Evaluating Larger Lookup Tables using CKKS

Jules Dumezy, Andreea Alexandru, Yuriy Polyakov,

Pierre-Emmanuel Clet, Olive Chakraborty, Aymen Boudguiga

Motivation and Contributions

Context: The CKKS scheme traditionally evaluates smooth functions on approximate values. Recent discrete CKKS trends process integers in batch using specific polynomials.

The Bottleneck: Current amortized functional bootstrapping methods (e.g., Alexandru et al. [Crypto'25], Bae et al. [Asiacrypt'24]) require evaluating a degree P polynomial. This is highly impractical and inefficient for large P.

Our Contribution: We propose two new algorithms that drastically slash runtime and memory complexity for large Lookup Tables (LUTs). This enables practical evaluation starting at 10 bits and scaling to 20+ bits.

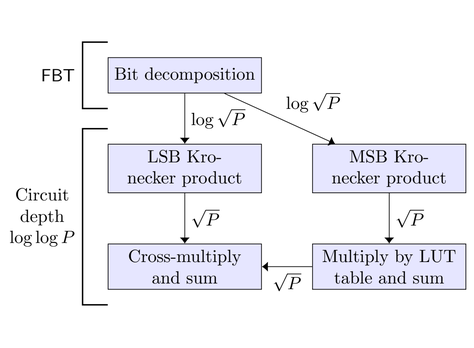
Method 1: BMT-FBT

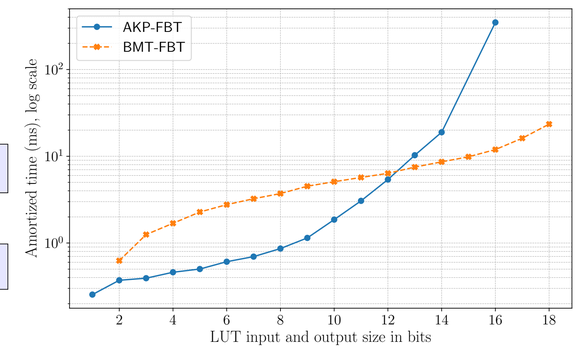
Goal: Reduce multiplicative depth to allow smaller parameters and break down overall complexity.

Intuition: Adapt the CMux tree approach (used in DM/CGGI circuit bootstrapping) for the specific constraints of CKKS.

Mechanism: Extract a one-hot indicator from a ciphertext's binary digits to select a LUT value. We emulate large multiplexers using binary multiplexers, combining them via a tree computed with CKKS leveled operations.

Optimization: Specific techniques are introduced to tightly control noise growth and computational costs.



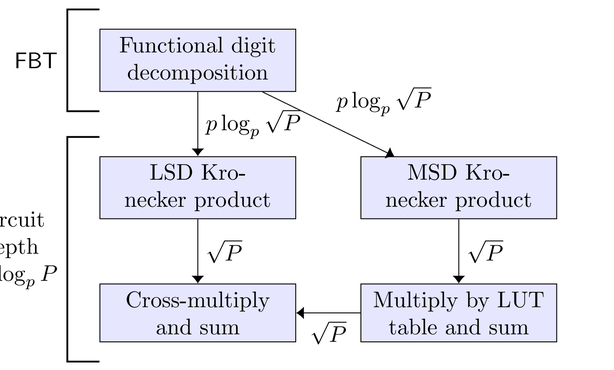


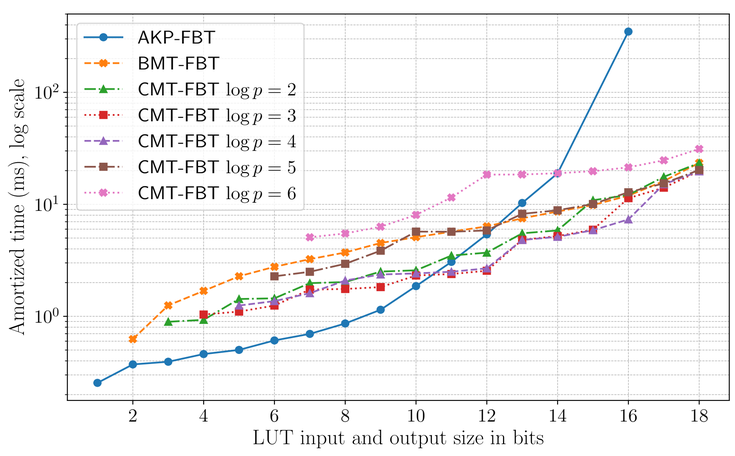
Method 2: CMT-FBT

Intuition: Choose a decomposition basis $p > 2$ and compute indicators with a small LUT (size p), and perform the Kronecker products and vector-matrix-vector product in the same way. We trade a binary multiplexer tree for a shorter but wider **p-ary multiplexer tree**.

