EvalRound Algorithm in CKKS Bootstrapping

Seonghak Kim[1], Minji Park[2], Jaehyung Kim[1],
Taekyung Kim[1] and Chohong Min[2*]

[1] Crypto Lab Inc. and [2] Ewha Womans University

[*] Corresponding Author

This talk is about …

Topic of this talk

- **CKKS**[1] is a representative HE for real numbers

- CKKS **bootstrapping**[2] is an operation which enables a CKKS ciphertext to be evaluated arbitrary times

- The efficiency of CKKS bootstrapping greatly depends on its **modulus consumption**

- We propose a simple yet effective variant of CKKS bootstrapping with **less modulus consumption** which can be implemented with a **negligible cost**

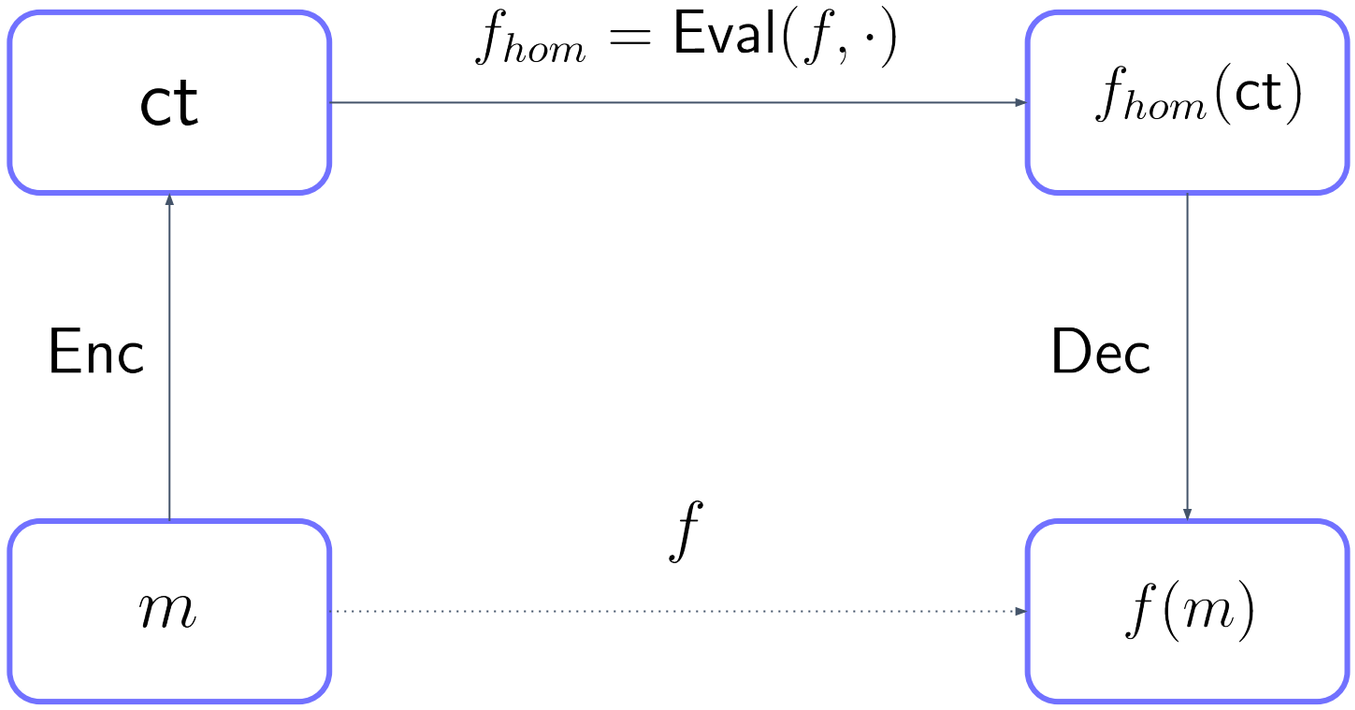
[1][Cheon et al., Asiacrypt17] Homomorphic Encryption for Arithmetic of Approximate Numbers

[2][Cheon et al., Eurocrypt18] Bootstrapping for Approximate Homomorphic Encryption

