清華大學
Tsinghua University

Faster BGV Bootstrapping for Power-of-two Cyclotomics through Homomorphic NTT

Shihe Ma, Tairong Huang, Anyu Wang, Xiaoyun Wang

Tsinghua University

Fully Homomorphic Encryption

- FHE enables computation over encrypted data without decryption key
  - Concept by Rivest et al. in 1978
  - First plausible scheme by Gentry in 2009
  - 4 generations of schemes: Gentry's; BGV/BFV; FHEW/TFHE; CKKS

- Bootstrapping: remove noise homomorphically to enable infinite homomorphic computation
- Single Instruction Multiple Data (SIMD) encoding: amortize cost in BGV/BFV/CKKS
- Rings with a power-of-two cyclotomic order are preferred in RLWE schemes
  - Exclusively used by SEAL, OpenFHE, lattigo

Why using power-of-two cyclotomics in BGV/BFV?

1. Fast and easy implementation with Cooley-Tukey NTT
2. Compatible with FHE standard
3. More efficient null-polynomial-based digit removal [MHWW, Eurocrypt 2024]
   - $\Pr\left[|I| > k\sqrt{\frac{h\phi(M)2^{\omega(M)}}{12M}}\right] < \phi(M)\cdot \operatorname{erfc}\left(\frac{k}{\sqrt{2}}\right)$
   - For different M with roughly the same $\phi(M)$, $\frac{\phi(M)2^{\omega(M)}}{M}$ is smaller when M is a power of two → smaller bound on I → null polynomials with lower degrees → faster digit removal
4. Simpler plaintext space structure for BGV/BFV
   - 1-D array or a $2\times\frac{L}{2}$-sized 2-D array

Problem and method overview

- We want to achieve bootstrapping of BGV/BFV that

1. fully exploits the SIMD encoding property
2. uses power-of-two cyclotomic rings
3. is efficient

Chen and Han, Eurocrypt 2018

Halevi and Shoup, JoC 2021

- Such a goal has not been realized because having many slots in a power-of-two ring means

1. having a large plaintext prime p, causing slow digit removal (without the techniques of [MHWW24])
2. the linear transformations during bootstrapping are slow, because
    1. their large dimensions require more computing time
    2. existing acceleration techniques based on decomposed linear transformations works only in non-power-of-two rings

Problem and method overview

- Main idea: decompose SlotToCoeff/CoeffToSlot matrices into the product of NTT matrices
- NTT matrices has much fewer nonzero diagonals → much faster homomorphic evaluation

- Similar techniques have been applied to CKKS bootstrapping [Chen, Chilloti, Song. Eurocrypt'19][Han, Hhan, Cheon, 2019]
- Porting to BGV/BFV is nontrivial because…

Scheme	Slot value	Slot arrangement	NTT type	Linear transformation type
CKKS	Complex number	1D array	Cooley-Tukey	On scalar vectors
BGV/BFV	Finite ring elements	1D or 2D array	Cooley-Tukey or Bruun	On vectors of small scalar vectors

- Other optimizations…
  - Faster linearized polynomial on subfield/subring
  - BSGS tailored for NTT matrices
  - Reordering of linear transformations

Structure of BGV/BFV plaintext space

BGV/BFV FHE schemes

- RLWE based encryption with cyclotomic ring $R_q = \mathbb{Z}_q[X]/(\Phi_M(X))$, plaintext modulus p^r
- Ciphertext format is $(b = -as + p^r e + m, a) \in R_q^2$ for BGV or $\left(b = -as + e + \left\lfloor \frac{q}{p^r} m \right\rceil, a\right) \in R_q^2$ for BFV, with randomness $a \leftarrow R_q$, Gaussian noise $e \in R$, small secret $s \in R$, and message $m \in R_{p^r}$

- *SIMD property.* Plaintext space R_{p^r} is isomorphic to E^L for some Galois ring/field E and integer L
- Supported homomorphic operations on E^L:
  (1) slot-wise addition, (2) slot-wise multiplication, (3) rotation of slots, (4) slot-wise Frobenius automorphism

Plaintext encoding in BGV/BFV

- **Cyclotomic ring factorization**
- Let $N = \phi(M)$
- Case of $r = 1$:
  - $\Phi_M(X) = \prod_{i=0}^{L-1} F_i(X)$, where $\deg(F_i(X)) = \mathrm{ord}_{\mathbb{Z}_M^*}(p)$ is denoted as $d. Ld = N$
  - $F_i(X)$ are monic, irreducible, distinct in $\mathbb{F}_p[X]$, i.e., $\mathbb{F}_p[X]/(F_i(X)) \cong \mathrm{GF}(p^d)$
  - $R_p \cong \prod_{i=0}^{L-1} \mathbb{F}_p[X]/(F_i(X)) \cong \mathrm{GF}(p^d)^L$, each $\mathrm{GF}(p^d)$ position is called a slot
- Case of $r > 1$:
  - Can be obtained from the previous case using Hensel Lifting
  - $R_{p^r} \cong \mathrm{GR}(p^r; d)^L$

