WAHC 2024 – 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography

Faster Homomorphic Evaluation of Arbitrary Bivariate Integer Functions via Homomorphic Linear Transformation

October 14th, 2024

○Akira Nakashima(NEC Corporation), Takuya Hayashi(NEC Corporation),
Hikaru Tsuchida(Saitama Institute of Technology), Yukimasa Sugizaki(NEC Corporation),
Kengo Mori(NEC), Takashi Nishide (University of Tsukuba)

Table of Contents

1. Introduction

2. Prior Work

3. Our Proposal

4. Comparison

Introduction

Fully Homomorphic Encryption (FHE)

- FHE: a cryptographic primitive that allows computation on encrypted data:
  - For plaintexts a, b and their encryptions $[\![a]\!], [\![b]\!]$, we can compute
    - $[\![a]\!] \oplus [\![b]\!] = [\![a + b]\!]$
    - $[\![a]\!] \otimes [\![b]\!] = [\![a \times b]\!]$

$[\![a]\!]$: an encryption of a

- In our work, we focus on the BFV scheme:
  - Suitable for integer-wise operations ($\oplus$ and $\otimes$).
  - Supports the packing method [SV14]
    - Which enables slot-wise computations for vectors :
      - $[\![(a_0, \ldots, a_{N-1})]\!] \oplus [\![(b_0, \ldots, b_{N-1})]\!] = [\![(a_0 + b_0, \ldots, a_{N-1} + b_{N-1})]\!]$
      - $[\![(a_0, \ldots, a_{N-1})]\!] \otimes [\![(b_0, \ldots, b_{N-1})]\!] = [\![(a_0 \times b_0, \ldots, a_{N-1} \times b_{N-1})]\!]$
  - Supports automorphism operations
    - Which allows special operations (e.g., rotations, total-sum, and more)
    - And relatively faster than homomorphic multiplication.
    - σ_j : jth automorphism

[SV14] N. P. Smart and F. Vercauteren. 2014. Fully Homomorphic SIMD Operations. Designs, Codes and Cryptography 71, 1 (2014), 57–81.

Evaluation of Non-linear functions for Integer-wise FHE

- For BFV (and BGV), evaluating non-linear functions, such as division, is a challenging task.
  - The computation time of non-linear functions is significantly larger than that of the basic arithmetic operations (such as addition or multiplication).

- Prior works:
  - Okada et al. [OCHK18] have proposed a method for evaluating arbitrary bivariate functions using BGV.
    - Efficient for small input domain (few bits)
    - Too impractical for relatively large input domains
      - e.g. estimated that 15 bits division takes 117.3 days…
  - Maeda et al. [MMNFT24] have proposed a method for evaluating arbitrary bivariate functions with relatively large bit lengths **instead of special plaintext encoding**, which is much faster than the method by Okada et al.