TABLE
OF
CONTENTS

Background

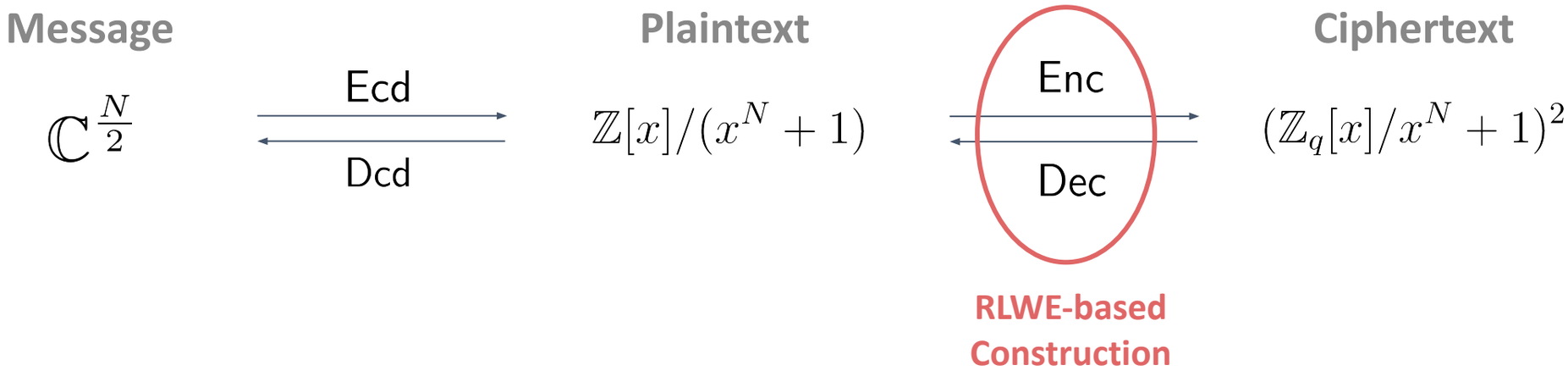
The concept of Homomorphic Encryption



CKKS : HE Scheme for real numbers

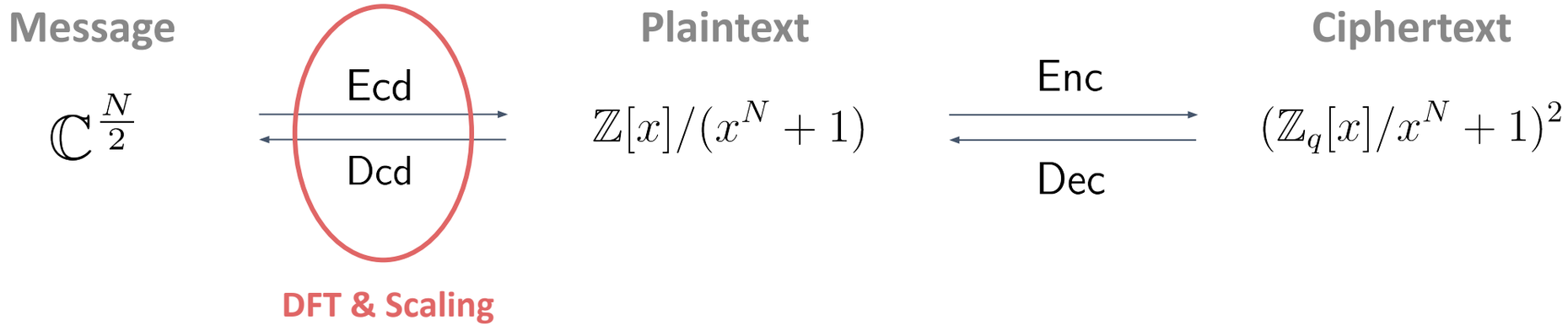
Message		Plaintext		Ciphertext
$\mathbb{C}^{\frac{N}{2}}$	Ecd ⇀ ↽ Dcd	$\mathbb{Z}[x]/(x^N+1)$	Enc ⇀ ↽ Dec	$(\mathbb{Z}_q[x]/x^N+1)^2$

CKKS : HE Scheme for real numbers



- Homomorphic structure is constructed on the **polynomial with finite field coefficients**
- $\mathsf{ct} = \mathsf{Enc}(\mathsf{pt}) \implies [\langle \mathsf{ct}, \mathsf{sk} \rangle]_q = \mathsf{pt} + e$

CKKS : HE Scheme for real numbers



- Homomorphism between polynomials and vectors by Discrete Fourier Transform
- Scale up / down to encode / decode real numbers
- $\mathsf{Ecd}(\mathbf{z};\Delta) = \lfloor \Delta \cdot \mathsf{iDFT}(\mathbf{z}) \rceil, \mathsf{Dcd}(\mathsf{pt};\Delta) = \mathsf{DFT}(\mathsf{pt}/\Delta)$