Plaintext encoding in BGV/BFV

- Fix a representation of $\mathrm{GR}(p^r; d)$, say $\mathbb{Z}_{p^r}[X]/\left(F_0(X)\right)$. Denote it as E
- $X^N + 1$ splits in E, denote one of the roots of $F_0(X)$ in E as η, then
- Each $F_i(X) = \prod_{j=0}^{d-1} \left(X - \eta^{s_i \cdot p^j}\right)$, and the set $\{s_i\} \subseteq \mathbb{Z}_M^*$ is a representative set of $H = \mathbb{Z}_M^*/\langle p \rangle$

- $\mathrm{Decode}(m) = \left(m(\eta^{s_0}), m(\eta^{s_1}), \ldots, m(\eta^{s_{L-1}})\right): R_{p^r} \to E^L$
- $\mathrm{Encode} = \mathrm{Decode}^{-1}$

Hypercube structure and rotation

- *Example.* $H = \langle g_1, g_2 \rangle$ with $\mathrm{ord}_H(g_i) = d_i$, by setting $s_{i,j} = g_1^i g_2^j$, the slots $\{f(\eta^{s_{i,j}})\}$ of $f(X) \in R_{p^r}$ forms

$$\begin{pmatrix} f(\eta^{s_{0,0}}) & \cdots & f(\eta^{s_{0,d_2-1}}) \\ \vdots & \ddots & \vdots \\ f(\eta^{s_{d_1-1,0}}) & \cdots & f(\eta^{s_{d_1-1,d_2-1}}) \end{pmatrix}$$

- Let $g_i^{d_i} \equiv p^{e_i} \bmod M$, Galois automorphism θ_i mapping $\eta \to \eta^{g_i}$ rotates the matrix up or left ($i = 0$ or 1), while the wrapped-around elements additionally go through Frobenius automorphism σ^{e_i} mapping $\eta \to \eta^{p^{e_i}}$
- The i-th dimension is good $\Leftrightarrow$ the rotation is perfect $\Leftrightarrow$ $e_i = 0$

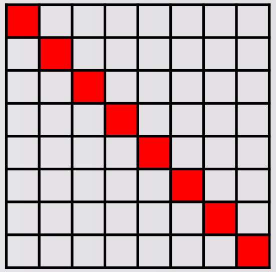
- Rotation by k positions in i-th dimension: $\rho_i^s = \theta_i^s$ or $\rho_i^s = \theta_i^s \cdot \mu_i(s) + \theta_i^{s-d_i} \cdot \mu_i'(s)$ for masks μ_i and μ_i'
- Homomorphic rotations are important in homomorphic linear transformations

Homomorphic linear transformations

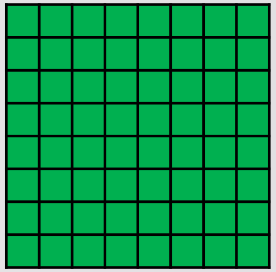
■ = general matrix in $\mathbb{Z}_{p^r}^{d\times d}$

■ = multiply by some element in E

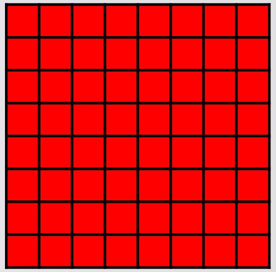
- Intra-slot $\mathbb{Z}_{p^r}$ linear transformation:
  - Computable through linearized polynomials. $f(x) = \sum_{i=0}^{d-1} a_i x^{p^i}$ (1)
  - Realized by homomorphic Frobenius automorphisms $\sigma(x) = x^p$
- Inter-slot 1-D linear transformation along dimension s:
  - E-linear case: $f(x) = \sum_{i=0}^{d_i-1} a_i \cdot \rho_s^i(x)$ (2)
  - $\mathbb{Z}_{p^r}$-linear case: $f(x) = \sum_{i=0}^{d_i-1} \sum_{j=0}^{d-1} a_{i,j} \cdot \sigma^j\left(\rho_s^i(x)\right)$ (3)
  - Each a_i ($a_{i,j}$) is nonzero if the i-th diagonal in the corresponding matrix is nonzero
  - Matrices on each hypercolumn along dimension s



