Accelerating HE Operations Using Key Decomposition

Yongsoo Song (Seoul National University)

Mar 26, 2023

The 2nd FHE.org Conference

Joint work with Miran Kim (Hanyang Univ), Dongwon Lee, Jinyeong Seo (SNU)

ia.cr/2023/413

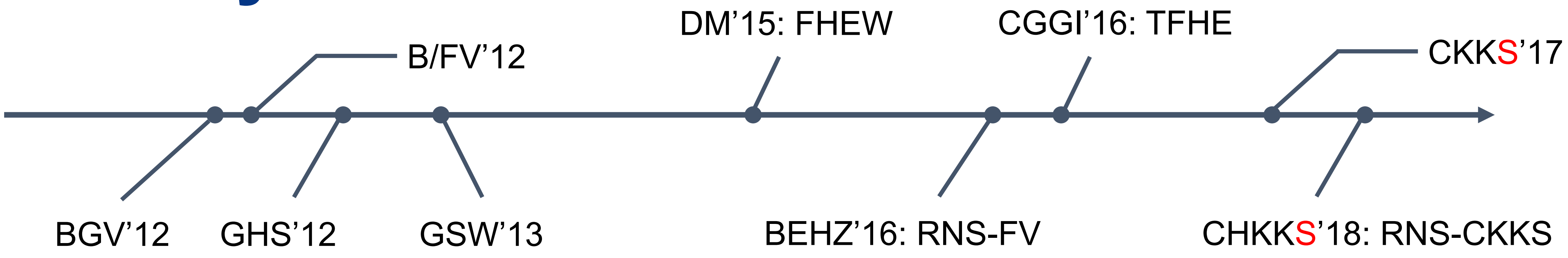
Table of Contents

01 Background

02 External Product & Key-switching

03 Our Contribution & Implementation Results

History



- Remarkable improvements of HE in 2012~2017

- Recent trends
  - Functionality (e.g., bootstrapping, non-arithmetic functions)
  - HE compilers
  - Hardware accelerator
  - Applications
  - Less interests in fundamental HE algorithms

Our Results

- There’s still room for algorithmic improvements!

- Accelerating **External Product**
  - A key building block of nonlinear HE operations
  - Key-switching (multiplication, automorphism) of BGV / BFV / CKKS
  - GSW operations

- Advantages & Implications
  - Better performance in terms of both asymptotic and concrete complexity
  - Compatible with the prior method and its implementation
  - May change future directions of hardware accelerator

Notation

Notation	Definition
N	Ring dimension (a power of two)
$R = \mathbb{Z}[X]/(X^N + 1)$	Ring of integers
$R_Q = \mathbb{Z}[X]/(X^N + 1)$	Residue ring
$Q = q_1 q_2 \ldots q_\ell$	Ciphertext modulus

RNS and NTT

- Residue Number System (RNS)
  - $Q = q_1 q_2 \dots q_d$, a product of distinct primes
  - Isomorphism: $R_Q \to \prod R_{q_i}$, $a \mapsto (a \bmod q_i)_{1 \le i \le \ell}$

- Number Theoretic Transform
  - ρ is a $(2N)$-th primitive root of unity ($q = 1 \pmod{2N}$).
  - NTT is a ring isomorphism $R_q \to \mathbb{Z}_q^N$ defined by $a(X) \mapsto \left(a(\rho), a(\rho^3), \dots, a(\rho^{2N-1})\right)$.

- Polynomial representations of $a(X) = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1} \in R_q$:
  - Coefficient form: $(a_0, a_1, \dots, a_{N-1})$.
  - NTT form: $\left(a(\rho), a(\rho^3), \dots, a(\rho^{2N-1})\right)$.

Gadget Decomposition

- Gadget vector: $\boldsymbol{g} = (g_1, \dots, g_k) \in R_Q^k$.
  - Denote the inner product function as $g: R^k \to R_Q$, $g(x_1, \dots, x_k) = \sum_i x_i \cdot g_i \pmod{Q}$.
  - The product is well-defined since R_Q is an R-module.

- Gadget decomposition: $h: R_Q \to R^k$
  - $h(a) = (b_1, \dots, b_k)$ is a "short" vector in $g^{-1}(a)$
  - In other words, b_i are small and $\sum_i b_i \cdot g_i = a$ over R_Q.

An Example

- RNS Basis $Q = q_1 q_2 \ldots q_\ell$ where $q_1, \ldots, q_\ell$ are distinct primes

- **Prime decomposition:** $h(a) = \left([a]_{q_i}\right)_{1 \le i \le \ell} \in R^\ell$

- Gadget vector: $\boldsymbol{g} = (g_1, \ldots, g_\ell) \in R_Q^\ell$
  - $g_i \equiv 1 \pmod{q_i}$ and $g_i \equiv 0 \pmod{q_j}$ for $j \neq i$.

