A White-Box Bootstrapping Approach for High Precision Comparison Over Homomorphic Encryption

Deokhwa Hong [1] Heesoo Lee [1] Young-Sik Kim [2] Yongwoo Lee[1]

[1]Inha University, Incheon, Republic of Korea [2]Daegu Gyeongbuk Institute of Science and Technology, Daegu, Republic of Korea

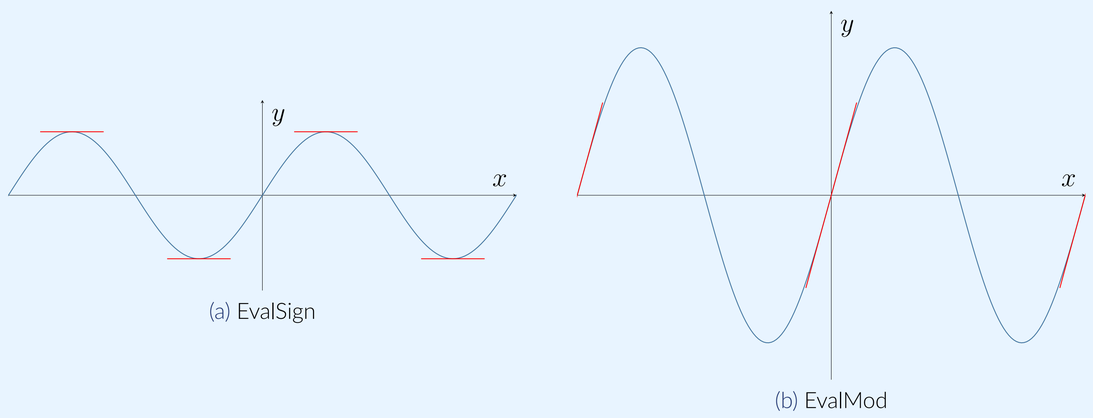
Summary

- **Problem**
  Sign evaluation in CKKS [1] requires intermediate bootstrapping, which introduces considerable noise, harming convergence and degrading accuracy.
- **Key Insight**
  Since $\mathrm{sign}(x) \in \{-1, 1\}$, we can treat CKKS messages as *discrete* and exploit this discrete nature (discrete-CKKS [2, 3, 4, 5]).
- **Sign Bootstrapping**
  We replace EvalMod with EvalSign, which simultaneously eliminates lifting noise and achieves quadratic noise reduction.
- **Results**
  The proposed method achieves **40-bit precision** (vs. 20-bit in prior work), requires less depth, and is numerically more stable.

Sign Bootstrapping: Key Idea

We replace EvalMod with $\mathrm{EvalSign}(x) = \sin(2\pi x)$ in bootstrapping:

- The periodicity of $\sin(2\pi x)$ naturally eliminates lifting noise (multiples of q).
- $\sin(2\pi x)$ reduces the noise introduced in sign messages quadratically.



(a) EvalSign (b) EvalMod

Theorem (Noise Reduction in Sign Bootstrapping)

Let $z \in \{-1, 1\}$, τ be a real number, and let EvalSign$(x) = \sin(2\pi x)$. Then, the following inequality holds:

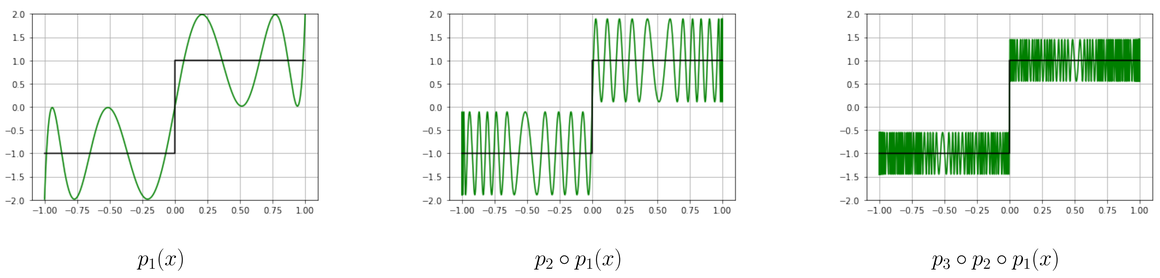
$$\left|\mathit{EvalSign}\left(\frac{z+\tau}{4} + I\right) - z\right| \leq \frac{\pi^2}{8}\tau^2,$$

where $I \in \mathbb{Z}$.