(1)



(2)



(3)

CoeffToSlot/SlotToCoeff as NTT matrices

CoeffToSlot and SlotToCoeff

$$\begin{array}{c}
\mathrm{Enc}(m; p^r) \\
\Big\downarrow \text{Decryption Formula Simplification and Homomorphic Inner Product} \\
\mathrm{Enc}(p^{e-r}m + \epsilon; p^e) \\
\Big\downarrow \text{CoeffToSlot and Unpacking} \\
\mathrm{Enc}([p^{e-r}m_{i\cdot n} + \epsilon_{i\cdot n}, \ldots, p^{e-r}m_{i\cdot n+n-1} + \epsilon_{i\cdot n+n-1}]; p^e) \quad \text{for } i = 0, 1, \ldots, d-1 \\
\Big\downarrow \text{Homomorphic Digit Removal} \\
\mathrm{Enc}([m_{i\cdot n}, \ldots, m_{i\cdot n+n-1}]; p^r) \quad \text{for } i = 0, 1, \ldots, d-1 \\
\Big\downarrow \text{Repacking and SlotToCoeff} \\
\mathrm{Enc}(m; p^r)
\end{array}$$

1. The slot vector: a $\mathbb{Z}_{p^r}^N$ vector formed by L coefficient vectors of the $\mathrm{GR}(p^r; d)$ value in each slot
2. The polynomial coefficients vector: a $\mathbb{Z}_{p^r}^N$ vector of $m \in R_{p^r}$ under basis $\{X^i\}$

- CoeffToSlot: move (2) into (1). SlotToCoeff: move (1) into (2)

Decoding/Encoding as a chain of ring isomorphisms

- A homomorphic $\mathrm{Decode}(m) = \left(m(\eta^{s_0}), m(\eta^{s_1}), \dots, m(\eta^{s_{L-1}})\right): \mathbb{Z}_{p^r}^N \to \mathbb{Z}_{p^r}^N$ in slots achieves SlotToCoeff
- $\mathrm{Decode} = \mathrm{Eval} \circ \mathrm{Red}$, with

$$\mathrm{Red}(m) = (m \bmod F_0, m \bmod F_1, \dots, m \bmod F_{L-1}): \qquad R_{p^r} \to \prod_{i=0}^{L-1} \mathbb{Z}_{p^r}[X]/F_i(X)$$

$$\mathrm{Eval}(m_0, m_1, \dots, m_{L-1}) = \left(m(\eta^{s_0}), m(\eta^{s_1}), \dots, m(\eta^{s_{L-1}})\right): \prod_{i=0}^{L-1} \mathbb{Z}_{p^r}[X]/F_i(X) \to E^L$$

- $\mathrm{Red}(\cdot)$ can be computed with NTT (and a bit-reversal permutation Perm)
  - Iterative CRT: $X^8 + 1 = (X^4 - \eta^4)(X^4 + \eta^4) = \left((X^2 - \eta^2)(X^2 + \eta^2)\right)\left((X^2 + \eta^6)(X^2 - \eta^6)\right) = \cdots$
  - Digit removal (or decryption formula simplification) is insensitive to the order of slots, i.e.,

$$\mathrm{Decode}^{-1} \circ \mathrm{Perm}^{-1} \circ \mathrm{DigitRemoval} \circ \mathrm{Perm} \circ \mathrm{Decode} = \mathrm{Decode}^{-1} \circ \mathrm{DigitRemoval} \circ \mathrm{Decode}$$

- $\mathrm{Eval}(\cdot)$ is intra-slot → linearized polynomial

Plaintext encoding for power-of-two M

- If $p \equiv 1 \bmod 4$, $H = \langle -1,5 \rangle$, $d_1 = 2$, $d_2 = \frac{L}{2}$. Dim 1 is good, dim 2 is good iff $d = 1$
  - We flatten the $2 \times \frac{L}{2}$ sized array by concatenating the first and second row, i.e., $s_{\frac{L}{2}i+j} = (-1)^i 5^j$
  - $F_k(X) = X^d - \zeta^{s_k}$ with $\zeta \in \mathbb{Z}_{p^r}$ as a $2L$-th primitive root of unity
  - Cooley-Tukey NTT
- If $p \equiv 3 \bmod 4$, $H = \langle 5 \rangle$, $d_1 = L$. Dim 1 is good iff $d = 2$
  - Only a 1D array
  - $F_k(X) = X^d - (\zeta^{s_k} + \zeta^{s_k \cdot p})X^{d/2} + \zeta^{s_k(p+1)}$ with $\zeta \in \mathrm{GR}(p^r; 2)$ as a $4L$-th primitive root of unity
  - Bruun NTT