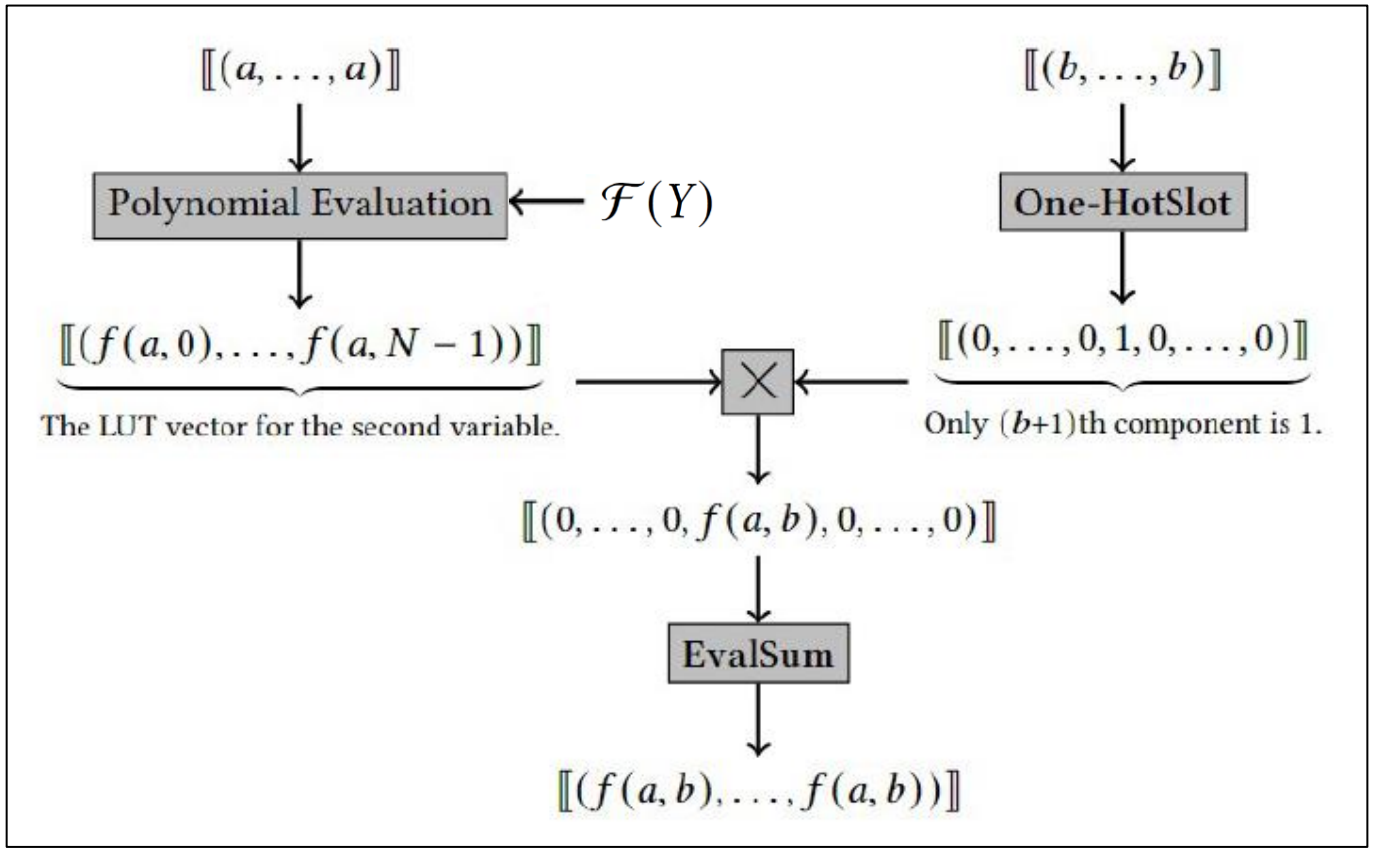
- Contribution:
  - We propose a **faster method** than the method by Maeda et al. with the same plaintext encoding.

[OCHK18] H. Okada, et al. ”Linear depth integer-wise homomorphic division.” WISTP2018, 2018
[MMNFT24] D. Maeda, K. Morimura, S. Narisada, K. Fukushima, and T. Nishide. 2024. Efficient Homomorphic Evaluation of Arbitrary Uni/Bivariate Integer Functions and Their Applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E107.A, 3 (2024), 234–247.

Prior Work by Maeda et al. [MMFFT24]

Prior Work: Evaluation of Arbitrary Bivariate Functions

- The method by Maeda et al. enables handling **relatively large input domain** in exchange for **filling all slots with a single value,** i.e., we use $[\![(a, \ldots, a)]\!]$ for a plaintext a.
- To maximize the number of slots:
  - We use a prime plaintext modulus t s.t. $t \equiv 1 \pmod{2N}$ for a ring dimension N (power of two).



Precomputation

- For a bivariate function $f : \{0,1 \ldots, N-1\} \times \{0,1 \ldots, N-1\} \to \mathbb{F}_t$, we can define a univariate function $f_y(x) = f(x, y)$ by fixing the second variable y.

- Since the range of $f_y(x)$ is included in a finite field $\mathbb{F}_t$, we can express $f_y(x)$ as a polynomial

$$f_y(x) = f_{0,y} + f_{1,y} \cdot x + \cdots + f_{N-1,y} \cdot x^{N-1} \in \mathbb{F}_t[x]$$

  for the first variable.

