TOWARDS LIGHTWEIGHT CKKS: ON CLIENT COST EFFICIENCY

Jung Hee Cheon[1,2], Minsik Kang[3], Jai Hyun Park[2]

SUMMARY

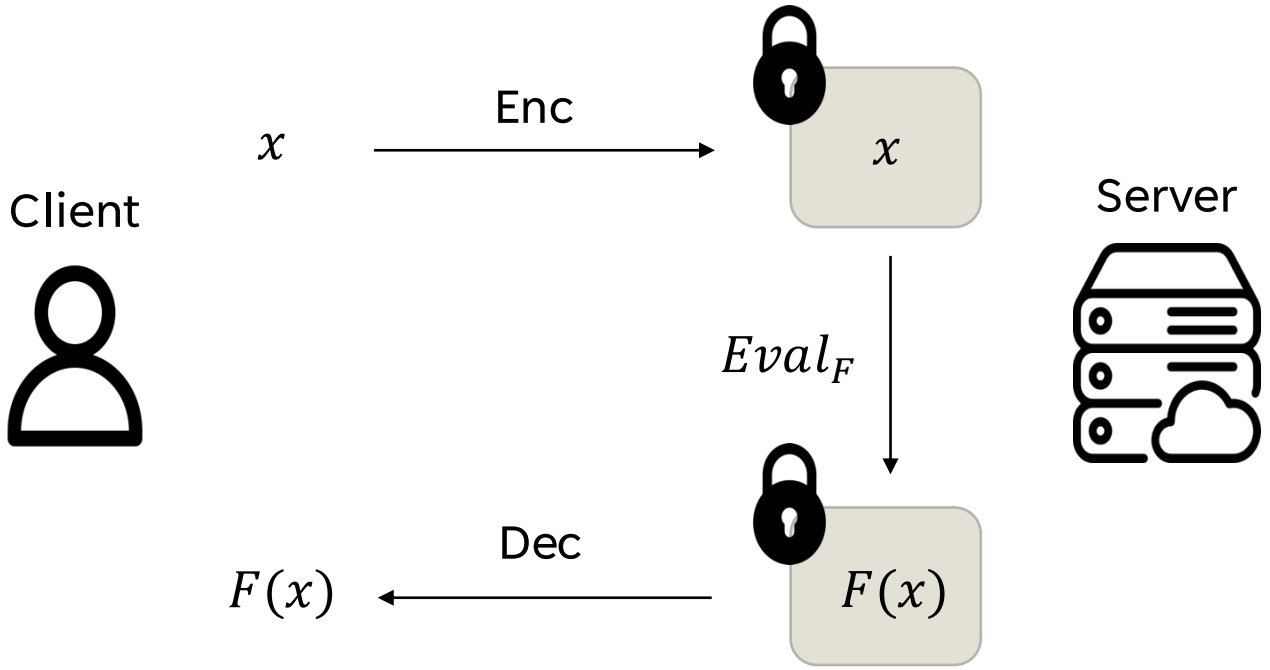
- New key management for FHE scheme.
  - Client transmits a few lightweight master keys.
  - Server derives all HE evaluation keys.

- Auxiliary parameter for key generation + ring-switching.
  - Fewer and smaller master keys than [LLKN23].

- To support the ResNet-20 inference [LLL+22],
  - KG+: From 3.49GB to 0.61GB (↓ 5.73x), without any compromise.
  - BTS+: Further reduce to 0.29GB (↓ 12.25x), no overhead for batching.

[LLKN23] J. Lee, et al., Rotation Key Reduction for Client-Server Systems of Deep Neural Network on Fully Homomorphic Encryption, ASIACRYPT 2023
[LLL+22] E. Lee, et al., Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions, ICML 2022

FULLY HOMOMORPHIC ENCRYPTION (FHE)



- FHE enables computations on encrypted data without decryption.
- It provides privacy-preserving computation with minimal communication.
- CKKS supports approximate computations on real/complex numbers.

CKKS FHE SCHEME

- For 2N-th primitive root ζ, define DFT : $\mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$:

$$DFT\big(a(X)\big) = (a(\zeta^{5^i}))_{0 \le i < N/2}$$

  - $5^{N/2} \equiv 1 \ \ (mod\ 2N)$, where N is a power-of-two.

