Efficient Full Domain Functional Bootstrapping from Recursive LUT Decomposition

Intak Hwang, **Shinwon Lee**, Seonhong Min, Yongsoo Song

Seoul National University

Fully Homomorphic Encryption

- **Fully Homomorphic Encryption(FHE)** enables direct computations on encrypted data.
- One of the most powerful tools for secure computation. (e.g. Privacy Preserving ML)

Various **FHE** schemes have been proposed based on the (R)LWE problem,
such as **BGV**, **BFV**, **CKKS** and **TFHE**

TFHE – Fully Homomorphic Encryption over the Torus

- While most **FHE** schemes focus on addition and multiplication, **TFHE** supports arbitrary Boolean gate evaluation.

- Key advancement: **Programmable Bootstrapping**
  - Supports **multi-bit ciphertexts**
  - Enables complex **lookup table(LUT)** evaluation without extra computational cost
  - LUT should satisfy negacyclic condition

Programmable Bootstrapping – Negacyclic constraint

- The test vector $tv \in \mathbb{Z}_q[X]/(X^N + 1)$ encodes the LUT values as its coefficients.
  - $tv = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-1} X^{N-1}$

Programmable Bootstrapping – Negacyclic constraint

- The test vector $tv \in \mathbb{Z}_q[X]/(X^N + 1)$ encodes the LUT values as its coefficients.
  - $tv = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-1} X^{N-1}$

- Multiply tv by X^{-m} to shift the desired LUT value to the constant term. **(BlindRotate)**
  - $X^{-m} \cdot tv = a_m + a_{m+1} X + a_{m+2} X^2 + \cdots - a_{m-1} X^{N-1}$
  - Then, we extract the constant term. **(SampleExtract)**

Programmable Bootstrapping – Negacyclic constraint

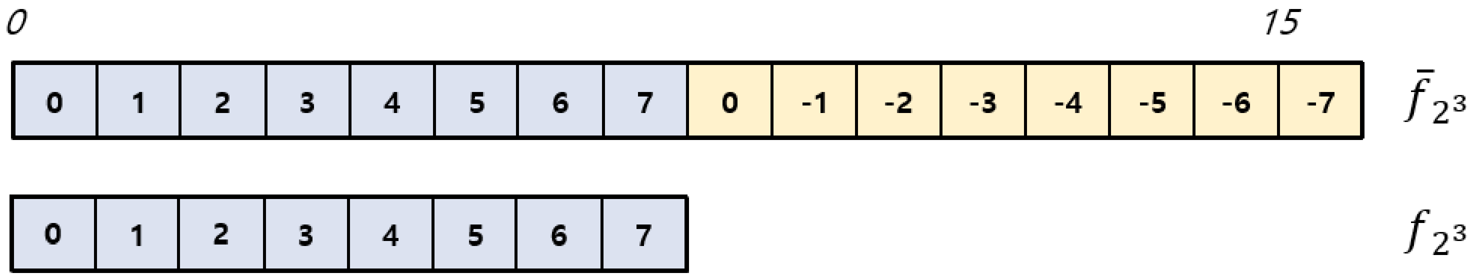
- The test vector $tv \in \mathbb{Z}_q[X]/(X^N+1)$ encodes the LUT values as its coefficients.
  - $tv = a_0 + a_1 X + a_2 X^2 + \cdots + a_{N-1} X^{N-1}$

- Multiply tv by X^{-m} to shift the desired LUT value to the constant term. **(BlindRotate)**
  - $X^{-m} \cdot tv = a_m + a_{m+1} X + a_{m+2} X^2 + \cdots - a_{m-1} X^{N-1}$
  - Then, we extract the constant term. **(SampleExtract)**

Problem : $X^{-m+N} \cdot tv = -a_m - a_{m+1} X - a_{m+2} X^2 - \cdots + a_{m-1} X^{N-1}$

- The lookup table (LUT) should satisfy the **negacyclic** condition.
  - Evaluated **LUT** $f_N: \mathbb{Z}_{2N} \rightarrow \mathbb{Z}_q$ should satisfy $f_N(i+N) = -f_N(i)$ for $i \in [0, N)$

Programmable Bootstrapping – Negacyclic constraint



Single Negacyclic Bootstrapping

FDFB (Bootstrapping ≥ 2)

- **Full Domain Functional bootstrapping (FDFB**)
  - Supports arbitrary LUT evaluation without the negacyclic restriction

- Existing FDFB schemes require more than **two** bootstrappings
  - ~2× latency compared to single bootstrapping.