- For each $y = 0, 1, \ldots N-1$, precompute the coefficients of $f_y(x)$ and store each coefficient vector.

$$(f_{0,0}, \cdots, f_{i,0}, \cdots, f_{N-1,0})$$ Coefficients of $f_0(x)$

$$(f_{0,1}, \cdots, f_{i,1}, \cdots, f_{N-1,1})$$ Coefficients of $f_1(x)$

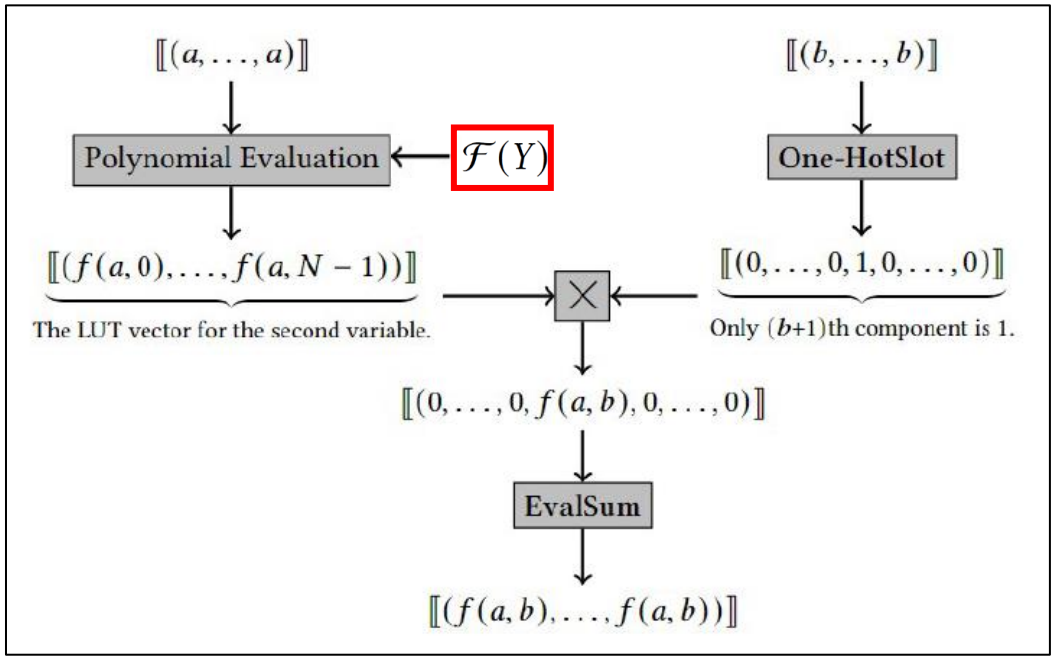
$$\vdots$$

$$(f_{0,N-1}, \cdots, f_{i,N-1}, \cdots, f_{N-1,N-1})$$ Coefficients of $f_{N-1}(x)$

- Then, we define

$$\mathcal{F}(z) := \sum_{i=0}^{N-1} [(f_{i,0}, f_{i,1}, \cdots, f_{i,N-1})] \cdot z^i \in R_t[z]$$

Packed vertically



Polynomial Evaluation for the First Variable

- By homomorphically substituting the ciphertext $[\![(a, \ldots, a)]\!]$ for the polynomial defined before

$$\mathcal{F}(z) = \sum_{i=0}^{N-1} \left[\left(f_{i,0}, f_{i,1}, \ldots, f_{i,N-1}\right)\right] \cdot z^i \in R_t[z],$$