Proposed: Hybrid Approach for High Precision Comparison

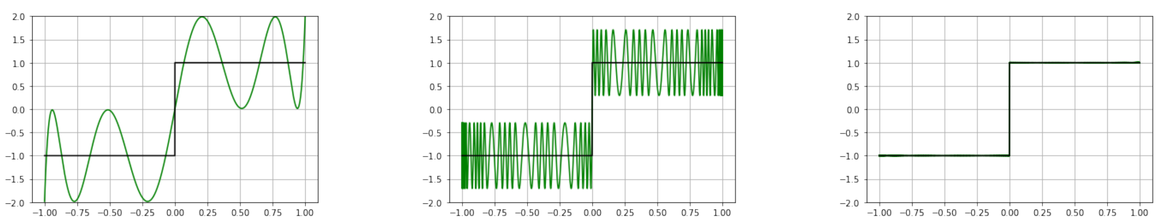
Goal: Achieve high-precision sign evaluation by combining fast convergence and numerical stability.

- **Minimax composition** [6]: Rapid initial convergence over $[0.001, 1]$, but suffers from numerical instability at high degrees.
- **Cleaning functions** [7]: Numerically stable via low-degree polynomials, but converge slowly in small magnitude messages, i.e., 0.001.
- **Sign Bootstrapping**: Sign bootstrapping provides a level of composition (or cleaning) while bootstrapping for free, which enables fast convergence.
- **Hybrid**: Employ minimax composition in the early stages for fast coarse approximation (∼10-bit at depth 15), then switch to cleaning functions for numerically stable convergence (∼40-bit at depth 33).



$p_1(x)$ $p_2 \circ p_1(x)$ $p_3 \circ p_2 \circ p_1(x)$

(a) Standard Minimax Composition



$p_1(x)$ $p_2'' \circ p_1(x)$ $p_3'' \circ p_2'' \circ p_1(x)$

(b) With Sign Bootstrapping

Implementation Results

- **Environment**: Intel Core i9-14900K, 64GB RAM, NVIDIA RTX 4090 GPU, and the desiloFHE library.

Table 1. Parameter set achieving 128-bit security.

$\log_2 N$	$\log_2 QP$	$\log_2 q_0$	MULT	StC	EvalSign	CtS	$\log_2 P$
16	1650	60	54×10	54×3	54×9	54×3	60×4

Table 2. Precision and runtime comparison of bootstrapping procedures.

Applied Bootstrap		Precision $\mathcal{D}_0$	$\mathcal{D}_{0.1}$	$\mathcal{D}_{0.2}$	$\mathcal{D}_{0.3}$	Measured Runtime
Classical	EvalRound	20.7 bits	4.3 bits	3.3 bits	2.7 bits	1.16 s
	StC-First	9.3 bits	4.3 bits	3.3 bits	2.7 bits	1.04 s
Proposed	$\mathrm{SignBoot}_p$	31.6 bits	7.9 bits	5.9 bits	4.8 bits	1.06 s
	$\mathrm{SignBoot}_r$	25.3 bits	7.9 bits	5.9 bits	4.8 bits	0.89 s

Table 3. Depth–precision trade-off comparison. (*) denotes sign bootstrapping applied.