Our Contribution

- **Propose a novel FDFB scheme based on a LUT decomposition structure**
    - Up to ~2× faster than previous FDFB schemes
    - Negligible parameter overhead
    - Highly parallelizable

Decomposition of lookup table

- **Key observation : LUT can be decomposed into smaller LUTs**
  - For $f_{2^m}: \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_{\mathrm{q}}, \quad f_{2^m}(i) = f_{2^{m-1}}\left([i]_{2^{m-1}}\right) + \bar{f}_{2^{m-1}}(i)$
    - $f_{2^{m-1}}: \mathbb{Z}_{2^{m-1}} \rightarrow \mathbb{Z}_{\mathrm{q}}$, Full domain LUT
    - $\bar{f}_{2^{m-1}}(i): \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_{\mathrm{q}}$, Negacyclic LUT

	0															15	
	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	f_{2^4}
=	-0.5	-1	-1.5	-2	-2.5	-3	-3.5	-4	0.5	1	1.5	2	2.5	3	3.5	4	$\bar{f}_{2^3}$
+	0.5	1	1.5	2	2.5	3	3.5	4	0.5	1	1.5	2	2.5	3	3.5	4	f_{2^3}

Decomposition of lookup table

- **Key observation : LUT can be decomposed into smaller LUTs**
  - For $f_{2^m}: \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_q$, $\quad f_{2^m}(i) = f_{2^{m-1}}([i]_{2^{m-1}}) + \bar{f}_{2^{m-1}}(i)$
    - $f_{2^{m-1}}: \mathbb{Z}_{2^{m-1}} \rightarrow \mathbb{Z}_q \quad f_{2^{m-1}}(i) = \frac{1}{2}\{f_{2^m}(i) + f_{2^m}(i + 2^{m-1})\}$
    - $\bar{f}_{2^{m-1}}(i): \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_q \quad \bar{f}_{2^{m-1}}(i) = \begin{cases} \frac{1}{2}\{f_{2^m}(i) - f_{2^m}(i + 2^{m-1})\} & (0 \leq i < 2^{m-1}) \\ -\bar{f}_{2^{m-1}}(i - 2^{m-1}) & (2^{m-1} \leq i < 2^m) \end{cases}$

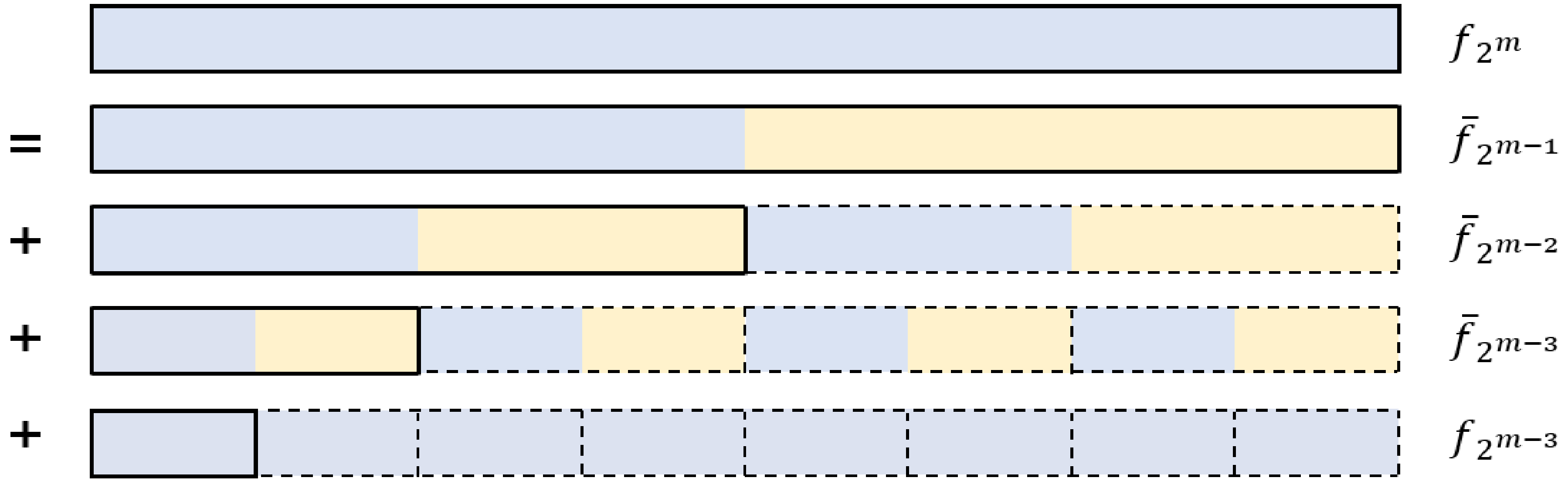
	0															15	
	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	f_{2^4}
=	-0.5	-1	-1.5	-2	-2.5	-3	-3.5	-4	0.5	1	1.5	2	2.5	3	3.5	4	$\bar{f}_{2^3}$
+	0.5	1	1.5	2	2.5	3	3.5	4	0.5	1	1.5	2	2.5	3	3.5	4	f_{2^3}