- Correctness: $\sum_i [a]_{q_i} \cdot g_i = a \ \ (mod\ Q)$
  - Chinese Remainder Theorem

Table of Contents

01 Background

02 External Product & Key-switching

03 Our Contribution & Implementation Results

Definition

- External Product $\boxdot: R_Q \times R_Q^{\ell} \rightarrow R_Q$
  - Input: $a \in R_Q$ & $\boldsymbol{u} = (u_1, \ldots, u_\ell) \in R_Q^d$
  - Output: $c \in R_Q$
  - Step 1 (decomposition): $(b_1, \ldots, b_\ell) \leftarrow h(a) \in R^{\ell}$
  - Step 2 (linear combination): $c = b_1 u_1 + \cdots + b_\ell u_\ell \in R_Q$

- Note : $a \boxdot \boldsymbol{g} = a$

- We write $a \boxdot \boldsymbol{U} = (a \boxdot \boldsymbol{u}_0, a \boxdot \boldsymbol{u}_1) \in R_Q^2$ for $\boldsymbol{U} = [\boldsymbol{u}_0 \mid \boldsymbol{u}_1] \in R_Q^{\ell \times 2}$

Key-switching

- Key-switching key: $\boldsymbol{U} = [\boldsymbol{u}_0 \mid \boldsymbol{u}_1] = [s \cdot \boldsymbol{u}_1 + \boldsymbol{e} \mid \boldsymbol{u}_1] + [s' \cdot \boldsymbol{g} \mid 0]$

- Input: a ciphertext component a (under a secret s')

- Output: $(c_0, c_1) \leftarrow a \boxdot \boldsymbol{U} \in R_Q^2$ is an RLWE ciphertext:

$$c_0 + c_1 s \approx a s' \pmod{Q}.$$

- Examples:
  - Relinearization: $s' = s^2$.
  - Automorphism: $s' = \varphi(s)$.

Implementing External Product

- Input: $a \in R_Q$ & $\boldsymbol{u} = (u_1, \dots, u_\ell) \in R_Q^d$
  - Step 1 (decomposition): $(b_1, \dots, b_\ell) \leftarrow h(a) \in R^\ell$
  - Step 2 (linear combination): $c = b_1 u_1 + \cdots + b_\ell u_\ell \in R_Q$

- **Key question:** How to compute the product of $b_i \in R$ and $u_i \in R_Q$?

- **Previously:**
  - Compute the RNS representation of $b_i \in R$ modulo Q : ℓ^2 NTTs
  - Then perform the inner product over R_Q: ℓ^2 mults

Table of Contents

01 Background

02 External Product & Key-switching

03 Our Contribution & Implementation Results

Our Method

- **Goal:** Compute the linear combination $c = b_1 u_1 + \cdots + b_\ell u_\ell \in R_Q$ more efficiently.

- **Main Idea:** perform the computation over R as much as possible
  - Suppose that the gadget decompositions of u_i are given, say $h(u_i) = \left(v_{i,1}, \dots, v_{i,\ell}\right) \in R^\ell$.
  - From $u_i = \sum_j v_{i,j} \cdot g_j \pmod{Q}$,

$$c = \sum_i b_i u_i = \sum_i b_i \cdot \left(\sum_j v_{i,j} \cdot g_j\right) = \sum_j \left(\sum_i b_i v_{i,j}\right) \cdot g_j \pmod{Q}.$$

Important Facts

$$c = \sum_j \left(\sum_i b_i v_{i,j} \right) \cdot g_j \pmod{Q}.$$

- $v_{i,j}$ can be precomputed
- $c_j = \sum_i b_i v_{i,j}$ are **small** elements of R.
  - Its upper bound is determined by h, not Q.
- $c = c_j \ (mod\ q_j)$, so $(c_1, \ldots, c_\ell)$ is the RNS representation of c modulo Q.

New External Product

- Input: $a \in R_Q$ and $h(u_i) = (v_{i,1}, \dots, v_{i,\ell}) \in R^\ell$ for $1 \le i \le \ell$.
  - $v_{i,j}$ are precomputed and given in the DFT form over R

- Step 1: Compute $b_1 = [a]_{q_1}, \dots, b_\ell = [a]_{q_\ell}$ over R: ℓ DFTs
- Step 2: Compute $c_j = \sum_i b_i v_{i,j}$ for each j: ℓ^2 mults in total
- Step 3: Convert c_j back into the coefficient form: ℓ inverse DFTs

- Output: $(c_1, c_2, \dots, c_\ell)$, the RNS rep. of c modulo Q.

This is purely an algorithmic optimization, computing the same ciphertext!

In the full version..

