Bootstrapping Small Integers with CKKS

Youngjin Bae[1], Jaehyung Kim[1], Damien Stehlé[1], **Elias Suvanto**[1,2]

eprint.iacr.org/2024/1637

1 HEAAN CRYPTO LAB

2 University of Luxembourg

CRYPTO LAB

Main results:

- The first CKKS functional bootstrapping algorithm for ciphertexts encoding small integers called **SI-BTS**
- A batch-bits bootstrapping **BB-BTS** algorithm with a **2.4x** throughput improvement compared to [BCKS24]

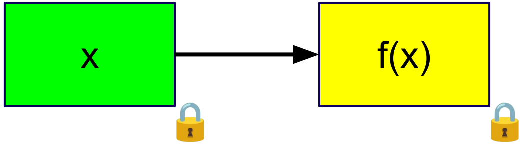
Amortized time for evaluating a **8-bit LUT** on encrypted integer:	Amortized boolean gate time:
$3.8ms$	$7.4\mu s$

single CPU thread, 128 bits of security

FHE background

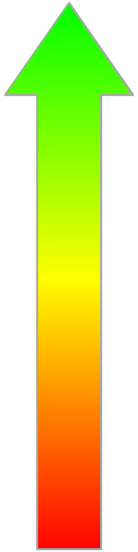
Fully Homomorphic Encryption

Computation in encrypted state



Different HE schemes, different native data types:

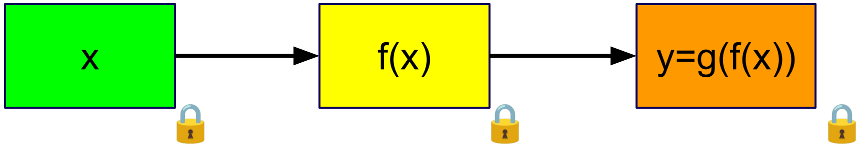
- Finite fields: **BGV/BFV**
- Integers: **TFHE**
- Floating point: **CKKS**



computation budget

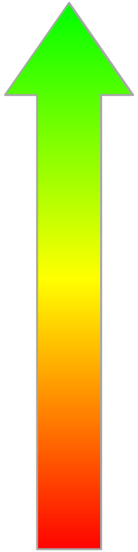
Fully Homomorphic Encryption

Computation in encrypted state



Different HE schemes, different native data types:

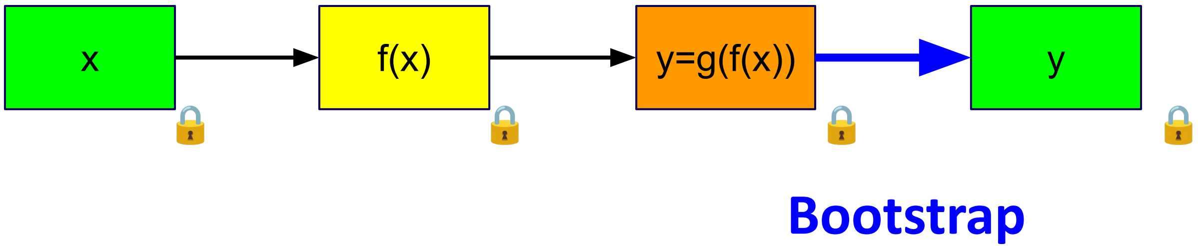
- Finite fields: **BGV/BFV**
- Integers: **TFHE**
- Floating point: **CKKS**



computation budget

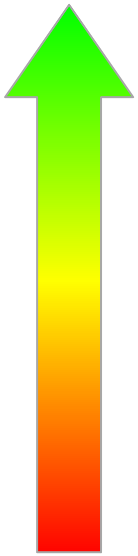
Fully Homomorphic Encryption

Computation in encrypted state



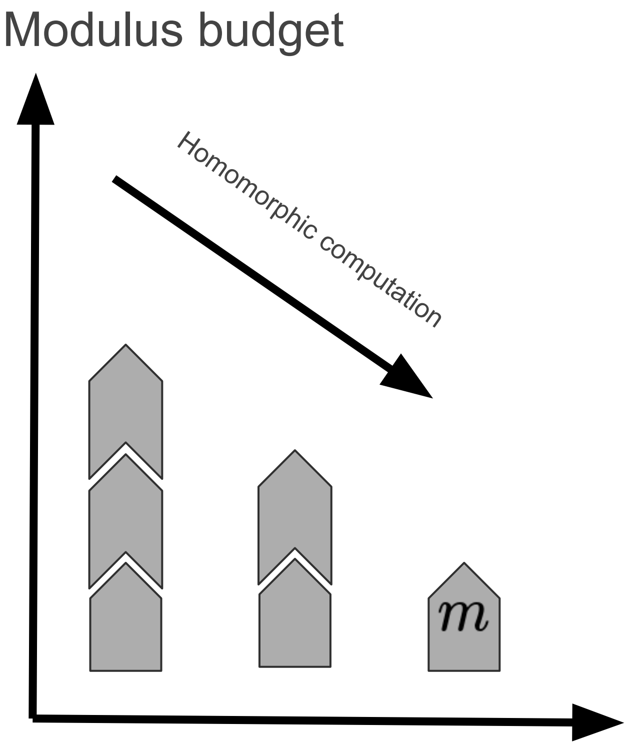
Different HE schemes, different native data types:

- Finite fields: **BGV/BFV**
- Integers: **TFHE**
- Floating point: **CKKS**

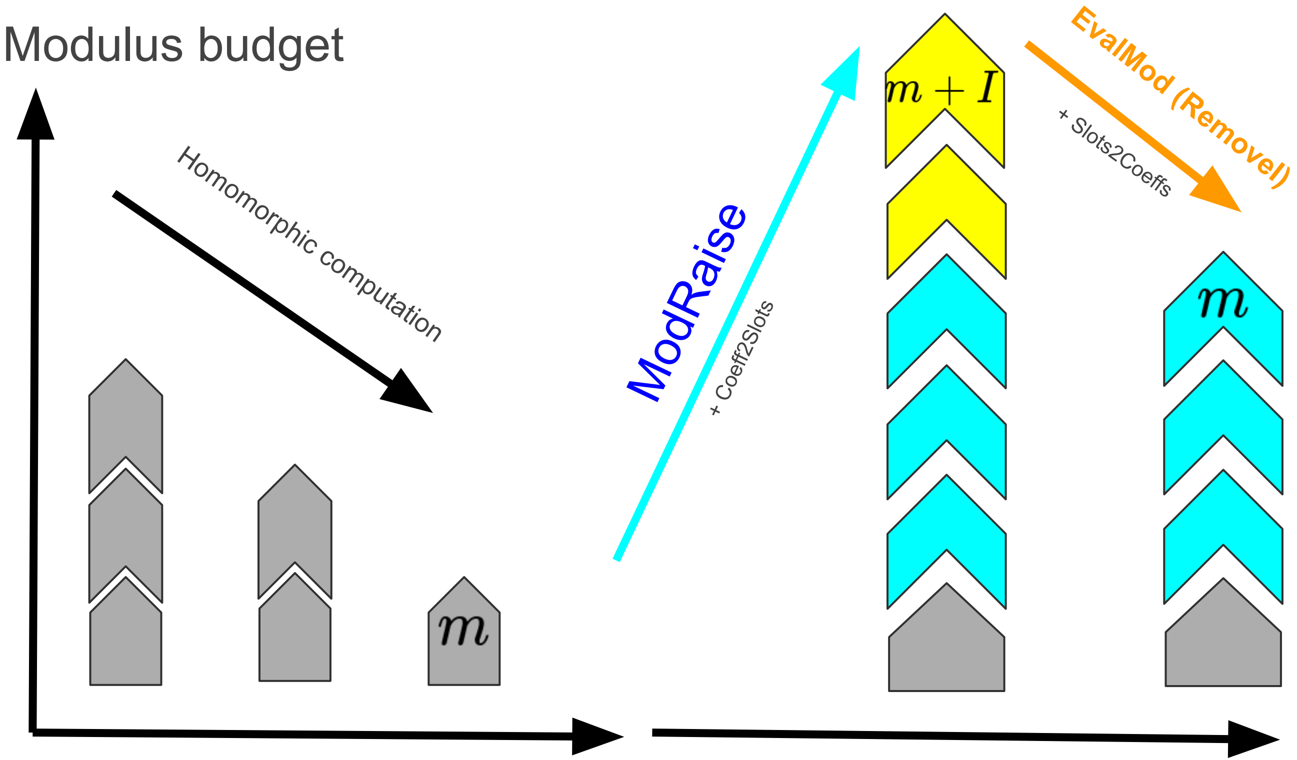


computation budget

CKKS bootstrapping for approximate numbers



CKKS bootstrapping for approximate numbers



FHE Schemes	Plaintext space	Bootstrapping task	**Functional bootstrapping**	Strength
DM/CGGI (2015)	Small integer	Decrease LWE error	✅	Low latency
CKKS (2017)	$\mathbb{C}^{N/2}$	Increase modulus	❌	High throughput

FHE Schemes	Plaintext space	Bootstrapping task	**Functional bootstrapping**	Strength
DM/CGGI (2015)	Small integer	Decrease LWE error	✅	Low latency
CKKS (2017)	$\mathbb{C}^{N/2}$	Increase modulus	❌	High throughput
discrete-CKKS (2024)	Small integers	Decrease LWE error and increase modulus	🆕 ✅	High throughput