Decomposition of lookup table

- **Key observation : LUT can be decomposed into smaller LUTs**
  - For $f_{2^m} : \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_q$, $\quad f_{2^m}(i) = f_{2^{m-1}}([i]_{2^{m-1}}) + \bar{f}_{2^{m-1}}(i)$

$$= \sum_{k=m-\mu}^{m-1} \bar{f}_{2^k}([i]_{2^{k+1}}) + f_{2^{m-\mu}}([i]_{2^{m-\mu}})$$

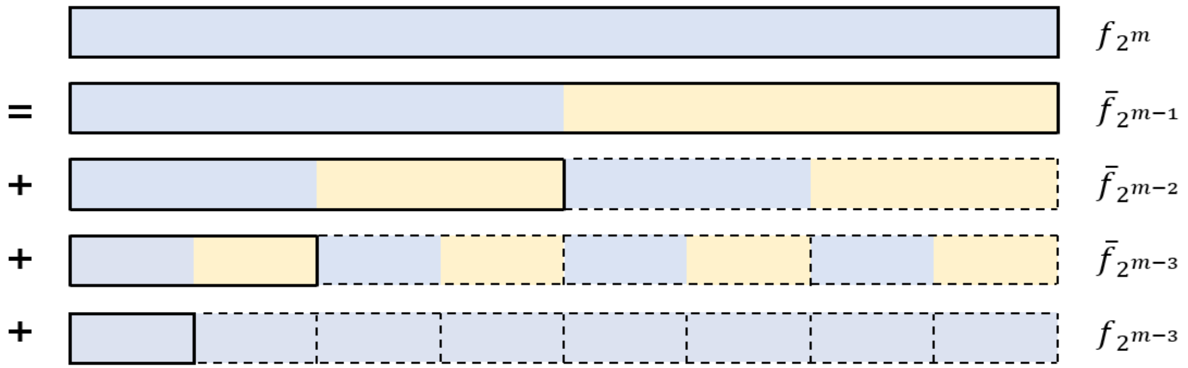
apply decomposition
μ times recursively



Efficient FDFB with LUT Decomposition

$$f_{2^m}(i) = \sum_{k=m-\mu}^{m-1} \bar{f}_{2^k}([i]_{2^{k+1}}) + f_{2^{m-\mu}}([i]_{2^{m-\mu}})$$

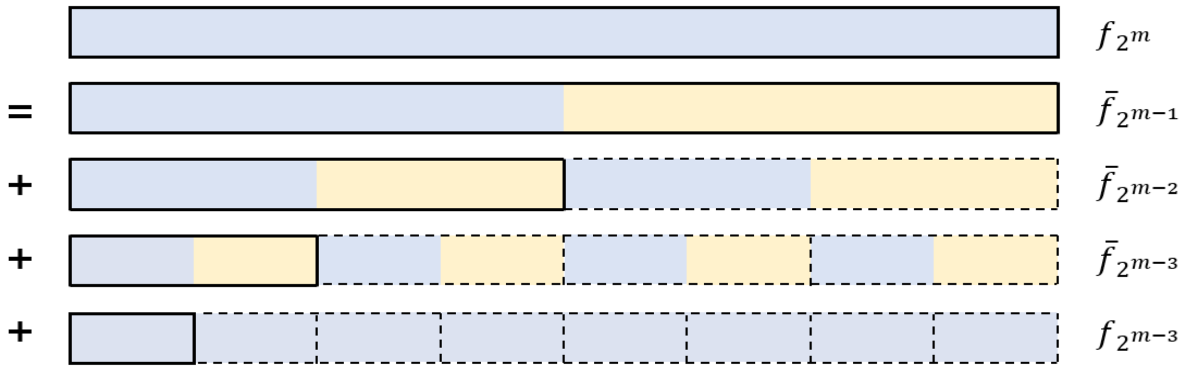
- Evaluate each $\bar{f}_{2^k}$ and $f_{2^{m-\mu}}$ term, and then sum them to obtain f_{2^m}
  - Most of the computations are handled by fast negacyclic bootstrapping.
  - FDFB is applied only to $f_{2^{m-\mu}}$, which has the smallest domain.