Inverse NTT decomposition of (the permuted) Red^{-1}

$$
\underset{N_1}{\begin{bmatrix}
* & & * & & & & & \\
 & * & & * & & & & \\
* & & * & & & & & \\
 & * & & * & & & & \\
 & & & & * & & * & \\
 & & & & & * & & * \\
 & & & & * & & * & \\
 & & & & & * & & *
\end{bmatrix}}
\begin{bmatrix}
*, F_0^{(0)} \\ *, F_1^{(0)} \\ *, F_2^{(0)} \\ *, F_3^{(0)} \\ *, F_4^{(0)} \\ *, F_5^{(0)} \\ *, F_6^{(0)} \\ *, F_7^{(0)}
\end{bmatrix}
=
\begin{bmatrix}
0*, F_0^{(1)} \\ 0*, F_1^{(1)} \\ 1*, F_0^{(1)} \\ 1*, F_1^{(1)} \\ 0*, F_4^{(1)} \\ 0*, F_5^{(1)} \\ 1*, F_4^{(1)} \\ 1*, F_5^{(1)}
\end{bmatrix},
\quad
\underset{N_2}{\begin{bmatrix}
* & * & & & & & & \\
* & * & & & & & & \\
 & & * & * & & & & \\
 & & * & * & & & & \\
 & & & & * & * & & \\
 & & & & * & * & & \\
 & & & & & & * & * \\
 & & & & & & * & *
\end{bmatrix}}
\begin{bmatrix}
0*, F_0^{(1)} \\ 0*, F_1^{(1)} \\ 1*, F_0^{(1)} \\ 1*, F_1^{(1)} \\ 0*, F_4^{(1)} \\ 0*, F_5^{(1)} \\ 1*, F_4^{(1)} \\ 1*, F_5^{(1)}
\end{bmatrix}
=
\begin{bmatrix}
00*, F_0^{(2)} \\ 10*, F_0^{(2)} \\ 01*, F_0^{(2)} \\ 11*, F_0^{(2)} \\ 00*, F_4^{(2)} \\ 10*, F_4^{(2)} \\ 01*, F_4^{(2)} \\ 11*, F_4^{(2)}
\end{bmatrix}
$$

$$
\underset{N_3}{\begin{bmatrix}
* & & & & * & & & \\
 & * & & & & * & & \\
 & & * & & & & * & \\
 & & & * & & & & * \\
* & & & & * & & & \\
 & * & & & & * & & \\
 & & * & & & & * & \\
 & & & * & & & & *
\end{bmatrix}}
\begin{bmatrix}
00*, F_0^{(2)} \\ 10*, F_0^{(2)} \\ 01*, F_0^{(2)} \\ 11*, F_0^{(2)} \\ 00*, F_4^{(2)} \\ 10*, F_4^{(2)} \\ 01*, F_4^{(2)} \\ 11*, F_4^{(2)}
\end{bmatrix}
=
\begin{bmatrix}
000*, F_0^{(3)} \\ 010*, F_0^{(3)} \\ 001*, F_0^{(3)} \\ 011*, F_0^{(3)} \\ 100*, F_0^{(3)} \\ 110*, F_0^{(3)} \\ 101*, F_0^{(3)} \\ 111*, F_0^{(3)}
\end{bmatrix}
$$

Fig. 2. An illustration of $\mathsf{Red}_{\mathrm{BR}}^{-1}$ for $D = 4$ and $p \equiv 1 \bmod 4$. A '$*$' in matrices stands for a nonzero entry that is a multiple of $\mathbf{I}_d$, while a '$*$' in the vectors means $\log_2(d)$ bits ranging from all zeros to all ones. Each slot stores part of the coefficients of $m \bmod F_i^{(j)}$. The (binary format of) indices of the coefficients are displayed along with the corresponding $F_i^{(j)}$. E.g., '$01*, F_0^{(2)}$' means that this slot stores $(m \bmod F_0^{(2)})[d +: d]$.