Modulus consumption of CKKS

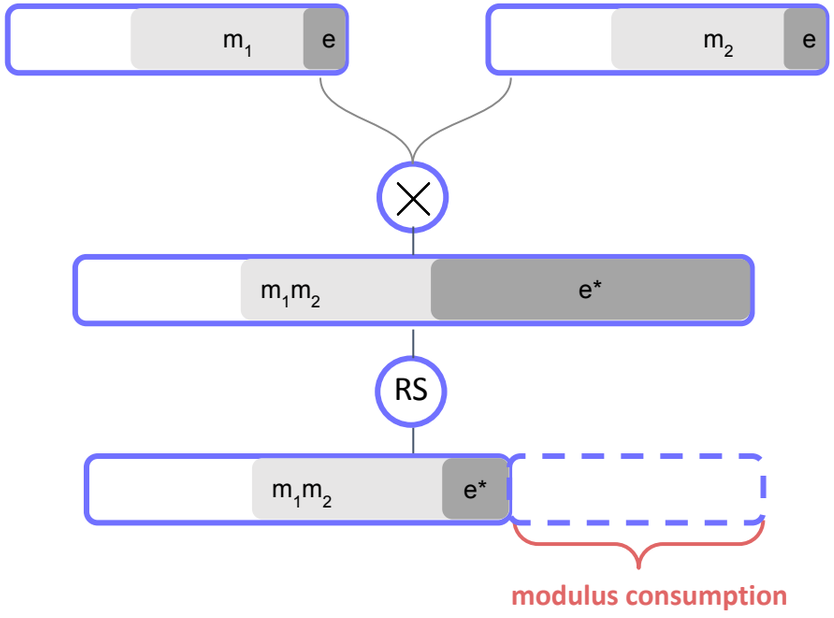
- Size of encrypted plaintext grows by each multiplication

$$\mathsf{pt}_1 = \mathsf{Ecd}(\mathbf{z}_1; \Delta_1), \mathsf{pt}_2 = \mathsf{Ecd}(\mathbf{z}_2; \Delta_2)$$
$$\Longrightarrow \ \mathsf{pt}_1 * \mathsf{pt}_2 = \mathsf{Ecd}(\mathbf{z}_1\mathbf{z}_2; \Delta_1\Delta_2)$$

- CKKS supports **rescaling** to maintain the size of encrypted plaintext

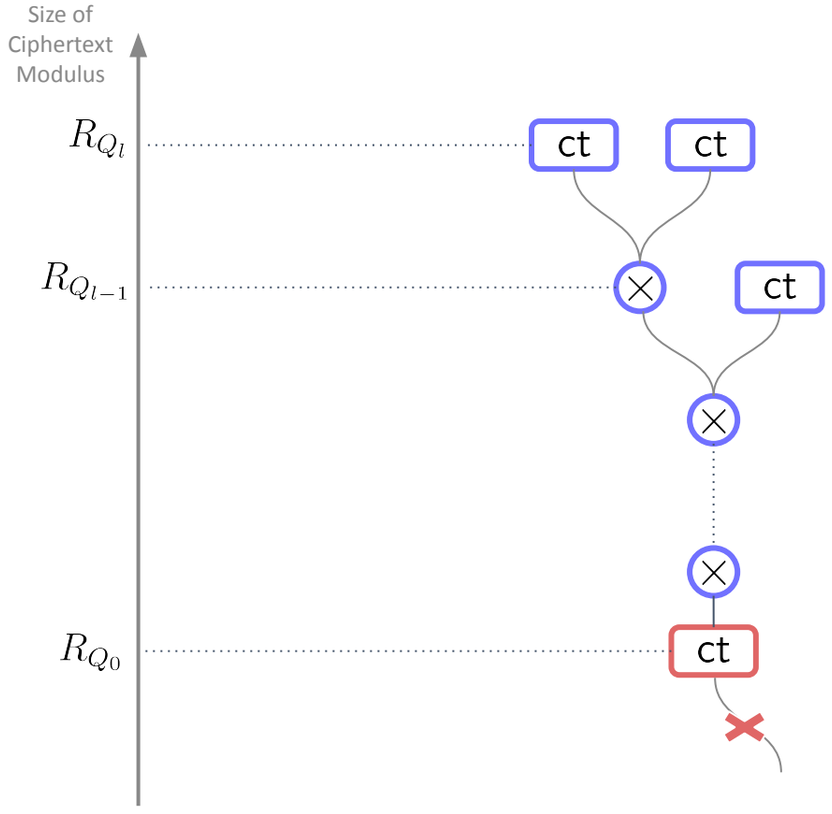
$$\mathsf{RS}(\ell, \mathsf{ct}) = \left\lfloor q_\ell^{-1}\mathsf{ct} \right\rceil \pmod{Q_{\ell-1}}$$

- Rescaling **consumes modulus** as much as the encrypted plaintext has scaled down