Efficient FDFB with LUT Decomposition

$$f_{2^m}(i) = \sum_{k=m-\mu}^{m-1} \bar{f}_{2^k}([i]_{2^{k+1}}) + f_{2^{m-\mu}}([i]_{2^{m-\mu}})$$

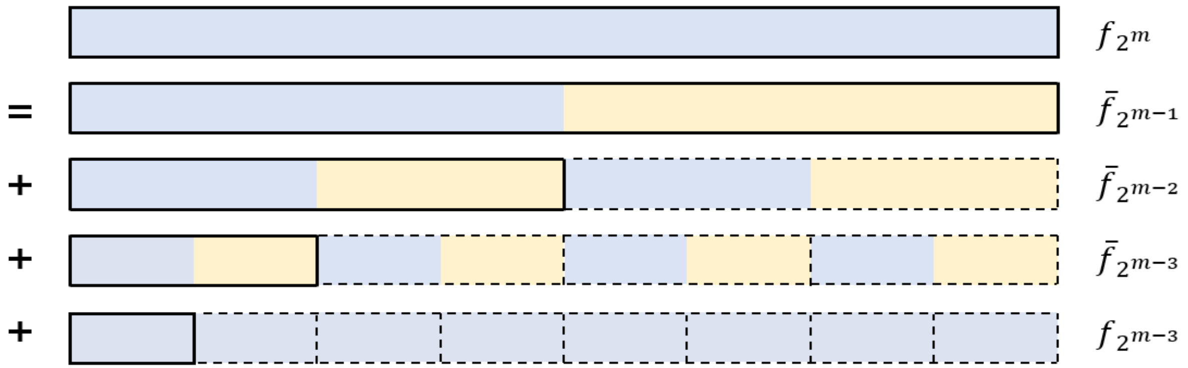
- **Time complexity**
  - Bootstrapping cost grows linearly w.r.t. the size of the LUT
  - Previous: $2 \times 2^m = \mathbf{2^{m+1}}$ unit time
  - Ours: $2^{m-1} + \ \ldots + 2^{m-\mu} + 2 \times 2^{m-\mu} = \mathbf{2^m} + \mathbf{2^{m-\mu}}$ unit time ($\mathbf{2^{m-1}}$ with parallelism)



Efficient FDFB with LUT Decomposition

$$f_{2^m}(i) = \sum_{k=m-\mu}^{m-1} \bar{f}_{2^k}([i]_{2^{k+1}}) + f_{2^{m-\mu}}([i]_{2^{m-\mu}})$$

- **Problem:** Need to maintain distinct evaluation keys for each LUT length
    - RLWE dimension depends on the LUT length



Extended Bootstrapping (EBS)

- **Extended Bootstrapping [LY23] (PKC 2023)**
  - Rewrite polynomial operations in higher dimensions as several independent operations in lower dimensions via a module isomorphism.

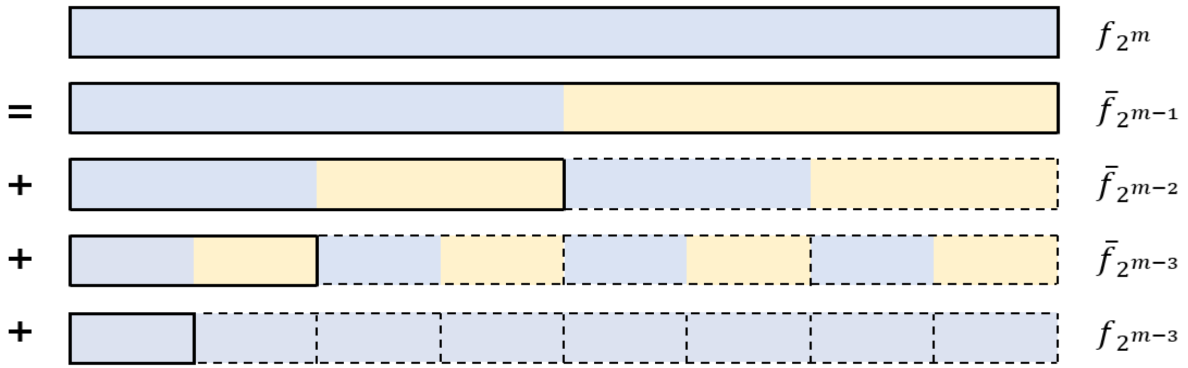
$$\mathbb{Z}_q[X]/(X^{2^m}+1) \to \left(\mathbb{Z}_q[X]/(X^{2^{m-\nu}}+1)\right)^{2^\nu}$$