- Toy example of permuted Red^{-1} when $p \equiv 1 \bmod 4$ and $L = 8$
- Summary of Red^{-1}

1. $\log_2 L - 1$ 1D E-linear transformations, each with 2-3 nonzero diagonals
2. One 2D E-linear transformations with 2 nonzero diagonals

- More than E-linear: multiply by something in E → multiply by some **integer**

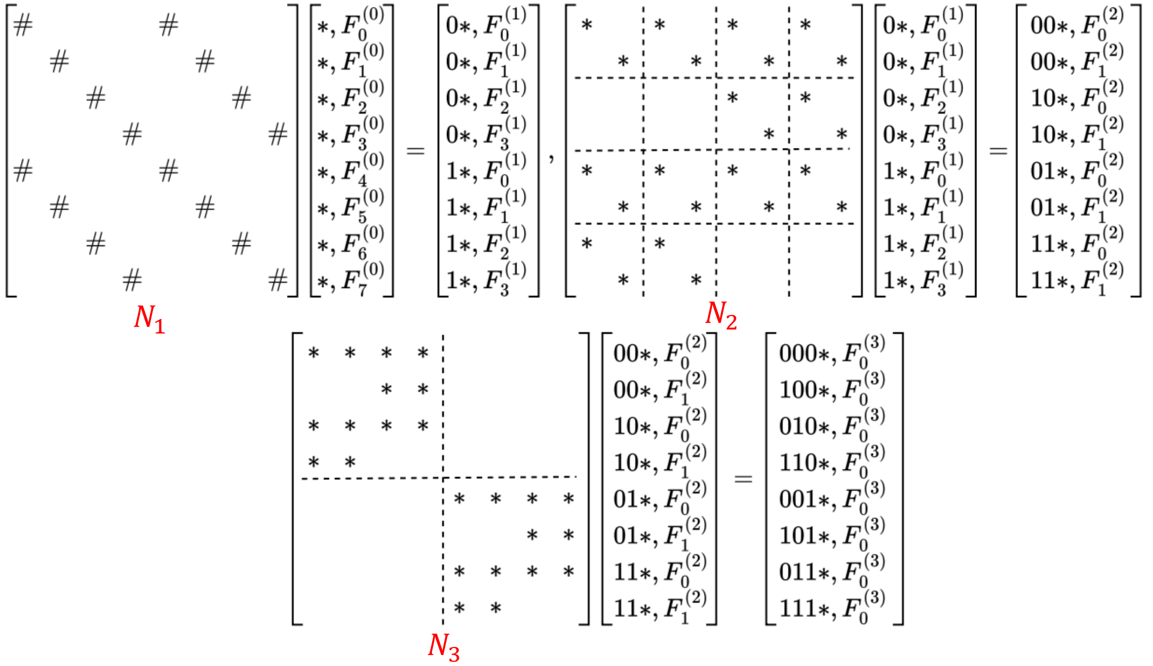


Fig. 3. An illustration of $\mathsf{Red}_{\mathrm{BR}}^{-1}$ in Bruun-style for $D = 8$ and $p \equiv 3 \bmod 4$. A '#' in matrices stands for a nonzero entry with the form of $\begin{bmatrix} a_0\mathbf{I}_{d/2} & a_1\mathbf{I}_{d/2} \\ a_2\mathbf{I}_{d/2} & a_3\mathbf{I}_{d/2} \end{bmatrix}$ for $a_i \in \mathbb{Z}_p$. Other symbols have the same meaning as in Figure 2.

