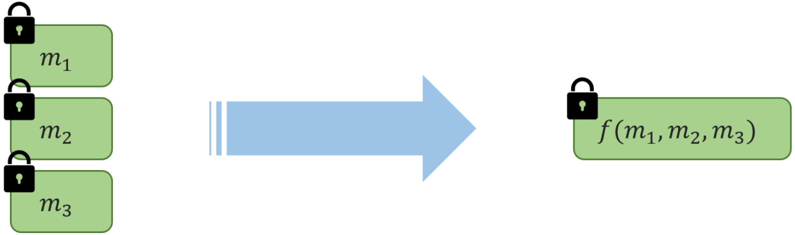
Towards Practical MK-TFHE:

Parallelizable, Quasi-linear and Key-compatible

Hyesun Kwak, **Seonhong Min**, Yongsoo Song

Seoul National University, Seoul

Fully Homomorphic Encryption



- **Fully Homomorphic Encryption (HE)** supports arbitrary function evaluation on encrypted data.
- **Various Applications**: privacy preserving machine learning, private information retrieval, private set intersection ...

FHE for Multiple Parties

	MKHE	(n-out-of-n) **Threshold HE**
Key structure	$\bar{\mathbf{s}} := (s_1 \vert s_2 \vert \ldots \vert s_k)$	$\bar{\mathbf{s}} := \sum_{i=1}^{k} s_i$
Dynamic	Dynamic	Static
Communication	Independent	Interactive
Time/Space Complexity	Dependent to k	Comparable to single-key

Table: Comparison between Multi-Party HE schemes.

Previous Works

- Theoretical studies
  - LATV12, CM15, MW16, PS16, BP16, CZW17
  - (Mostly) GSW scheme
  - No implementations
- Practical schemes
  - CCS19[1] : TFHE/FHEW, quadratic complexity
  - CDKS19[2] : CKKS/BFV, quadratic complexity
- Better time complexity
  - KKLSS22[3] : CKKS/BFV, quasi-linear complexity
  - **This work** : TFHE/FHEW, quasi-linear complexity

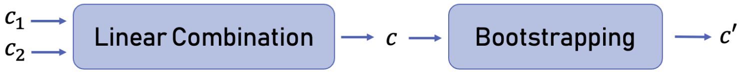
[1] Chen, Chillotti and Song, Asiacrypt '19
[2] Chen, Dai, Kim and Song, CCS '19
[3] Kim, Kwak, Lee, Seo and Song, CCS '23

TFHE/FHEW scheme description

- FHE scheme that supports bits operations (NAND, AND, OR...).
- **Secret Key**:
  - LWE secret $\mathbf{s} = (s_1, \ldots, s_n)$
  - RLWE secret $t \in R = \mathbb{Z}[X]/(X^N + 1)$
- **Encoding**: $m \in \{-1, 1\} \mapsto \mu = \frac{q}{8} m \in \mathbb{Z}_q$
- **Decoding**: $\begin{cases} 1 & \text{if } \mu > 0 \\ -1 & \text{otherwise} \end{cases}$
- **Encryption**: $c = (b, \mathbf{a}) \in \mathbb{Z}_q^{n+1}$ for $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $e \leftarrow$ small dist., $b = -\langle \mathbf{a}, \mathbf{s} \rangle + \mu + e \pmod{q}$.
- **Decryption**: $b + \langle \mathbf{a}, \mathbf{s} \rangle = \mu + e \pmod{q}$

Homomorphic Gate Evaluation (TFHE/FHEW)

- Each bit operation consists of the following pipeline:



- **Linear Combination** : The linear combination corresponding to a Boolean gate is evaluated.
  - ex) NAND : $c = (\frac{q}{8}, \mathbf{0}) - c_1 - c_2$
  - output ciphertext contains a **large noise** e.

