[X] \Big/ \left\langle X^N + 1, Q \right\rangle \quad \xleftarrow{\text{Approximate}} \quad \mathbb{C}^{N/2}$$

- Core idea in discrete CKKS: encode bits in complex slots

$$\Delta(b_0, b_1, \ldots, b_{n-1}) \in \mathbb{C}^{n/2}$$

- Bit-AND can be done through Mult in $\mathbb{C}$!

$$\Delta(b_0 \& b_0') = \frac{1}{\Delta} \Delta b_0 \cdot \Delta b_0'$$

- Left/Right shift can be done through CKKS rotation

The discrete CKKS framework: Small Integers

$$\mathbb{Z}[X]\Big/\left\langle X^N+1, Q\right\rangle \quad \xleftarrow{\text{Approximate}} \quad \mathbb{C}^{N/2}$$

- Power-of-two p and plaintext space $\mathbb{Z}_p$ (e.g. $p = 2^4$ and $p = 2^8$)
- Encode short integers $x_i \in \mathbb{Z}_p$ in complex slots

$$\frac{\Delta}{p}(x_0, x_1, \ldots, x_{N/2-1}) \in \mathbb{C}^{N/2}$$

- Focusing on a single slot, its content with noise is

$$\Delta\left(\frac{x}{p} + e\right)$$

- Example: 2 in $\mathbb{Z}_{16}$ encoded as $\Delta \times 0.125000000871231$

Discrete CKKS Functional Bootstrapping

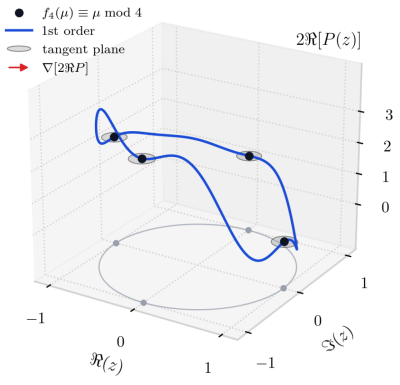
$$\Delta\left(\frac{x}{p}+e\right) \quad \mapsto \quad \Delta\left(\frac{f(x)}{p}+e'\right)$$

- Now we want to evaluate a function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ on the encrypted data
- Discrete CKKS Functional Bootstrapping
  1. Evaluate f
  2. Clean noise e to smaller e'
- Example: $f_i(x) =$ the i-th bit of x (it is non-linear!)
  - Input: $\Delta \times 0.125000008712313328$ (i.e., 2 in $\mathbb{Z}_{16}$)
  - Output: $\Delta \times 0.000000000000123456$ (i.e., 0 in $\mathbb{Z}_{16}$)

Discrete CKKS Functional Bootstrapping: Multi-Value

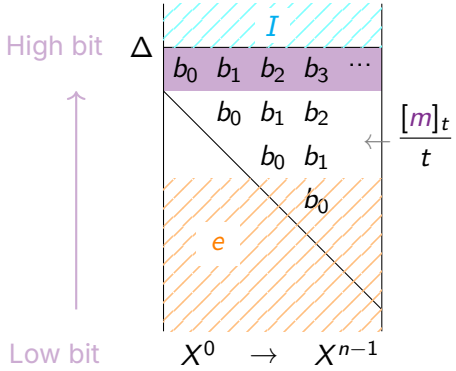
$$\Delta\left(\frac{x}{p}+e\right) \overset{\text{EvalExp}}{\mapsto} \Delta\exp\left(2\pi i\frac{x}{p}\right)+e' \overset{\text{EvalLUT}}{\mapsto} \Delta\left(\frac{f(x)}{p}+e''\right)$$

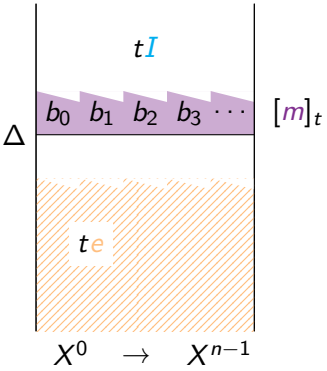
- Fig: Sparse-THI in [AKPZ26] (AC'26)
- EvalExp: move $\mathbb{Z}_p$ into the unit circle (gray dots in the bottom)
- EvalLUT: evaluate $\sum P_k z^k$
- Different functional $f \Rightarrow$ different P_k's
- We can evaluate *many* functions at once
- Takeaway: Use 1 BTS to extract all bits!



The Triangle and the Carries

$$\Delta\left(I + \frac{[m]_t}{t}\right) + e \xrightarrow{\times t} \Delta\,(tI + [m]_t) + te \xrightarrow{\text{BTS}} \text{All bits of } (tI + [m]_t)$$





The 3-to-2 Contraction Circuit

All bits of $(tI + [m]_t)$

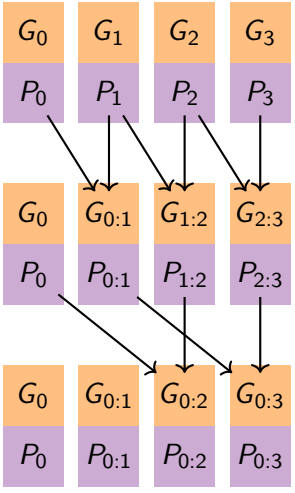
- We obtain 7 bit strings from one MV-FBT
- We need to reduce to 2 bit strings for applying carry-lookahead
- For each group consisting of 3 bit strings, we can reduce it to 2 bit string
- For input bit x, y, z, we have

$$c := yz + (y + z - 2yz)x, \qquad s := x + y + z - 2c$$

- And we rotate c by one slot to obtain the other bit string
- This consumes two CKKS levels
- $7 \to 5 \to 4 \to 3 \to 2$ (8 levels)

Carry-Lookahead Explained

- Need to combine 2 bit strings
- Naive way: $O(n)$ carry propagation
- Carry-Lookahead: $O(\log n)$ depth
  - For each bit, it has two states
  - Generate: $G_i = a_i \& b_i$
  - Propagate: $P_i = a_i \oplus b_i$
  - Use $G_{0:i}$ and $P_{0:i}$ to determine the carry
- This is Boolean circuit (with bit-shift)
- $O(\log n)$ **levels** instead of $O(\log n)$ **BTS**



The $O(1)$ BTS for Arith-To-Boolean (Sketch)

1. After A2A-I, $\|I\|$ is bounded by a constant K (e.g. $K = 16$)
   - It is unrelated to the bit-width n of the triangle
   - It is related to the secret key distribution
2. We multiply by t, to form $(tI + [m]_t)$
   - $[m]_t$ is the bits
   - tI is the carries
   - They are mixed and should propagate to separate them
3. Use Multi-Value Functional BTS to extract all bits of $(tI + [m]_t)$
4. Use 3-to-2 contraction Boolean circuit to reduce to 2 bit strings
5. $O(\log n)$ depth of carry-lookahead circuit $\Rightarrow$ $[m]_t$
6. $[m]_t$ in $\mathbb{C}^n$ is the bit representations.

We obtain all bits with $O(1)$ BTS, and $O(\log n)$ extra CKKS levels!

Technical Summary

1. Triangle Encoding $\Rightarrow \mathbb{Z}_{2^n}$ arithmetic
2. Homomorphism Chain $\Rightarrow$ SIMD
3. RLWE Noisy Modulo Op $\Rightarrow O(1)$ BTS for A$\rightarrow$A
4. Multi-Value FBT + Boolean Circuit $\Rightarrow O(1)$ BTS for A$\rightarrow$B