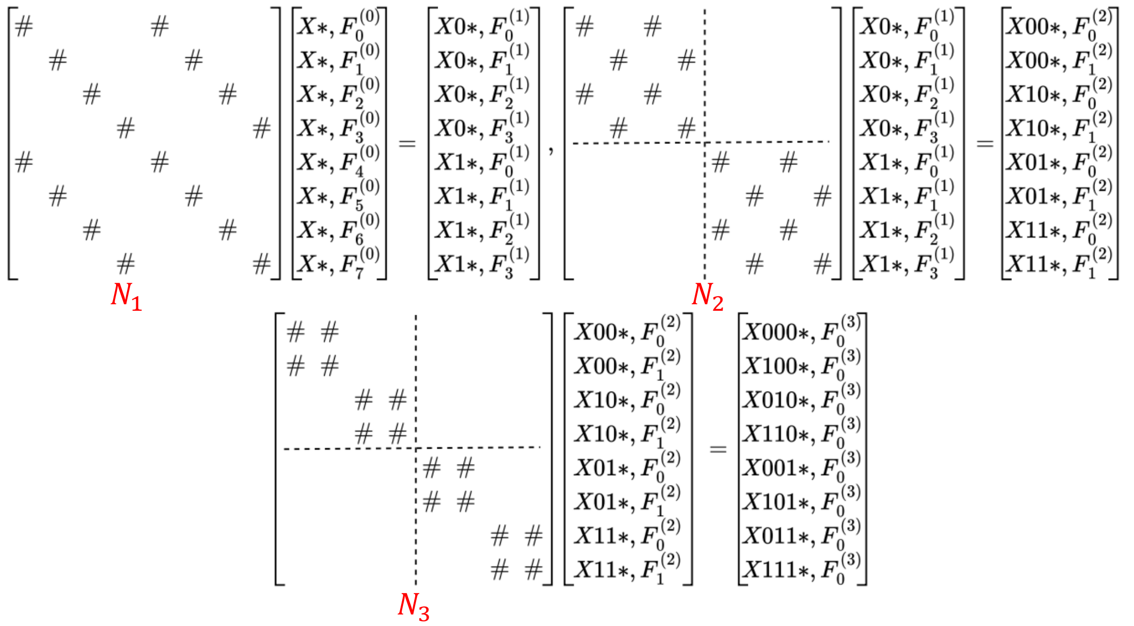


Fig. 4. An illustration of $\mathsf{Red}_{\mathrm{BR}}^{\prime-1}$ in Radix-2 style for $D = 8$ and $p \equiv 1 \bmod 4$. A '$*$' in vectors means $\log_2(d) - 1$ bits ranging from all zeros to all ones while a 'X' means a single bit running from 0 to 1. For example, when $d = 8$, '$X0*$' stands for $(0000, 0001, 0010, 0011, 1000, 1001, 1010, 1011)$. Other symbols have the same meaning as in Figure 2 and Figure 3.

Toy examples of permuted Red^{-1} when $p \equiv 3 \bmod 4$ and $L = 8$, with different butterfly arrangement in Bruun NTT

- Bruun style
  - One 1D $\mathbb{Z}_{p^r}$-linear transformation with 2 diagonals
  - $\log_2 D - 1$ 1D E-linear transformations with ≤7 diagonals
- Radix-2 style
  - $\log_2 D$ 1D $\mathbb{Z}_{p^r}$-linear transformations with 2-3 diagonals

Formulas for CoeffToSlot/SlotToCoeff

- $p \equiv 1 \bmod 4$
- General bootstrapping (**SlotToCoeff first**)
  - $\mathrm{PtoN} \circ \mathrm{Red}_{\mathrm{BR}}^{-1} \circ \mathrm{Eval}^{-1} \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}} = \mathrm{Red}_{\mathrm{BR}}^{-1} \circ (\mathrm{PtoN} \circ \mathrm{Eval}^{-1}) \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}}$
- Thin bootstrapping (SlotToCoeff first, only integers in slots)
  - $\mathrm{Red}_{\mathrm{BR}}^{-1} \circ \mathrm{Eval}^{-1} \circ \mathrm{Rm} \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}}$, where Rm removes extra coefficients in plaintext polynomial
- $p \equiv 3 \bmod 4$
- General bootstrapping (**SlotToCoeff first**)
  - $\mathrm{Red}_{\mathrm{BR}}^{-1} \circ (\mathrm{PtoN} \circ \mathrm{Eval}^{-1}) \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}}$
- Thin bootstrapping (SlotToCoeff first, only integers in slots)
  - Bruun style: $\mathrm{Red}_{\mathrm{BR}}^{-1} \circ \mathrm{Eval}^{-1} \circ \mathrm{Rm} \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}}$
  - Radix-2 style: $\mathrm{Rm}' \circ \mathrm{Red}_{\mathrm{BR}}^{-1} \circ \mathrm{Eval}^{-1} \circ \mathrm{Rm} \circ \cdots \circ \mathrm{Eval} \circ \mathrm{Red}_{\mathrm{BR}}$, where Rm' removes extra coefficients in slots

$$\mathrm{Red}_{\mathrm{BR}}^{-1} = \begin{cases} N_{\log_2 L} \circ \cdots \circ N_2 \circ N_1, & \text{Bruun style} \\ N_{\log_2 L} \circ \cdots \circ N_1, & \text{Radix2 style} \end{cases}$$