CKKS Bootstrapping

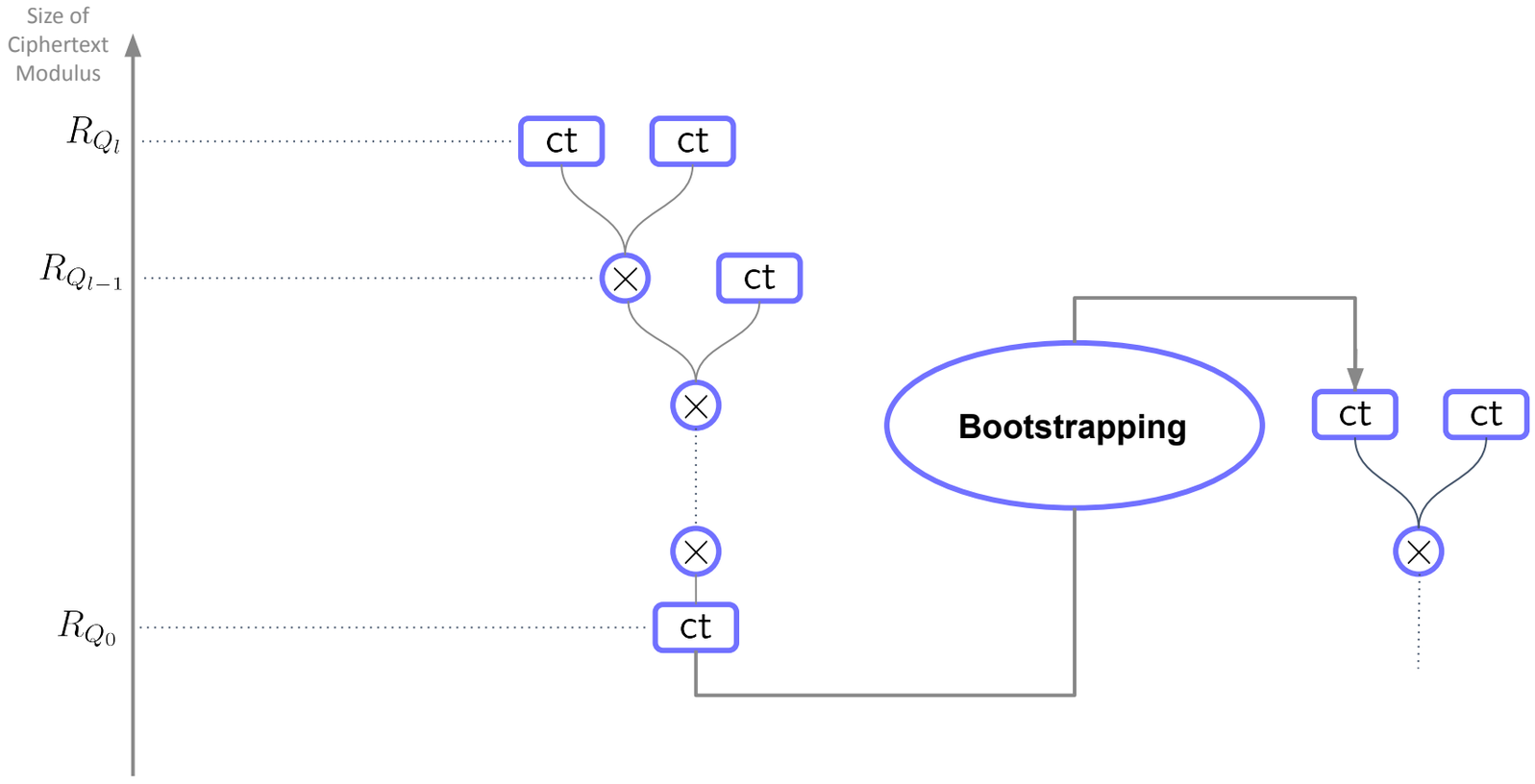
- **Bootstrapping**[2] is the operation of recovering the modulus of ciphertext



[2][Cheon et al., Eurocrypt18] Bootstrapping for Approximate Homomorphic Encryption