- CKKS encodes a vector $\vec{z} \in \mathbb{C}^{N/2}$ into a plaintext:
  - $m = Ecd(\vec{z}) \coloneqq \lfloor \Delta \cdot DFT^{-1}(\vec{z}) \rceil \in R = \mathbb{Z}[X]/(X^N + 1)$

- CKKS ciphertext: a pair over $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$

$$ct_s(m) = (b, a) \in R_Q^2 \colon \quad b + a \cdot s = m + e \ \ (mod\ Q)$$

  - Under RLWE assumption, $Q < Q_{max}$ is bounded above.

HOMOMORPHIC ROTATION

- $\sigma_r : R \to R$ given by $X \mapsto X^{5^r}$ is ring automorphism on R.
  - $\sigma_r = \sigma_1 \circ \sigma_1 \circ \cdots \circ \sigma_1$, and $\sigma_{N/2} = id$
- $m(X) = Ecd(z_1, z_2, \dots, z_{N/2}) \xrightarrow{\sigma_r} m\left(X^{5^r}\right) = Ecd(z_{r+1}, \dots, z_{N/2}, z_1, \dots, z_r)$
- For $ct_s(m) = (b, a)$, we rotate by r-shift:

$$b + a \cdot s \approx m \ (mod\ Q)$$

$$\downarrow \text{Evaluate } \sigma = \sigma_r$$

$$\sigma(b) + \sigma(a) \cdot \sigma(s) \approx \sigma(m) \ (mod\ Q) \quad \Leftarrow ct_{\sigma(s)}\left(\sigma(m)\right) = \left(\sigma(b), \sigma(a)\right)$$

$$\downarrow \sigma(s) + \epsilon = \beta + \alpha s$$

$$b' + a' \cdot s \approx \sigma(m) + \sigma(a) \cdot \epsilon \ (mod\ Q) \quad \Leftarrow ct_s\left(\sigma(m)\right) = (b', a')$$

- Each rotation requires an evaluation key: $ct_s\left(\sigma(s)\right) = (\beta, \alpha)$

KEY-SWITCHING UNDER MODULUS BUDGET (1)

- Rotation key $ct_s\big(\sigma_r(s)\big) \in R_Q^2$ gives error $\sigma(a) \cdot \epsilon$ with $\sigma(a) \in R_Q$.
- To control the error, we need $rotk_r = ct_s\big(P \cdot \sigma_r(s)\big) \in R_{PQ}^2$, instead.

  $\Rightarrow$ Then, the error becomes $\sigma(a) \cdot \epsilon / P$.

- P is auxiliary modulus that satisfies

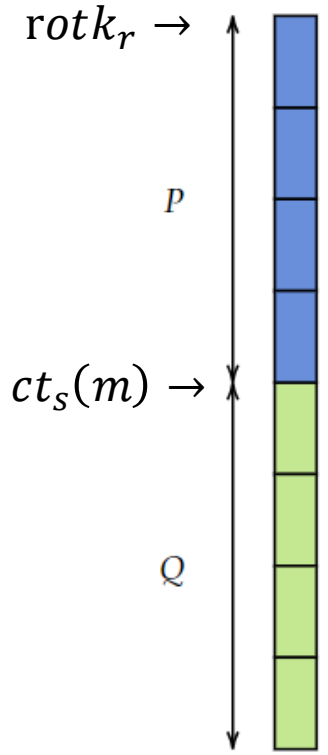
$$P \approx Q$$

- Under RLWE assumption,

$$P \cdot Q \leq Q_{max}$$

- Therefore, ciphertext modulus restricts to

$$\log Q \approx \tfrac{1}{2} \cdot \log Q_{max}$$



KEY-SWITCHING UNDER MODULUS BUDGET (2)