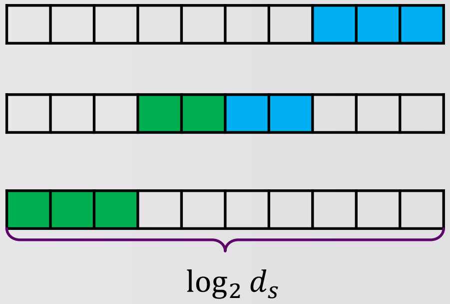
Combining consecutive NTT matrices

- Combine consecutive NTT matrices (and Eval or PtoN) to save some levels
  - Level collapsing from CKKS bootstrapping
  - More nonzero diagonals after combination: tradeoff between running time and remaining capacity
- $p \equiv 1 \bmod 4$
- General & thin bootstrapping: ${\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$
  - The product of k NTT matrices (or their inverses) has $< 2^{k+1}$ nonzero diagonals
  - ${\color{green}\blacksquare}$ in both ends are 2-dimensional
- $p \equiv 3 \bmod 4$
- General & thin bootstrapping:
  - ${\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$ for Bruun style, $< 7 \cdot 2^k$ nonzero diagonals
  - ${\color{red}\blacksquare} \circ \cdots \circ {\color{red}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{red}\blacksquare} \circ \cdots \circ {\color{red}\blacksquare}$ for Radix-2 style, $< 2^{k+1}$ nonzero diagonals

Optimized BSGS matrix multiplication

- BSGS matrix multiplication: reduce computation cost from $O(d_s)$ to $O\left(\sqrt{d_s}\right)$
  - Giant step g, number of giant steps $h = \left\lceil \frac{d_s}{g} \right\rceil$. Let $i = j + gk$ for $0 \le i < d_s$, where $0 \le j < g$. $g = O\left(\sqrt{d_s}\right)$ is optimal
  - Rotation keys for ρ_s^j and ρ_s^{gk} are included in the public key
  - E-linear case: $f(x) = \sum_{i=0}^{d_i-1} a_i \cdot \rho_s^i(x)$ → $f(x) = \sum_{k=0}^{h-1} \rho_s^{gk}\left(\sum_{j=0}^{g-1} \rho_s^{-gk}(a_i)\rho_s^j(x)\right)$
  - $\mathbb{Z}_{p^r}$-linear case: $f(x) = \sum_{i=0}^{d_i-1} \sum_{j=0}^{d-1} a_{i,j} \cdot \sigma^j\left(\rho_s^i(x)\right)$ is similar
- Hoisting: computing multiple automorphisms on the same input is faster
  - Switching the order of $\sum_j$ and $\sum_k$ to minimize the number of unhoisted automorphisms
- Reduce the number of small-step automorphisms
  - Diagonals of $N_k \cdots N_j$ roughly have indices $2^{-k} d_s \cdot \left[-c \cdot 2^{1+k-j}, c \cdot 2^{1+k-j}\right]$, with $c = 1$ or 3
  - Use a power-of-two g close to $\sqrt{d_s}$ → the range of j in $\sum_j$ is small

Binary representation of $i = j + gk$ for nonzero a_i



Faster $\mathbb{Z}_{p^r}$-linear transformation in thin bootstrapping

- During SlotToCoeff/CoeffToSlot in thin bootstrapping, the slot values lie in a subring $F < E$
  - Linearized polynomial needs $[F : \mathbb{Z}_{p^r}] - 1$ Frobenius automorphisms
- $p \equiv 1 \bmod 4$
  - $F = \mathbb{Z}_{p^r}, \mathrm{Eval}/\mathrm{Eval}^{-1}$ is omitted
  - $\blacksquare \circ \cdots \circ \blacksquare \circ \mathrm{Rm} \circ \mathrm{Nonlinear} \circ \blacksquare \circ \cdots \circ \blacksquare$
- $p \equiv 3 \bmod 4$
  - $[F : \mathbb{Z}_{p^r}] = 2$
  - $\blacksquare \circ \cdots \circ \blacksquare \circ \mathrm{Rm} \circ \mathrm{Nonlinear} \circ \blacksquare \circ \blacksquare \circ \cdots \circ \blacksquare$ for Bruun style
  - $\blacksquare \circ \cdots \circ \blacksquare \circ \mathrm{Rm} \circ \mathrm{Nonlinear} \circ \blacksquare \circ \blacksquare \circ \cdots \circ \blacksquare$ for Radix-2 style

Experiment Results

Table 2. The parameter sets. h and λ are the Hamming weight and the security level of the main secret key, while h' and λ' are those for the encapsulated bootstrapping key.

ID	p	r	M	L	D	d	$\log_2(Q)$	h	λ	h'	λ'
I	65537		65536	32768	16384	1				26	134.4
II	8191	1	65536	4096	4096	8	1332	120	81.13	24	129.8
III	131071		65536	16384	16384	2				26	133.81

Table 3. The partitions for general and thin bootstrapping.