CKKS Bootstrapping

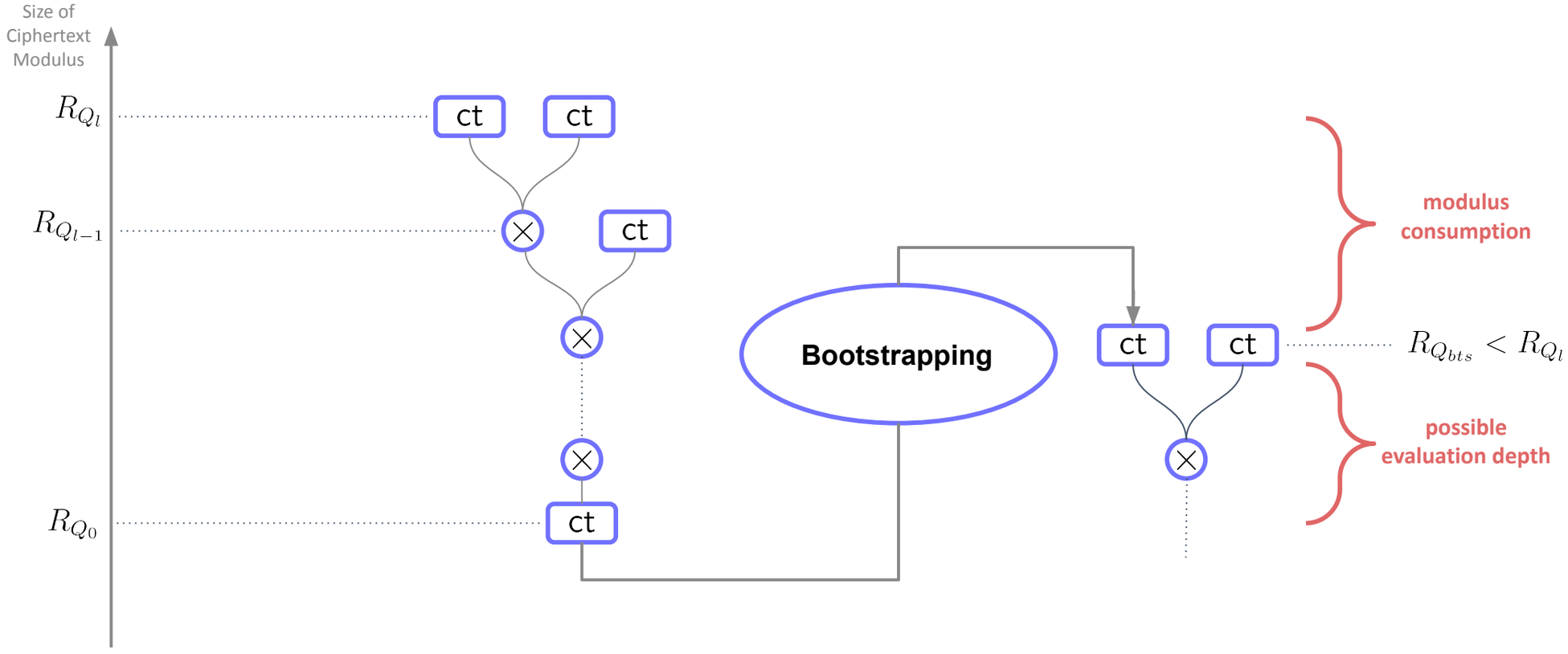
- **Bootstrapping**[2] is the operation of recovering the modulus of ciphertext



[2][Cheon et al., Eurocrypt18] Bootstrapping for Approximate Homomorphic Encryption

CKKS Bootstrapping

- **Bootstrapping**[2] is the operation of recovering the modulus of ciphertext
- Our goal is to **reduce the modulus consumption** of CKKS Bootstrapping

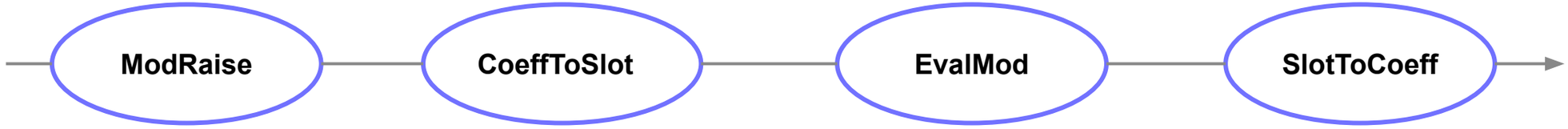


[2][Cheon et al., Eurocrypt18] Bootstrapping for Approximate Homomorphic Encryption

Conventional bootstrapping algorithm

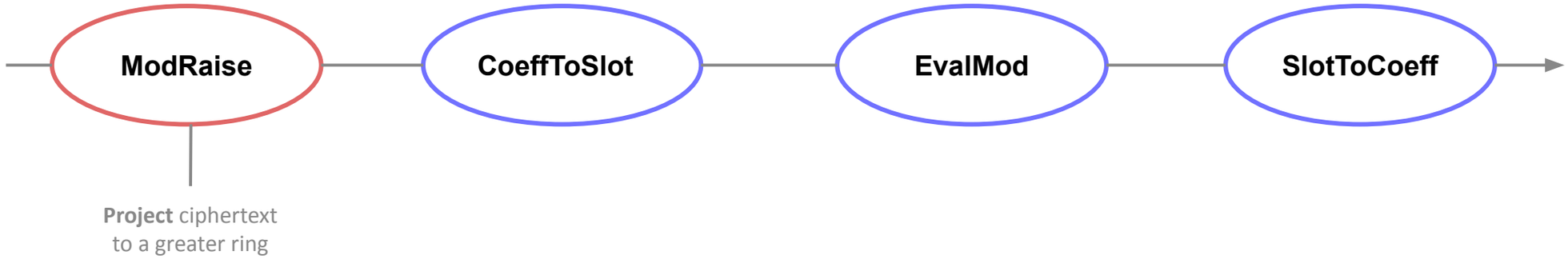
Steps of conventional bootstrapping algorithm