- **Bootstrapping** : Reduces the size of noise for further evaluation.
  - ex) $\|e\| < \frac{q}{8} \to \|e'\| < \frac{q}{16}$
  - Consists of **Blind Rotation** and **Key Switching**

Blind Rotation

- **Input** : $\mathbf{c} = (b, \mathbf{a})$ such that $b + \langle \mathbf{a}, \mathbf{s} \rangle = \frac{q}{8}m + e \pmod{q}$.
- Let $\tilde{b} = \left\lfloor \frac{2N}{q} \cdot b \right\rceil$, $\tilde{\mathbf{a}} = \left\lfloor \frac{2N}{q} \cdot \mathbf{a} \right\rceil$.
  - $\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle = \frac{2N}{8}m + \tilde{e} \pmod{2N}$.
- Pre-assign the coefficients to a polynomial tv, so that the constant term of $tv \cdot X^{\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle} \in R_q = R/qR$ is $\frac{q}{8}m$.
  - Since $X^N + 1 = 0$, mod 2N is naturally supported over the exponent.
- We can bootstrap the input ciphertext by computing $tv \cdot X^{\tilde{b} + \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle}$, and extracting the constant term.
- Homomorphically multiply $[X^{a_i s_i}]_t$ to $tv \cdot X^b$ iteratively.
- This is the main bottleneck of TFHE/FHEW bootstrapping.

MKTFHE description

- **Setup:** Each i-th party samples...
  - LWE secret $\mathbf{s}_i = (s_{i,1}, \ldots, s_{i,n})$
  - RLWE secret $t_i \in R$
- MK secret is the concatenation of each party's secret.
  - LWE secret $\bar{\mathbf{s}} = (\mathbf{s}_1 | \ldots | \mathbf{s}_k)$
  - RLWE secret $\bar{t} = (t_1, \ldots, t_k)$
- **Ciphertext**: $c = (b | \mathbf{a}_1 | \ldots | \mathbf{a}_k) \in \mathbb{Z}_q^{kn+1}$
  - $b + \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{s}_i \rangle \approx \mu \pmod{q}$.
- **Decryption**: $b + \sum_{i=1}^{k} \langle \mathbf{a}_i, \mathbf{s}_i \rangle = \mu + e$

Blind Rotation (CCS19)

- Homomorphically multiply monomials $[X^{a_{i,j}s_{i,j}}]_{t_i}$ to $tv \cdot X^b$ iteratively.
- Major building block: **Hybrid product**
  - homomorphic multiplication between MK-RLWE ciphertext and single-key RGSW-style encryption.
  - $\tilde{O}(kn)$ time complexity
- kn hybrid products, therefore overall time complexity is $\tilde{O}(k^2n^2)$.
- The timing **scales quadratically** as # of parties grows.

Our Idea

Motivation : Perform blind rotation party-wisely in a single-key manner, to achieve linear complexity $\tilde{O}(kn^2)$.

Challenge : No known homomorphic multiplication algorithm between multi-key and 'noisy' single-key ciphertexts.

Our Result :

1. **Generalized External Product**
   - A new homomorphic multiplication operation between MK-RLWE and generic single-key RGSW-like ciphertexts
2. **Improved Hybrid Product**
   - We improve Hybrid product by reducing the number of gadget decompositions.
3. **Faster Blind Rotation**
   - The time complexity is reduced to $\tilde{O}(kn^2)$.
   - Parallelizable, Key-compatible.

Generalized External Product (Simplified)

- **Input:**
  - MK-RLWE encryption $\overline{\text{ct}} = (c_0, \ldots, c_k)$ such that $\sum_{j=0}^{k} c_j \cdot t_j \approx m \pmod{q}$.
  - RGSW-like (noisy) encryption $\mathbf{C}$ of μ under secret t_i
  - RGSW-like (fresh) encryption **rlk** of t_i under secret t_i
- **Idea:**
  - Multiply $\mathbf{C}$ to each index of $\overline{\text{ct}}$ to obtain MK-RLWE encryption $\overline{\text{ct}}' = (\mathbf{x}|\mathbf{y})$ of $m \cdot \mu$.
    - However, key is changed to $(1, t_i) \bigotimes (1, t_1, \ldots, t_k)$!
    - i.e., $\langle \mathbf{x}, (1, t_1, \ldots, t_k) \rangle + \langle \mathbf{y}, t_i \cdot (1, t_1, \ldots, t_k) \rangle \approx m \cdot \mu \pmod{q}$
  - Multiply **rlk** to $\mathbf{y}$ using hybrid product, and add to $\mathbf{x}$.
    - Key is changed back to $(1, t_1, \ldots, t_k)$.
- **Time complexity:** $\tilde{O}(kn)$