	Bootstrapping Type	I	Style	II	III
Partition	Thin	(1,6,12,16)	Bruun	(1,6,10,13)	(1,7,12,15)
			Radix-2	(1,5,9,13)	(1,6,10,15)
	General	(1,6,12,16)	Bruun	(1,5,10,13)	(1,7,12,15)
			Radix-2	(1,5,9,13)	(1,6,10,15)

Table 4. Benchmark results for thin bootstrapping. Capacity refers to the capacity consumed by each stage of bootstrapping. The speedup is computed as the ratio of throughput with respect to the baseline case.

Parameter Set		I		II			III		
Method		Baseline	Ours	Baseline	Ours Bruun	Ours Radix2	Baseline	Ours Bruun	Ours Radix2
Capacity (bits)	Initial	941	941	947	947	947	939	939	939
	CoeffToSlot	62	134	56	119	118	64	144	143
	SlotToCoeff	39	79	33	70	69	39	85	85
	Digit extract	265	265	232	231	232	277	276	277
	Remaining	556	446	610	511	513	540	415	415
Time (sec)	CoeffToSlot	320	12.8	53	15.1	11.8	170	16.4	14.0
	SlotToCoeff	58	4.4	11.2	3.9	3.2	33	5.0	3.9
	Digit extract	6.0	5.9	5.9	6.1	6.0	6.1	5.7	5.7
	Total	385	23.4	71	26	21.6	209	27	24.1
Throughput (bps)		1.45	19.0	8.6	20.0	23.8	2.6	15.1	17.2
Speedup		1x	13.2x	1x	2.32x	2.8x	1x	5.9x	6.7x
Memory Usage (GB)		398	31	52	9.7	8.8	201	24.1	23.6

Table 5. Benchmark results for general bootstrapping. Capacity refers to the capacity consumed by each stage of bootstrapping. The speedup is computed as the ratio of throughput with respect to the baseline case.

Parameter Set		I		II			III		
Method		Baseline	Ours	Baseline	Ours Bruun	Ours Radix2	Baseline	Ours Bruun	Ours Radix2
Capacity (bits)	Initial	941	941	947	947	947	939	939	939
	CoeffToSlot†	62	134	58	120	129	65	143	143
	SlotToCoeff†	38	78	36	72	80	39	84	84
	Digit extract	265	264	297	293	295	326	327	326
	Remaining	557	447	541	447	428	489	366	366
Time (sec)	CoeffToSlot	316	12.7	1017	19.0	23.4	1624	16.1	14.1
	SlotToCoeff	316	12.8	1015	18.8	20.9	1625	15.9	14.0
	Digit extract	6.1	5.8	49	48	48	12.2	11.7	11.9
	Total	639	32	2082	86	93	3261	44	40
Throughput (bps)		0.87	14.1	0.26	5.2	4.6	0.15	8.3	9.1
Speedup		1x	16.2x	1x	20.0x	17.8x	1x	55x	60x
Memory Usage (GB)		398	31	744	11.8	13.6	392	24.1	23.6

Comparison with the concurrent work by Geelen [CIC'24]

- THEIRS
- N_i are 1D E-linear transformations with 3 nonzero diagonals
- $p \equiv 1 \bmod 4$
  - General bootstrapping ${\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare}$, ours is better
  - Thin bootstrapping ${\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\mathrm{Trace}} \circ \mathrm{Nonlinear} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$, both methods are the same
- $p \equiv 3 \bmod 4$
- General bootstrapping:
  - ${\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare}$
  - Better than our Radix-2 one
  - Compared to our Bruun one: fewer nonzero diagonals but two more '${\color{red}\blacksquare}$'
- Thin bootstrapping:
  - ${\color{olive}\mathrm{Trace}'} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\mathrm{Trace}} \circ \mathrm{Nonlinear} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$, theirs is better

- OURS
- $p \equiv 1 \bmod 4$, General & thin bootstrapping:
  - ${\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$
  - The product of k NTT matrices (or their inverses) has $< 2^{k+1}$ nonzero diagonals
  - ${\color{green}\blacksquare}$ in both ends are 2-dimensional
- $p \equiv 3 \bmod 4$, General & thin bootstrapping:
  - ${\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{green}\blacksquare} \circ \cdots \circ {\color{green}\blacksquare}$ for Bruun style, $< 7 \cdot 2^k$ nonzero diagonals
  - ${\color{red}\blacksquare} \circ \cdots \circ {\color{red}\blacksquare} \circ {\color{red}\blacksquare} \circ \mathrm{Nonlinear} \circ {\color{red}\blacksquare} \circ {\color{red}\blacksquare} \circ \cdots \circ {\color{red}\blacksquare}$ for Radix-2 style, $< 2^{k+1}$ nonzero diagonals

Thank you for listening

- Q&A