- **Input :** $\mathsf{ct} = \mathsf{Enc}_{Q_0}(\mathsf{pt}) \in R^2_{Q_0}$
- **Desired output :** $\mathsf{ct}_{bts} = \mathsf{Enc}_{Q_{bts}}(\mathsf{pt}) \in R^2_{Q_{bts}}$ for some $Q_0 < Q_{bts}$



Conventional bootstrapping algorithm

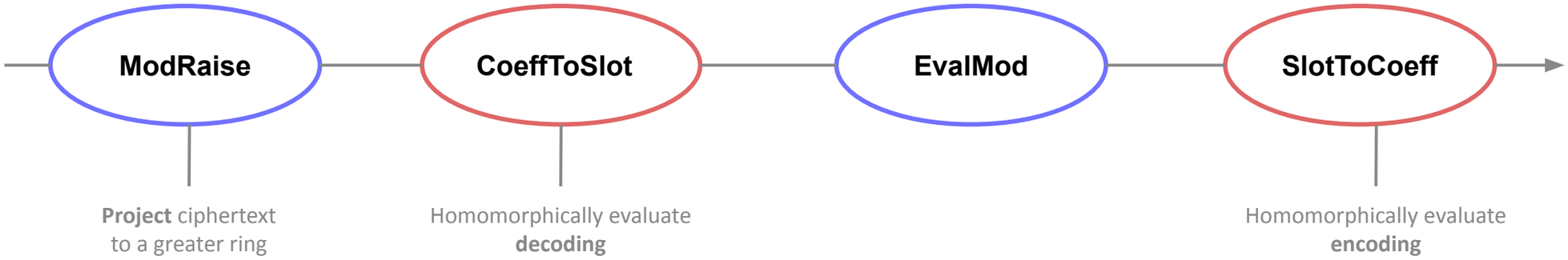
Step 1 : ModRaise



- $\mathsf{ct} \in R^2_{Q_0} \subset R^2_{Q_l}, \text{ for } \forall l \geq 0$
- $\text{Recall that } [\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_0} = \mathsf{pt}; [\langle \mathsf{ct}, \mathsf{sk} \rangle]_{Q_l} = \mathsf{pt} + q_0 \cdot I$
- How can we remove $q_0 \cdot I$ and recover original encrypted plaintext?

Conventional bootstrapping algorithm

Step 2 : CoeffToSlot (and SlotToCoeff)



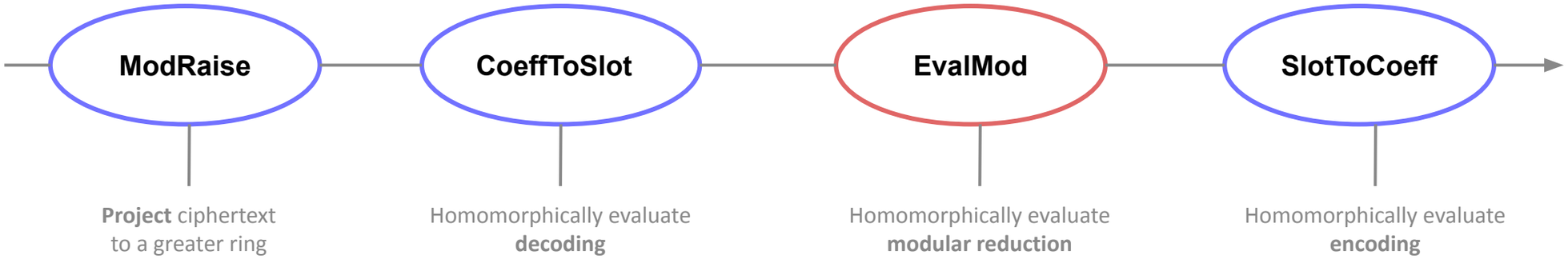
- We want manipulate the **coefficients** of plaintexts,
  but we have only methods for manipulating the encrypted **message(slots)**

- As $\mathsf{Ecd}(\mathbf{z};\Delta) = \lfloor \Delta \cdot \mathsf{iDFT}(\mathbf{z}) \rceil$ is a linear map, we can homomorphically evaluate encoding / decoding