Applied Methods	Precision [0.001, 0.25]	[0.25, 0.5]	[0.5, 0.75]	[0.75, 1]	Consumed Depth
Minimax	1.1 bits	1.1 bits	1.1 bits	1.1 bits	10
Minimax*	1.7 bits	1.7 bits	1.7 bits	1.7 bits	10
Clean	2.5 bits	20.8 bits	13.8 bits	15.1 bits	10
Clean*	3.2 bits	31.0 bits	26.4 bits	28.1 bits	10
Hybrid	1.1 bits	1.1 bits	1.1 bits	1.1 bits	10
Minimax	10.2 bits	20.8 bits	20.8 bits	20.8 bits	20
Minimax*	10.7 bits	31.0 bits	31.1 bits	31.1 bits	20
Clean	5.4 bits	39.4 bits	40.5 bits	40.4 bits	20
Clean*	6.7 bits	40.4 bits	40.6 bits	40.5 bits	20
Hybrid	12.1 bits	40.5 bits	40.5 bits	40.5 bits	21
Minimax	9.1 bits	20.7 bits	20.8 bits	20.8 bits	30
Clean	8.3 bits	40.8 bits	40.4 bits	40.5 bits	30
Clean*	10.3 bits	41.0 bits	40.5 bits	40.5 bits	30
Hybrid	30.1 bits	40.5 bits	40.5 bits	40.5 bits	31
Hybrid	40.5 bits	40.5 bits	40.5 bits	40.5 bits	33

Applications

- **ResNet20** (Table 4): Sign bootstrapping reduces inference runtime by 17.6%.
- **Sorting** (Table 5): Bitonic sort with sign bootstrapping achieves higher precision and fewer bootstrapping operations, reducing runtime by ∼25%.
- **Top-k** (Table 6): $\mathrm{SignBoot}_p$ achieves up to 28.4-bit precision (vs. 14.8-bit with EvalRound), while $\mathrm{SignBoot}_r$ achieves the fastest runtime.

Table 4. ResNet20 inference runtime with and without sign bootstrapping.

EvalRound	$\mathrm{SignBoot}_p$	Reduction
5h 52m 15s	4h 50m 6s	17.6%

Table 5. Precision and runtime comparison of sorting.

Number of Elements	EvalRound Precision	EvalRound Runtime	Ours ($\mathrm{SignBoot}_p$) Precision	Ours ($\mathrm{SignBoot}_p$) Runtime	Precision Gain	Runtime Reduction
2^{10}	21.3 bits	293.1 s	28.5 bits	223.7 s	25.3%	23.7%
2^{11}	23.2 bits	352.7 s	25.7 bits	265.0 s	9.7%	24.9%
2^{12}	23.6 bits	417.5 s	26.7 bits	309.6 s	13.1%	25.8%
2^{13}	25.1 bits	482.0 s	27.3 bits	357.7 s	8.1%	25.8%
2^{14}	23.0 bits	555.6 s	26.8 bits	409.6 s	14.2%	26.3%
2^{15}	25.0 bits	636.3 s	29.2 bits	464.7 s	14.4%	27.0%

Table 6. Runtime and precision comparison of the top-k algorithm.

Applied Bootstrap		4-to-2 Precision	4-to-2 Runtime	4-to-2 Amortized	8-to-2 Precision	8-to-2 Runtime	8-to-2 Amortized
Classical	EvalRound	14.8 bits	41.7 s	1.273 ms	12.1 bits	137.2 s	4.187 ms
	StC-First	9.8 bits	32.5 s	0.991 ms	8.6 bits	106.2 s	3.24 ms
Proposed	$\mathrm{SignBoot}_p$	28.4 bits	32.9 s	1.003 ms	25.6 bits	108.4 s	3.309 ms
	$\mathrm{SignBoot}_r$	22.7 bits	21.6 s	0.661 ms	25.2 bits	69.9 s	2.132 ms

References

[1] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. pages 409–437, 2017.

[2] Heewon Chung, Hyojun Kim, Young-Sik Kim, and Yongwoo Lee. Amortized large look-up table evaluation with multivariate polynomials for homomorphic encryption. Cryptology ePrint Archive, Report 2024/274, 2024.

[3] Nir Drucker, Guy Moshkowich, Tomer Pelleg, and Hayim Shaul. BLEACH: Cleaning errors in discrete computations over CKKS. 37(1):3, January 2024.

[4] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, and Damien Stehlé. Bootstrapping bits with CKKS. pages 94–123, 2024.

[5] Youngjin Bae, Jaehyung Kim, Damien Stehlé, and Elias Suvanto. Bootstrapping small integers with CKKS. pages 330–360, 2024.

[6] Eunsang Lee, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. Minimax approximation of sign function by composite polynomial for homomorphic comparison. *IEEE Transactions on Dependable and Secure Computing*, 19(6):3711–3727, 2022.

[7] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. pages 221–256, 2020.