- To ensure larger modulus Q, we use CRT gadget decomposition [GHS12].
- We decompose Q into $d = dnum$ digits:

$$Q = Q_1 Q_2 \cdots Q_d$$

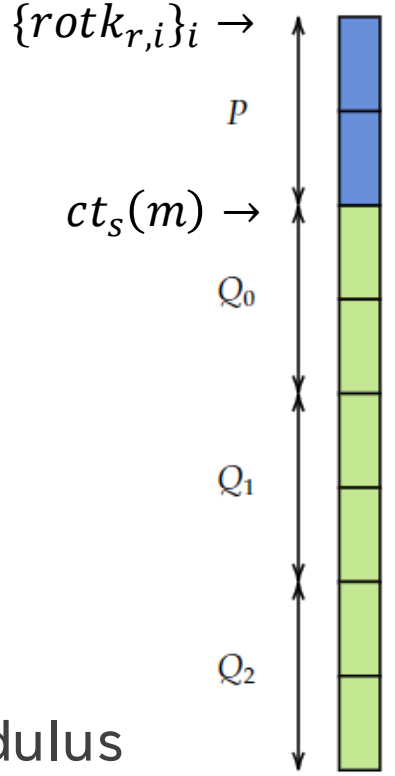
- A ciphertext $rotk_r$ decomposes into d ciphertexts $\{rotk_{r,i}\}_{i\in[d]}$
- The condition for P alleviates to

$$P \approx \max_i Q_i$$

- We can take the modulus Q as:

$$\log Q \approx \frac{d}{d+1} \cdot \log Q_{max} \quad \text{with} \quad Q_1 \approx \cdots \approx Q_d$$

⇒ dnum controls key size vs. available computation modulus



[GHS12] C. Gentry et. al., Homomorphic evaluation of the AES circuit, Crypto 2012.

LARGE EVALUATION KEY SIZE

- FHE parameters use the 2nd type for computation → large dnum.
- With only $rotk_1$, r-shift rotation requires r key-switchings,
  - The server needs $rotk_r$ for many r for efficiency.

- Each $rotk_r$ has the size:

$$dnum \cdot 2 \cdot N \cdot \log PQ$$

- Moreover, the client should generate and transmit them.

  CKKS bootstrapping: up to 3.04 GB

  ResNet-20 inference: up to 14.5 GB

- Can we reduce (1) # of transmission keys and (2) dnum?

PREVIOUS WORK AND LIMITATION

- Rotation key can be obtained on the server side [LLKN23]:

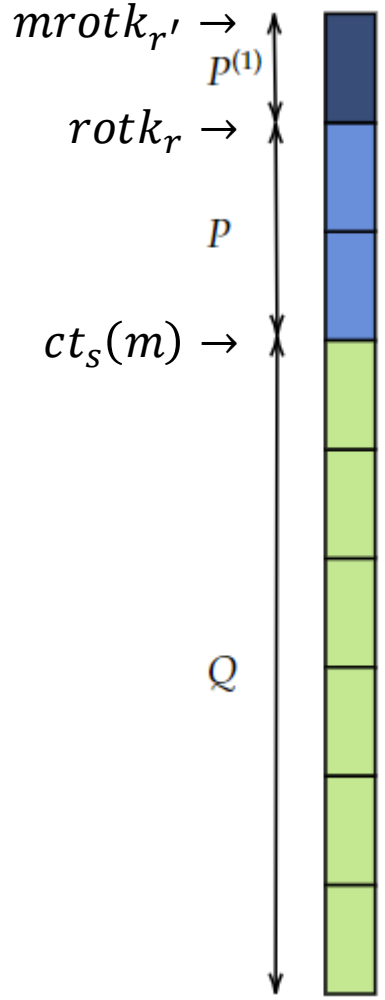
$$Rot_{r'}\,(rotk_r) = Rot_{r'}(ct_s(\sigma_r(s))) = ct_s(\sigma_{r+r'}(s)) = rotk_{r+r'}$$

  - [LLKN23] generates 265 rotation keys from 8 master keys.

- Master key for rotation keys need larger modulus $P^{(1)}PQ \leq Q_{max}$
  - Much larger dnum for modulus PQ since $PQ \lesssim Q_{max}$
  - Too small $P^{(1)} \rightarrow$ dnum increases from 4 to 30

- Transmission key size : 14.5 GB → 3.49 GB (× 4.14 smaller)

Is 3.49 GB satisfactory for the client?