- By definition, we get the ciphertext encrypting $\mathsf{pt} + q_0 \cdot I$ on slot side by homorphic decoding

Conventional bootstrapping algorithm

Step 3 : EvalMod



- Homomorphically evaluate **modular reduction** to compute pt from $\mathrm{pt} + q_0 \cdot I$
- Can simply approximated by sine function; various approaches[3,4,5] have been proposed

[3][Chen et al., Eurocrypt19] Improved bootstrapping for approximate homomorphic encryption

[4][Han et al., CT-RSA20] Better bootstrapping for approximate homomorphic encryption

[5][Lee et al., Eurocrypt22] High-precision bootstrapping for approximate homomorphic encryption by error variance minimization

EvalRound algorithm

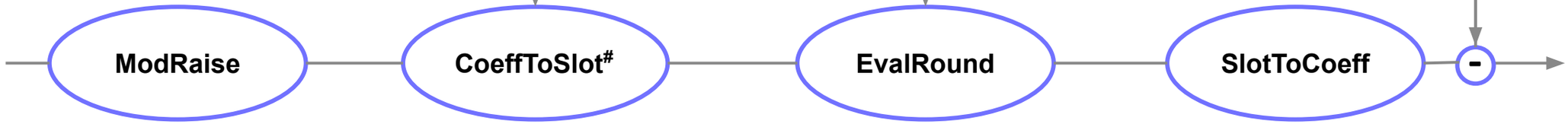
Construction of EvalRound algorithm

Conventional Bootstrapping

EvalRound Algorithm

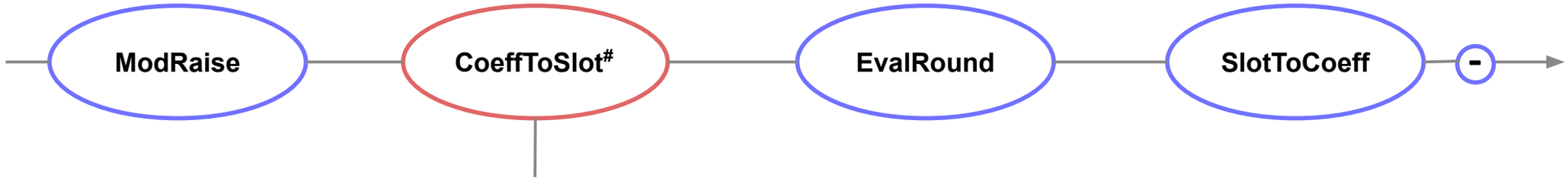
Compute **imprecisely** to **consume less modulus**

Perform modular **rounding** instead of **reduction**



EvalRound algorithm

Modification 1 : CoeffToSlot to CoeffToSlot#



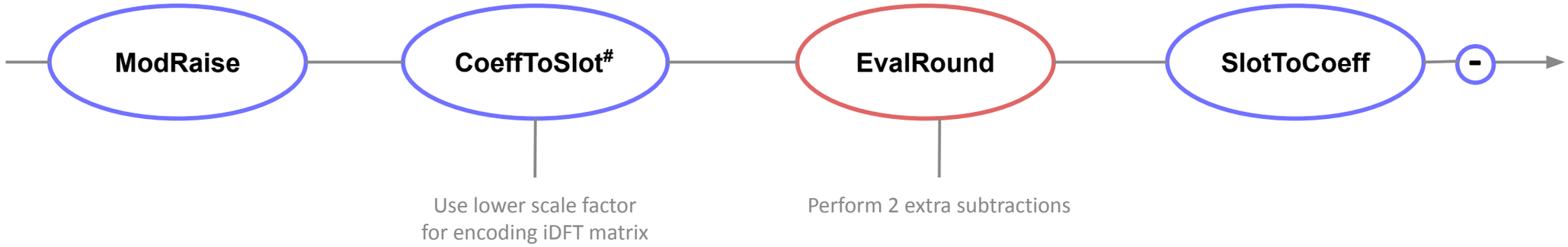
- As CoeffToSlot is multiplying iDFT matrix homomorphically,

$$\mathrm{pt}_{CtS} = \mathrm{pt}_{diag_1} * \mathrm{pt}_{z_1} + \ldots + \mathrm{pt}_{diag_k} * \mathrm{pt}_{z_k}$$

where pt_{diag_i} are encoded diagonals of iDFT matrix
and pt_{z_i} are corresponding rotated plaintexts.