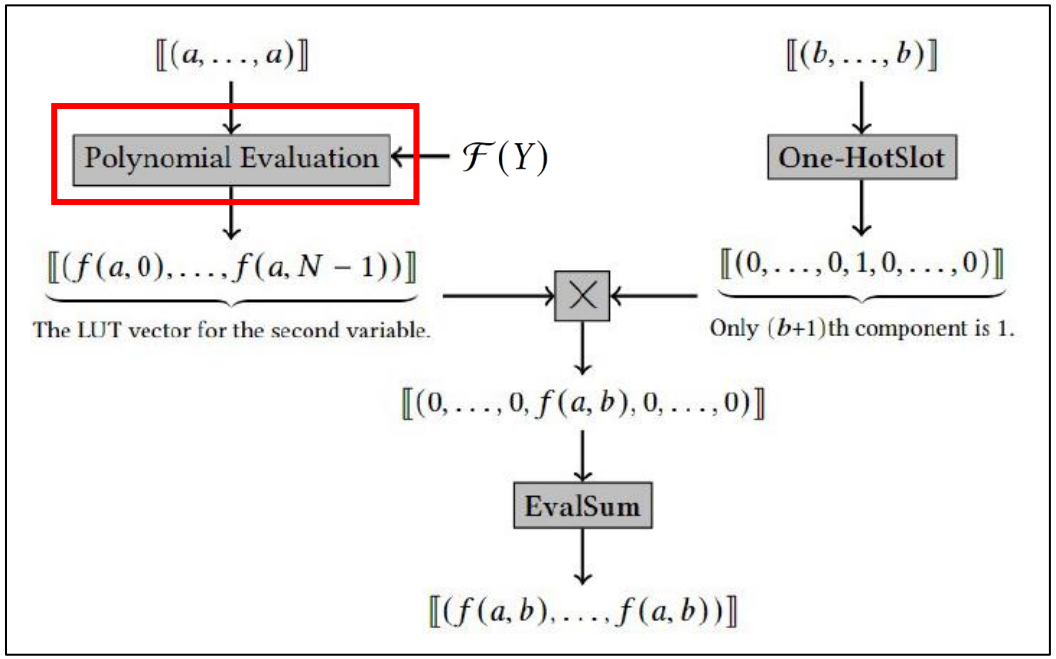
  we can obtain **a ciphertext of a LUT vector for second variables**, since

$$\mathcal{F}([\![(a, \ldots, a)]\!]) = \sum_{i=0}^{N-1} \left[\left(f_{i,0}, f_{i,1}, \ldots, f_{i,N-1}\right)\right] \cdot [\![(a^i, \ldots, a^i)]\!] = [\![(f_0(a), f_1(a), \ldots, f_{N-1}(a))]\!] = [\![(f(a,0), f(a,1) \ldots, f(a,N-1))]\!]$$

Evaluated $f_0(x), f_1(x), \ldots, f_{N-1}(x)$ in parallel.

- The above polynomial evaluation can be computed by $O(\log(N))$ depth.

- The total number of homomorphic multiplications is about $2\sqrt{N}$ via the Paterson–Stockmeyer method [PS73], but it is still too slow…



[PS72] M. Paterson and L. J. Stockmeyer. 1973. On the Number of Nonscalar Multiplications Necessary to Evaluate Polynomials. SIAM J. Comput. 2, 1 (1973), 60–66.

Conversion of the Second Variable to the One-Hot Vector

- By subtracting $[(0, 1, \ldots, N-1)]$ from the second ciphertext $[\![(b, b, \ldots, b)]\!]$

$$[\![(b, b, \ldots, b)]\!] - [(0, 1, \ldots, N-1)] = [\![(b, b-1, \ldots, 0, \ldots, b-N+1)]\!]$$

  obtain a ciphertext of a vector s.t. only $(b+1)$th component is zero and others are non-zero.