SI-BTS design

Building blocks of SI-BTS

Discrete computation over CKKS

SI-BTS relies on 3 ingredients to functional bootstrap encrypted integers:

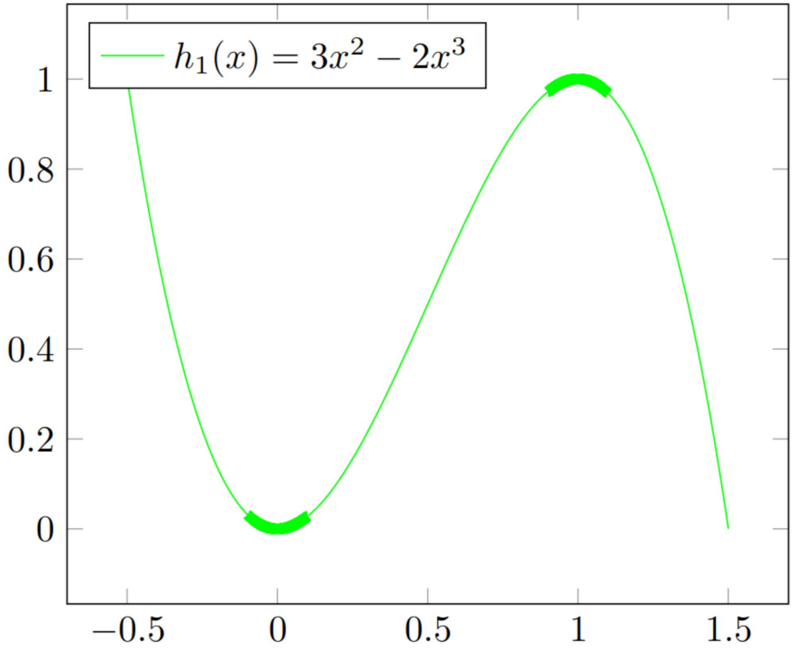
1. **Ring-packing** from LWE to RLWE [BCKPS23]
2. **Hermite interpolation** (similar to BLEACH technique) to decrease the LWE error and evaluate a LUT
3. **IntRootBoot**: a CKKS bootstrap mapping integers to roots of unity

Zero derivative: a very useful tool for discrete CKKS!

BLEACH

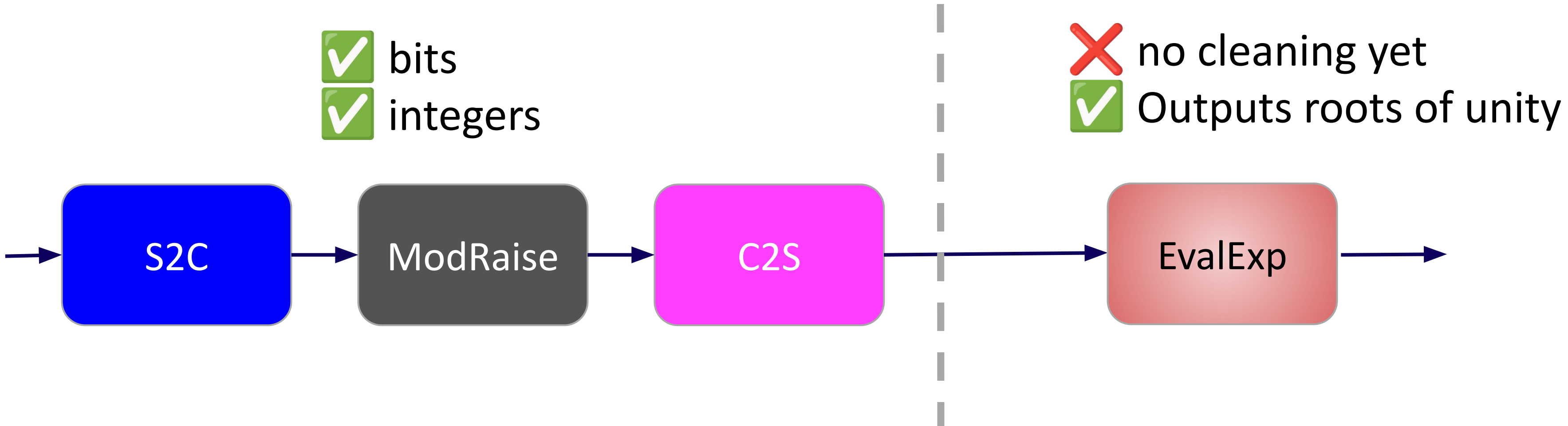
- Encrypts bits as CKKS approximate numbers
- Cleaning function cleans the LWE noise of bits (**BLEACH**)
- It can be generalized to integers thanks to Hermite interpolation
- Integers: **poor grid** for Hermite interpolation
- Roots of unity: **more numerical stability**

Remark: CKKS bootstrapping is very convenient to map integers to complex roots of unity



IntRootBoot

Linear computation / nonlinear computation



This construction is similar to the scheme switching from TFHE to CKKS of CHIMERA [BGGJ19].