- Using lower scale factor on the matrix causes :
  - **Less modulus consumption** by rescaling
  - **Imprecise result** due to the imprecise value of iDFT matrix

Modification 2 : EvalMod to EvalRound

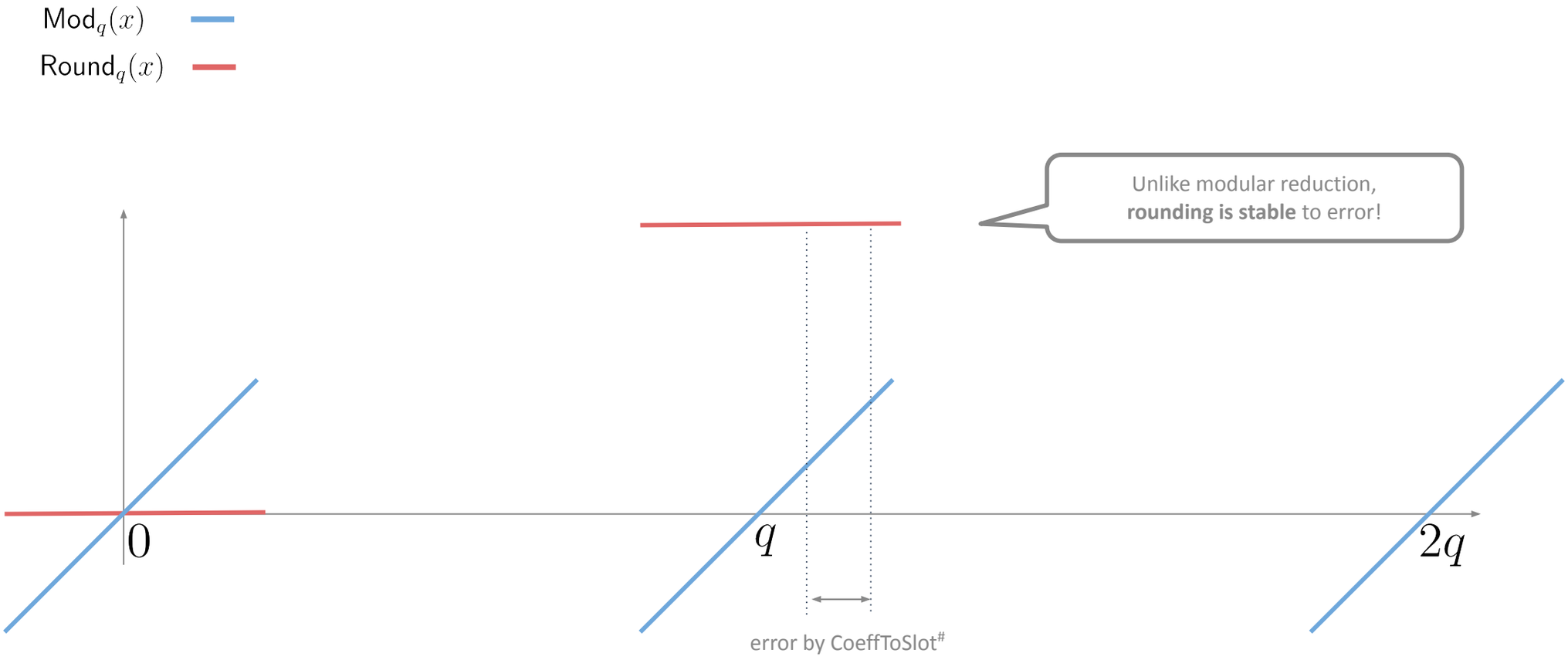


- Define the homomorphic operation EvalRound as, $\mathsf{EvalRound} := \mathsf{id} - \mathsf{EvalMod}$

- The defined EvalRound corresponds to the modular rounding

- $$\begin{aligned}\mathsf{StC} \circ \mathsf{EvalMod} \circ \mathsf{CtS} &= \mathsf{StC} \circ (\mathsf{id} - \mathsf{EvalRound}) \circ \mathsf{CtS} \\ &= \mathsf{StC} \circ \mathsf{id} \circ \mathsf{CtS} - \mathsf{StC} \circ \mathsf{EvalRound} \circ \mathsf{CtS} \\ &= \mathsf{id} - \mathsf{StC} \circ \mathsf{EvalRound} \circ \mathsf{CtS}\end{aligned}$$

Stability of EvalRound



Evaluation Result

	Size of Message	Bit size of Modulus	Scale Factor for CoeffToSlot	Bootstrap Bit Precision	Modulus Consumption
Conventional	2^{16}	2900	2^{60}	-12.53	1160
Proposed			2^{29}	-14.80	1076 = 1160 - 84

- Reduced **84** bits on a practical parameter, which is sufficient to increase **evaluation depth**s
- Effective when the evaluation depth after bootstrap is low
- **No assumption / negligible effort** is needed to upgrade the original bootstrapping to EvalRound

CRYPTO LAB

Crypto Lab Inc.