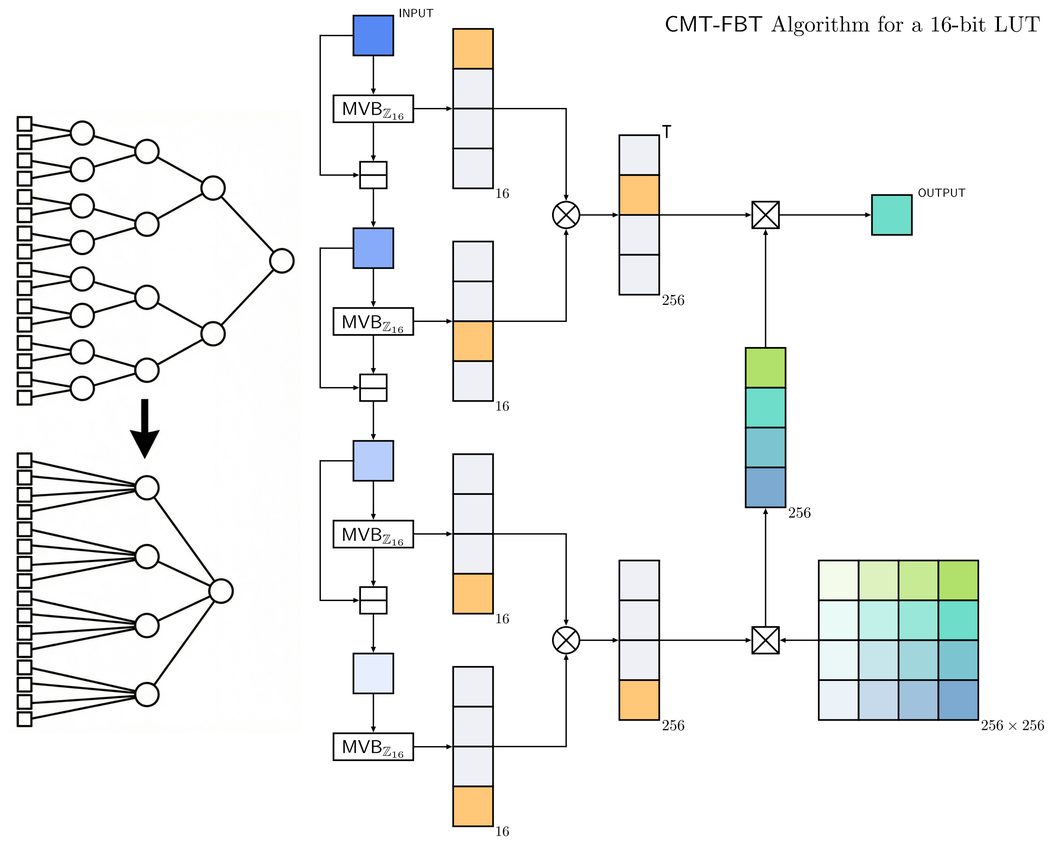
Mechanism: The indicators can be computed through **functional digit decomposition**, which leverages *multi-value bootstrapping* to evaluate a LUT while extracting digits.

→ We reduce the depth of the circuit from $\mathrm{loglog}P$ to $\mathrm{loglog}_p P$.





Example



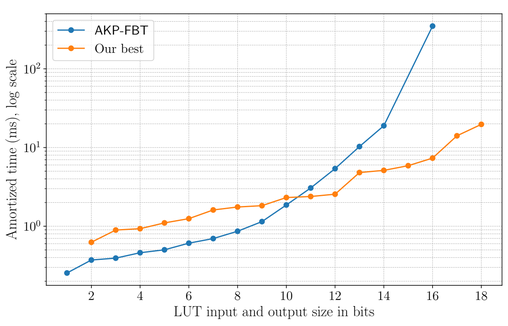
Key Results & Benchmarks

Implementation: An OpenFHE proof-of-concept shows drastic improvements over state-of-the-art techniques.

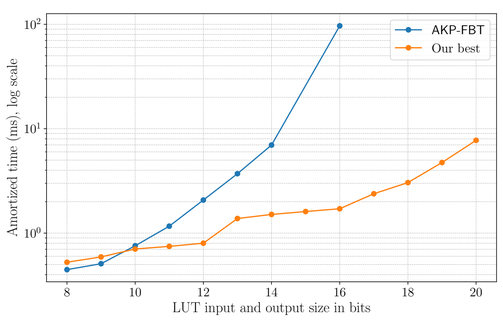
Performance: Superior amortized time begins at 11 bits (10 bits when parallelized). We observe massive speed-ups, such as a 47.5x improvement for a 16-bit LUT, and evaluate LUTs up to 24 bits.

$\log P$	10	12	14	16
AKP-FBT Peak RAM (GB)	22.9	27.4	32.3	110.2
BMT-FBT Peak RAM (GB)	22.4	22.6	**24.8**	**26.2**
CMT-FBT Peak RAM (GB)	**13.8**	**13.9**	27.3	28.5

Method	Ring dimension	Amtz. time (ms)	Latency
AKP-FBT	2^{17}	349	12h40
BMT-FBT	2^{16}	11.9	13 min
CMT-FBT	2^{16}	7.34	8 min
CMT-FBT (MT)	2^{16}	1.71	< 2 min



Single Thread Amortized Time



Multi Thread Amortized Time

Conclusion & Future work

This work proves that functional bootstrapping on large lookup tables is now practically feasible in CKKS, unlocking new capabilities for evaluating **arbitrary discontinuous functions** on large integers.

Future Work:

- Explore a non-black-box approach of functional bootstrapping in CKKS to further improve performance.
- Apply some of the ideas to TFHE's bootstrapping algorithms.

QR Code to the full paper (eprint 2025/1301).

To be presented at CHES'26 in October.

Will be included in a future release of OpenFHE.