- The **special-modulus** technique is applied
  - A key-switching key has a larger modulus $PQ > Q$.
- General RNS-based gadget decompositions
  - $h(a) = \left([a]_{D_1}, \ldots, [a]_{D_k}\right)$ for some $D_1 \ldots D_k = Q$
  - Another gadget decomposition over R_{PQ} for u_i.
- Need multi-precision integral polynomial arithmetic
  - Precision depends on the decomposition bounds

Implementation

- Used the Lattigo library

- Base Ring of Integers
  - Instantiate R by R_B for sufficiently large $B > 0.$
  - B is a product of r' prime numbers p_i
  - A DFT over R corresponds to r' NTTs

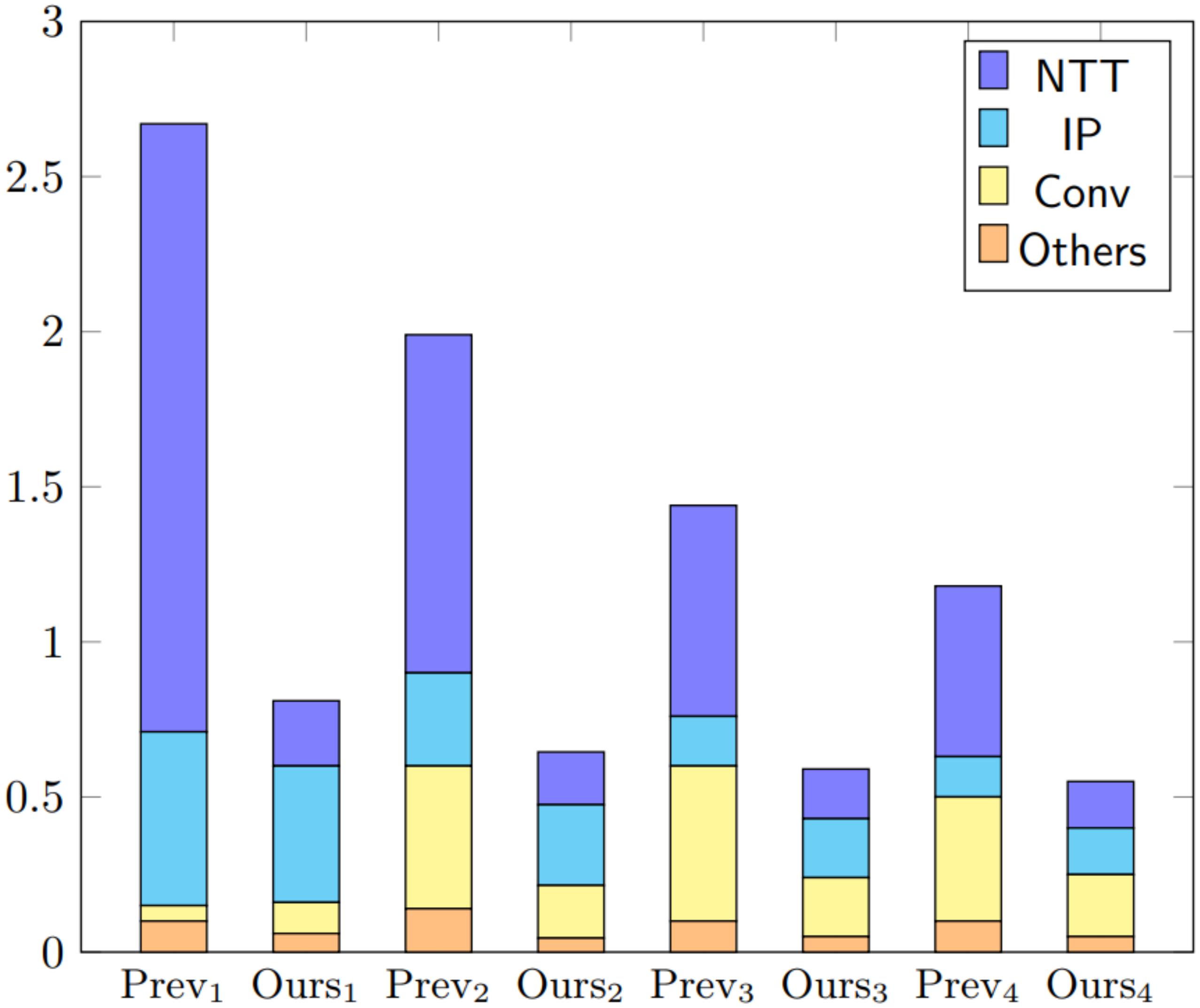
- Parameter setup
  - The usual HE parameters
  - Choose the best-performing decomposition over R_{PQ}

Experimental Results (CPU)

N	ℓ	r	Prev	Ours	Speedup
32768	24	1	0.317	0.140	2.3
		2	0.204	0.110	1.9
		3	0.151	0.101	1.5
65536	48	1	2.688	0.818	3.3
		2	1.980	0.655	3.0
		3	1.438	0.585	2.5
		4	1.191	0.550	2.2

N: RLWE dimension, r: decomposition size (no. of primes in each digit)

Experimental Results (CPU)



$r = 1$ $r = 2$ $r = 3$ $r = 4$