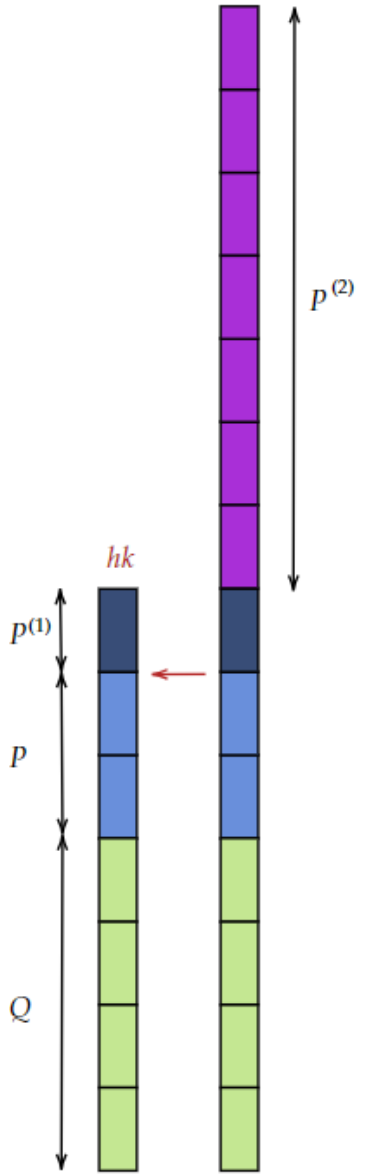
BRIDGE PARAMETERS VIA RING-SWITCHING

- We separate parameters for different roles.

- Key generation: 1st-type parameter over extension 2N-ring
  - $\log Q_{max,2N} \approx 2 \cdot \log Q_{max,N}$, so richer modulus for auxiliary P
  - Fewer and lightweight master keys

- Computation: 2nd-type parameter over original N-ring
  - Still efficient HE computations

- Bridge: ring-switching [GHPS13] for evaluation keys.



[GHPS13] C. Gentry, et al., Field switching in bgv-style homomorphic encryption. Journal of Computer Security, 21(5):663–684, 2013.

HOMING THE EVALUATION KEYS

- Embed $R = \mathbb{Z}[X]/(X^N + 1) \to R' = \mathbb{Z}[Y]/(Y^{2N} + 1)$ by $X \mapsto Y^2$
- (Goal) homing R'-key back to R-key.
- Decompose $s'(Y) = s(X) + Y s_1(X) \in R'$ with $s = s_0, s_1 \in R$,

$$\longrightarrow \sigma(s') = \sigma(s) + Y \cdot (Y^{-1}\sigma(Y)\sigma(s_1)) \text{ with } \sigma(s),\ Y^{-1}\sigma(Y)\sigma(s_1) \in R$$

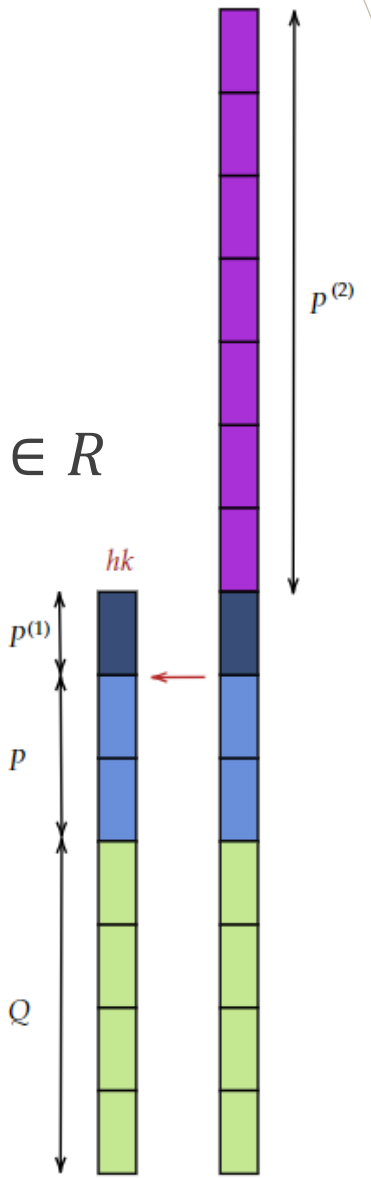
$$b' + a' \cdot s' \approx \sigma(s') \text{ over } R'_{PQ}$$