Why EvalExp

Benefits

Pros of EvalExp

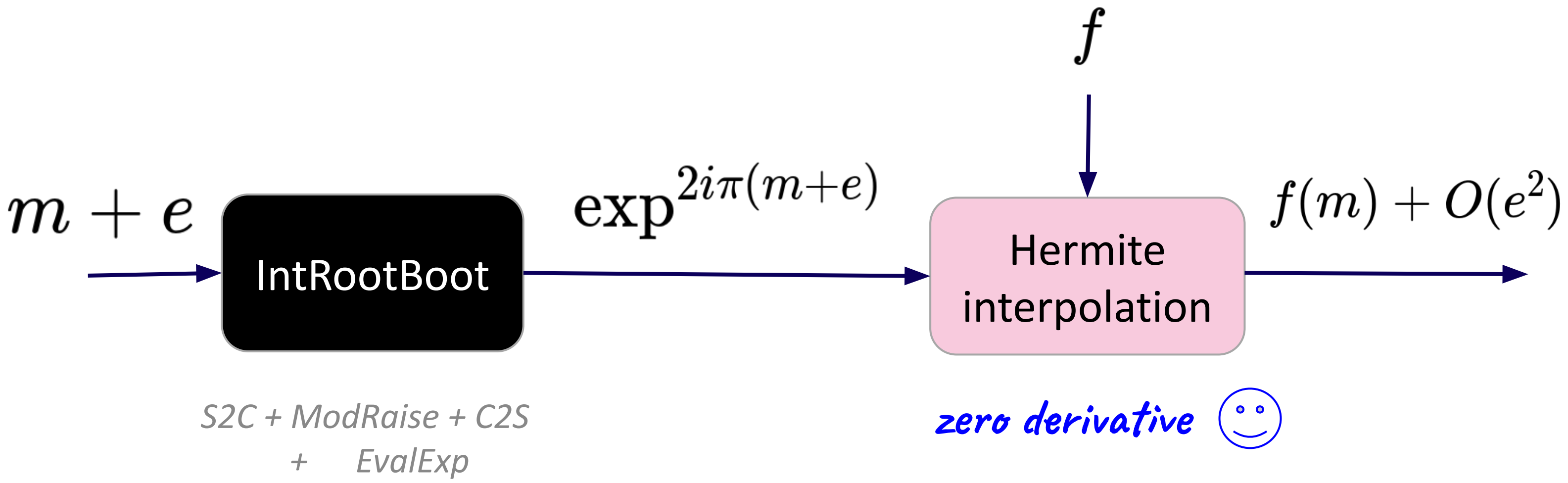
- ❑ Periodic (remove I part)
- ❑ It outputs roots of unity
- ❑ Reduced modulus consumption (no gap between the scaling factor and the bottom modulus q0)

Pros of roots of unity

- ❑ Numerical stability of interpolation (Lagrange, Hermite)
- ❑ Conjugation
- ❑ Efficient bits extraction

Bootstrapping small integers

SI-BTS!



Remark: EvalExp + Hermite interpolation = Trigonometric Hermite interpolation
Remark: cancelling higher order derivative is possible (with more computation)