  - “Simulates” operations over $N = 2^m$ with operations over $N = 2^{m-\nu}$
    - Operations are performed over reduced dimension $2^{m-\nu}$

Optimization via Extended Bootstrapping

$$f_{2^m}(i) = \sum_{k=m-\mu}^{m-1} \bar{f}_{2^k}([i]_{2^{k+1}}) + f_{2^{m-\mu}}([i]_{2^{m-\mu}})$$

- EBS enables all operations to run within a fixed ring dimension.
- **Evaluation Key Size**
  - Key Size grows linearly w.r.t. the RLWE degree
  - Without EBS : $2^{m-1} + \ldots + 2^{m-\mu} = \mathbf{2^m - 2^{m-\mu}}$ unit size
  - With EBS: $\mathbf{2^{m-\mu}}$ unit size



Implementation

- Used the TFHE-go library
  - Single-threaded execution

- Compared with FDFB-Compress (TCHES 2024)

- Evaluated with plaintext modulus p from 2^5 to 2^8
  - $N = 2^{12}$ to 2^{15}

- The decomposition depth μ is set as large as possible while preserving RLWE security.
  - Reduced RLWE dimension $\mathrm{N}/2^{\mu} \geq 2^{11}$

Experimental Results

		$p = 2^5$	$p = 2^6$	$p = 2^7$	$p = 2^8$
Non-EBS	FDFB-Compress	79 ms	297 ms	655 ms	1470 ms
	Ours	59 ms	146 ms	300 ms	648 ms
EBS	FDFB-Compress	70 ms	134 ms	393 ms	823 ms
	Ours	57 ms	91 ms	234 ms	431 ms

Table 3. FDFB performance for each plaintext modulus p.

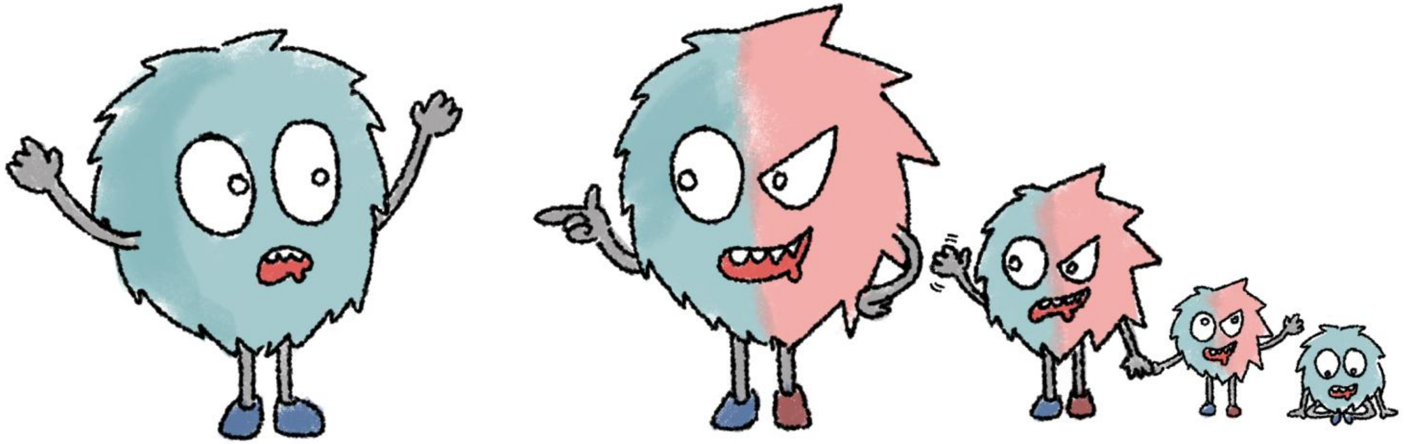
- Up to 1.91× faster than FDFB-Compress
- Only 4.7% slower than a single negacyclic bootstrapping

Experimental Results

		$p = 2^5$	$p = 2^6$	$p = 2^7$	$p = 2^8$
Non-EBS	FDFB-Compress	254 MB	798 MB	1.56 GB	3.12 GB
	Ours	127 MB	598 MB	1.36 GB	2.98 GB
EBS	FDFB-Compress	127 MB		199 MB	
	Ours				

Table 4. Bootstrapping key size for each plaintext modulus.

- Key size growth is efficiently mitigated with EBS
  - No additional key size growth



Thank you for listening!