- The above ciphertext to the power t is

Thanks to Fermat little theorem

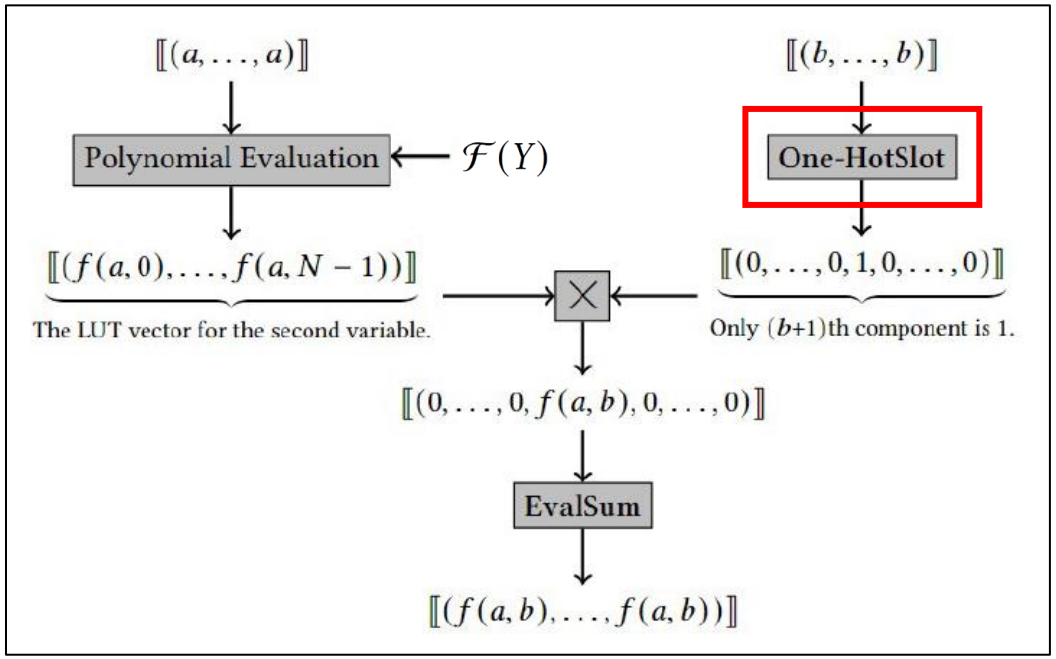
$$[\![(b, b-1, \ldots, 0, \ldots, b-N+1)]\!]^{t-1} = [\![(1, 1, \ldots, 0, \ldots, 1)]\!]$$

- By subtracting the above ciphertext from the packed plaintext $[1,1,\ldots,1]$ filled with 1, we obtain a ciphertext of a one-hot slot vector

$$[\![(0, \ldots, 1, \ldots, 0)]\!]$$

Only $(b+1)$th component is 1.

- This One-HotSlot procedure can be efficiently computed since it only requires about $\log(t)$ times homomorphic multiplications.



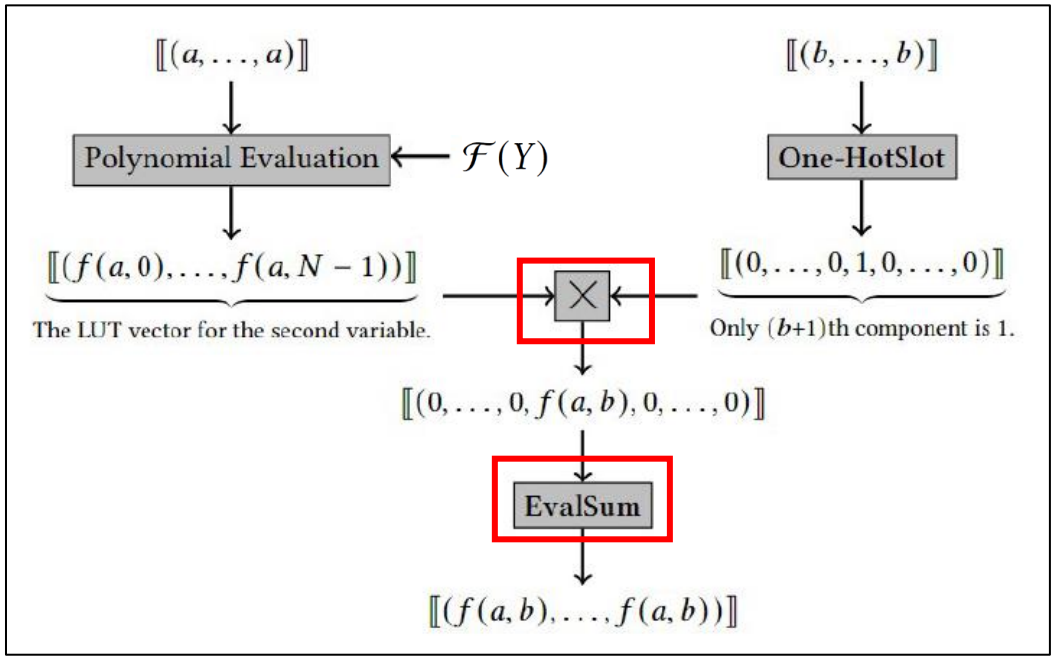
Extraction from the LUT vector

- By multiplying the ciphertext of the LUT vector and the ciphertext of the one-hot vector, we can extract $f(a,b)$