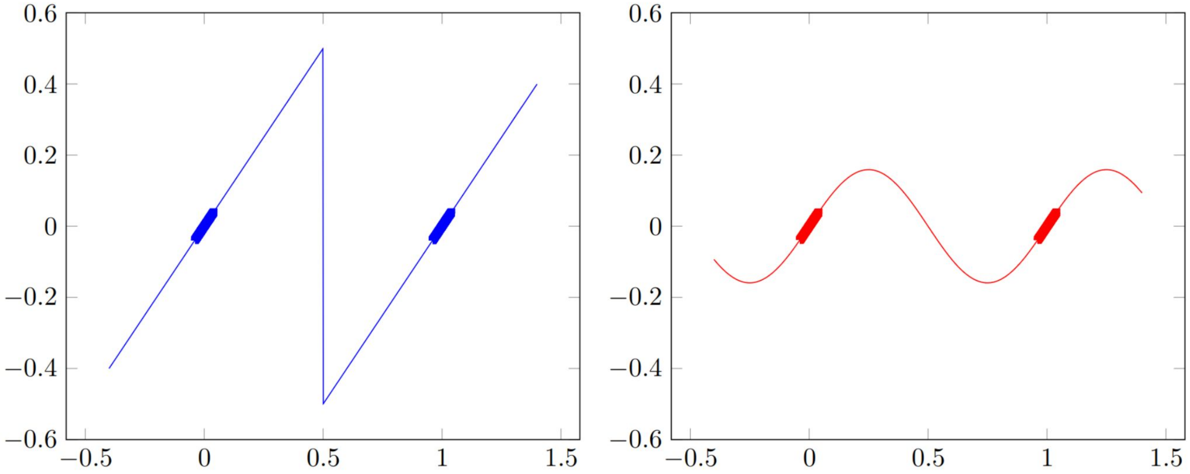
Modulus engineering of SI-BTS

The modulus gap in usual CKKS bootstrapping

Removel

ModRaise: $\Delta \cdot x \to \Delta \cdot z + q_0 \cdot I$ for small I

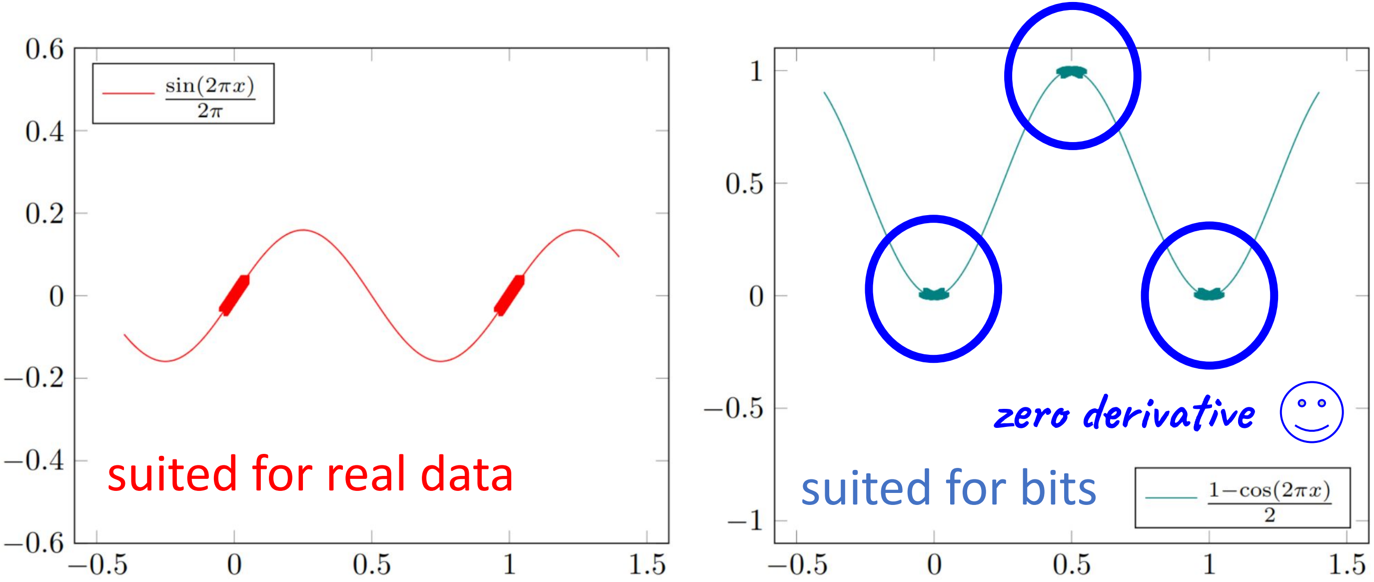
Removel: homomorphic modulo implemented as an approximation of sine.



$$\frac{q_0}{\Delta} \simeq 2^{10}$$

BinBoot: removing the modulus gap

Saves a lot of modulus consumption during bootstrapping!