Faster Blind Rotation

- **Our Algorithm:**
  1. Compute $[X^{\langle \mathbf{a}_i, \mathbf{s}_i \rangle}]_t$ for each i-th party with RGSW-like ciphertext.
  2. Multiply them to $X^b \cdot tv$ iteratively, using the generalized external product.
- **Time Complexity:**
  - The first step requires $\tilde{O}(n^2)$ time complexity for each party.
  - The second step requires k generalized external products.
  - In total, the time complexity is $\tilde{O}(kn^2 + k^2 n)$.
  - In practice, $k \ll n$ and therefore **quasi-linear**.
- **Parallelizable:** The first step can be **algorithmically parallelizable**.
- **Key-Compatible:** The public key is identical to the single-key scheme, with an extra relinearization key.

Faster Blind Rotation

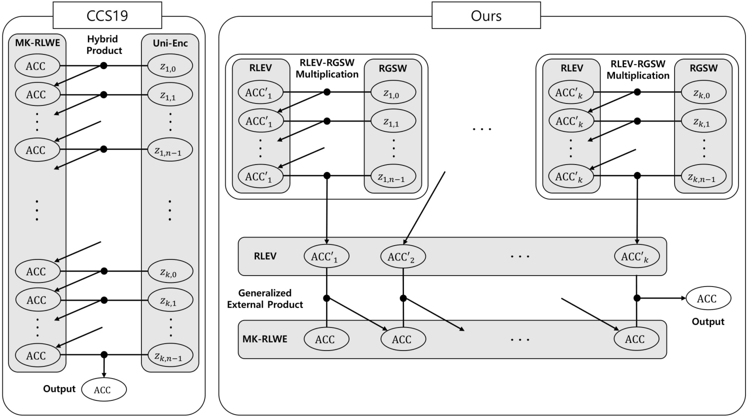


Figure: High-level overview of the blind rotation algorithm of MK variant of TFHE from CCS19 and Ours.

Timing Results

k	CCS19	Ours	Parallelized
2	0.24s	0.24s	0.17s
4	0.89s	0.88s	0.27s
8	3.32s	2.23s	0.35s
16	24.72s	5.65s	0.47s
32	-	13.94s	0.88s

Table: The elapsed time of our scheme and the CCS19 scheme.

- We achieve **4.38x** speedup without parallelization!
- **52.60x** speedup with parallelization!
- CCS19 doesn't support a practical parameter for $\geq$32 parties.

Timing Results

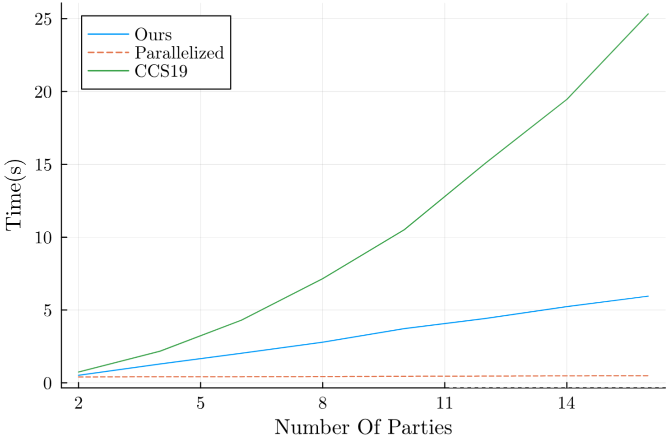
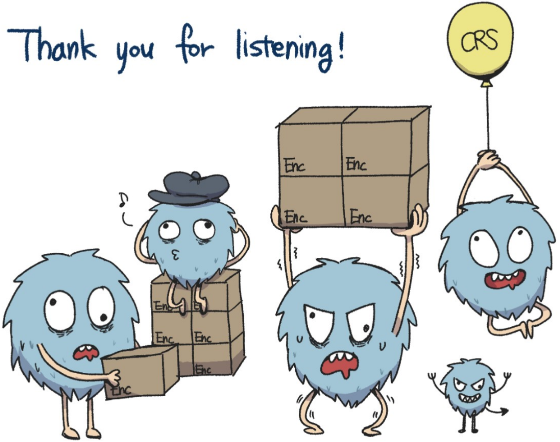


Figure: The elapsed time of our scheme and the CCS19 scheme.



- Julia : https://github.com/SNUCP/MKTFHE
- Go : https://github.com/sp301415/tfhe-go