$$\downarrow \text{Extract even-degree part}$$

$$b_0 + a_0 s + (X a_1) s_1 \approx \sigma(s) \text{ over } R_{PQ}$$

$$\downarrow s_1 = \beta + \alpha s - \epsilon;\ hk = ct_s(s_1)$$

$$b + a \cdot s \approx \sigma(s) \text{ over } R_{PQ}$$



KG+: NEW KEY MANAGEMENT SYSTEM

Master rotation keys of shift $\{1, 256\}$ over $R'_{P^{(2)}P^{(1)}PQ}$

↓ Generate 8 keys

Intermediate keys of shift $\{1, 4, 4^2, \dots, 4^7\}$ over $R'_{P^{(1)}PQ}$

↓ Generate required shifts

Rotation keys of target shifts over R'_{PQ}

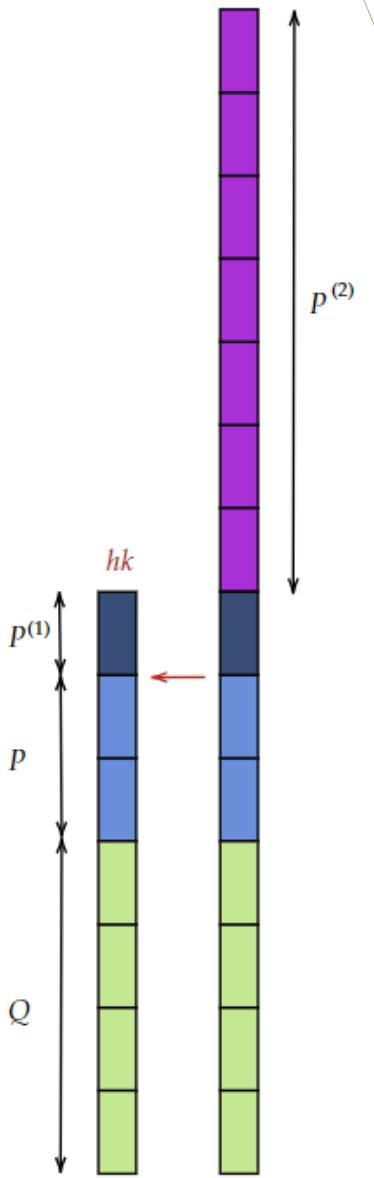
↓ homing

Target rotation keys over R_{PQ}

- Therefore, the client only needs to transmit:

  2 master rotation keys, homing key.

  - Master rotation keys have dnum = 1.



IMPLEMENTATION RESULTS

Method		dnum	Key Size (GB)	Client KeyGen (s)	Server KeyGen (s)
BTS-A	Original	4	2.18	12.3	-
	[LLKN23]	(4, 30)	2.65	10.6	8.06
	KG+	(4, 30, 1)	0.609 (↓ 4.35x)	0.99 (↓ 10.71x)	40.9 (↑ 5.07x)
BTS-B	Original	5	2.62	14.2	-
	[LLKN23]	(5, 15)	1.40	6.25	8.06
	KG+	(5, 15, 1)	0.397 (↓ 3.08x)	1.02 (↓ 6.13x)	32.5 (↑ 4.03x)
ResNet-A	Original	4	14.5	61.6	-
	[LLKN23]	(4, 30)	3.49	14.1	45.7
	KG+	(4, 30, 1)	0.609 (↓ 5.73x)	0.99 (↓ 14.24x)	107.0 (↑ 2.34x)
ResNet-B	Original	5	17.4	73.0	-
	[LLKN23]	(5, 15)	1.82	7.83	49.9
	KG+	(5, 15, 1)	0.397 (↓ 4.58x)	1.02 (↓ 7.68x)	96.3 (↑ 1.93x)

- Based on the FHE parameters with $N = 2^{16}$, $\log PQ = 1714$.

Param. Info. Intel Xeon gold 6542Y with a single-thread CPU & NVIDIA A100 80GB PCIe GPU, HEaaN library

THANK YOU!

eprint.iacr.org/2025/720