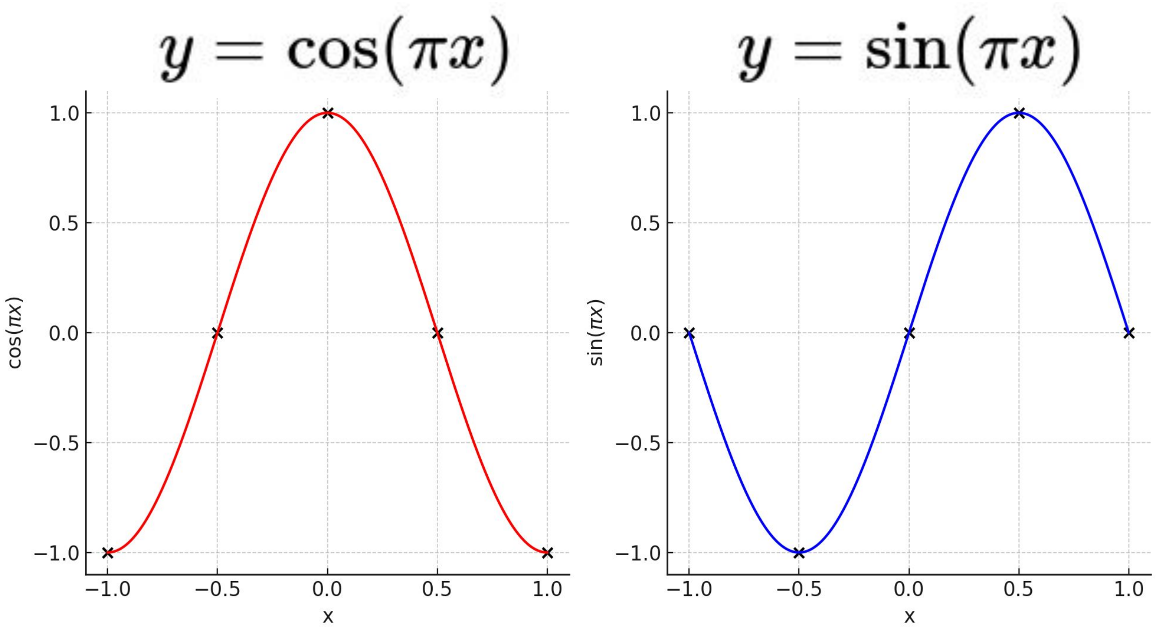


$$\frac{q_0}{\Delta} \simeq 2^{10} \longrightarrow \frac{q_0}{\Delta} \simeq 2$$

EvalExp = cos + i sine

Periodic function

No modulus gap in **EvalExp**



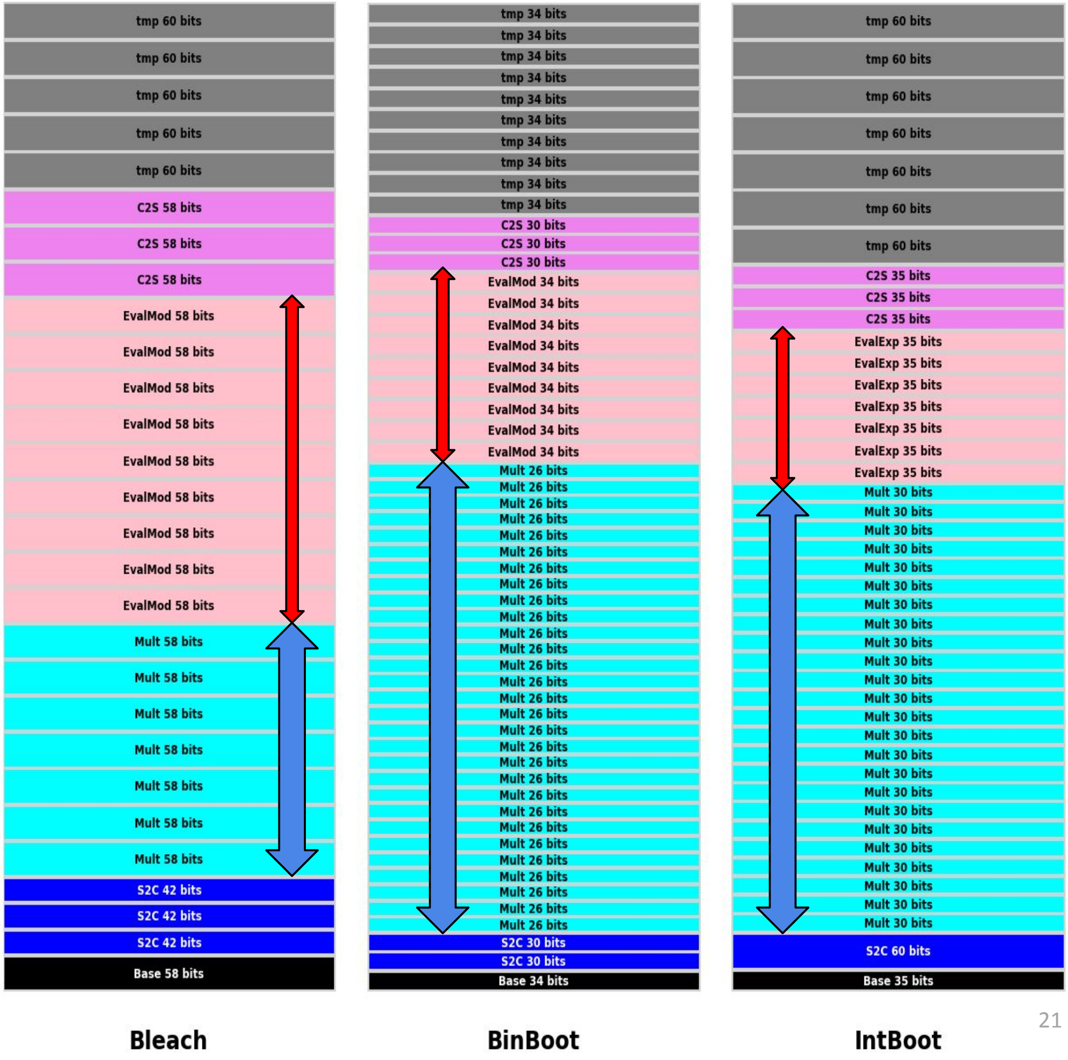
Bootstrapping Small Integers with CKKS

Modulus Engineering

Scaling factors tailored to precision need.

More multiplicative levels:

- Less frequent bootstrapping
- Improved throughput

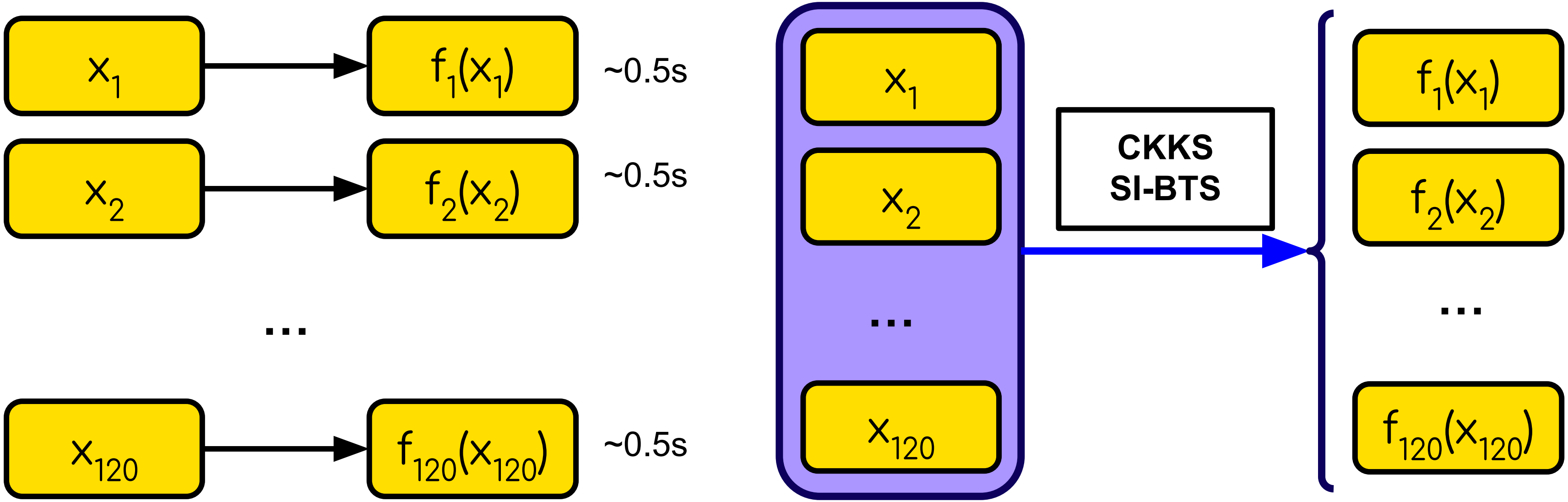


HEAAN CRYPTO LAB

Applications of SI-BTS

Bypassing TFHE for 8-bit integers

Functional bootstrap of 120 integers

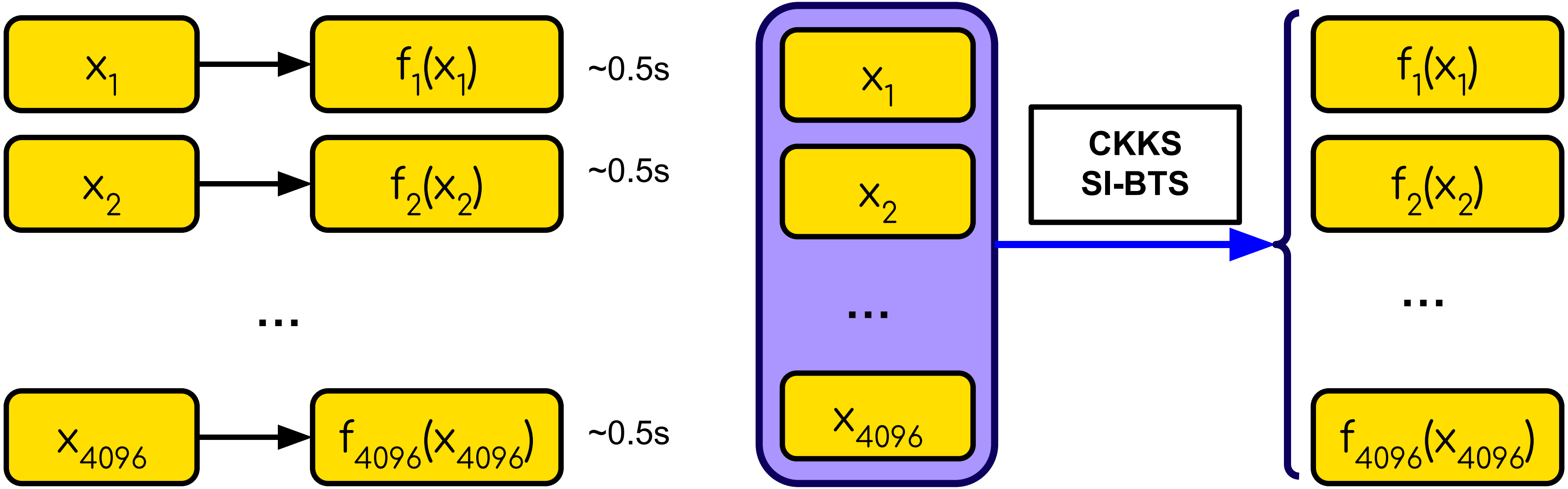


120 TFHE BTS: 1 minute

SI-BTS: 15.4s

Bypassing TFHE for 8-bit integers

Functional bootstrap of 4096 integers



half an hour

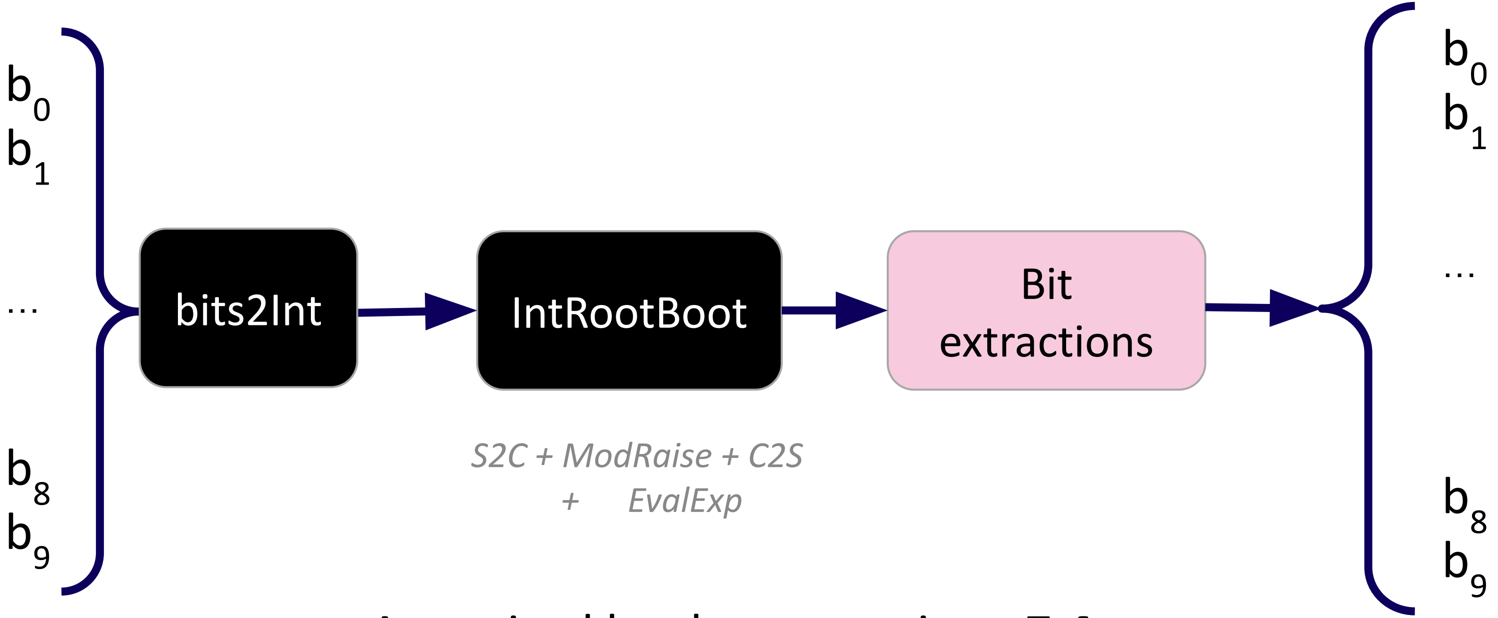
still 15.4s

Experimental results

Table

	Number of input LWE ciphertexts	Integer bootstrapping	Amortized time
[Zam24]	1	0.5s	0.5s
[LW23] 9-bit integers	2^{15}	220s	6.7ms
This work	2^{12}	15s	3.8ms

Batch Bits Bootstrapping



Amortized boolean gate time: **7.4 µs**

Conclusion:

- A new CKKS bootstrapping method **SI-BTS** specifically optimized for integers
- Building blocks are CHIMERA scheme switching and Hermite interpolation
- A batch bits bootstrapping with even higher throughput than [BCKS24]

Amortized time for evaluating a **8-bit LUT** on encrypted integer:	Amortized boolean gate time:
$3.8ms$	$7.4\mu s$

Questions