$$[\![(0,\dots,f(a,b),\dots,0)]\!] = [\![(f(a,0),\dots,f(a,N-1))]\!] \otimes [\![(0,\dots,1,\dots,0)]\!]$$

- After that, taking the summation of all slots by performing the EvalSum operation (which requires $\log(N)$ automorphisms), we finally obtain $[\![(f(a,b),\dots,f(a,b))]\!]$.

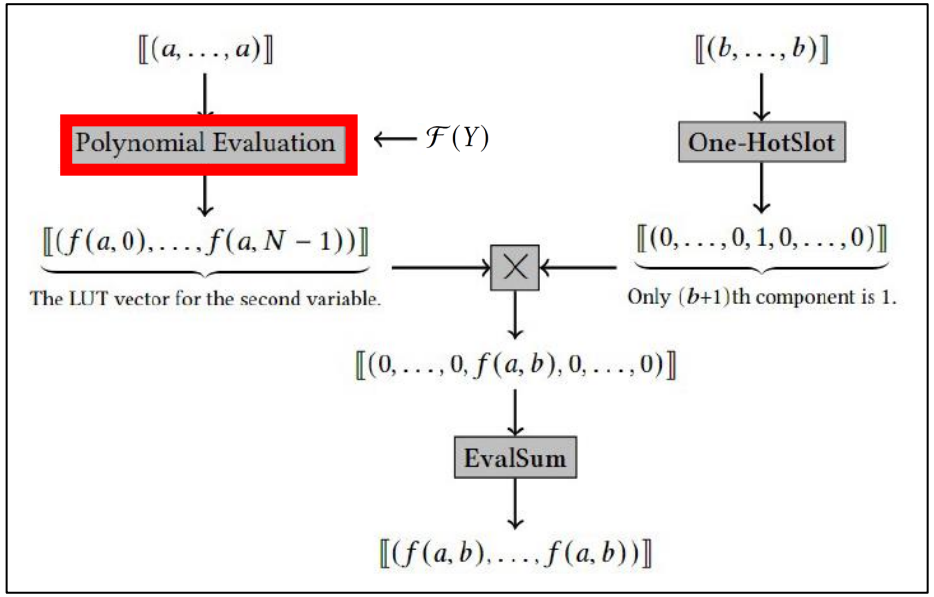
- This extraction step has few effects on the total time of the evaluation of a bivariate function since this step only requires
  - Single homomorphic multiplication
  - $\log(N)$ automorphisms
    - Automorphism is a relatively faster operation than homomorphic multiplication.



Our Proposal

Idea of Our Proposal

- In the prior work by Maeda et al., transforming a first variable ciphertext $[\![(a,\ldots,a)]\!]$ to a LUT vector $[\![(f(a,0),\ldots,f(a,N-1))]\!]$ is the most time-consuming part,
  since it requires **many homomorphic multiplications**
  during evaluating the predefined polynomial.



- Our idea is to use a lookup table as a **matrix:**

$$F = \begin{pmatrix} f(0,0) & \cdots & f(0,N-1) \\ f(1,0) & \cdots & f(1,N-1) \\ \vdots & \ddots & \vdots \\ f(N-1,0) & \cdots & f(N-1,N-1) \end{pmatrix}$$

- We can multiply the matrix F to a packed ciphertext via **homomorphic linear transformation.**

Homomorphic Linear Transformation

- Homomorphic linear transformation [HS18] is a technique for multiplying a packed ciphertext by a matrix, i.e., for a packed ciphertext $[\![\vec{a}]\!] = [\![(a_0, \ldots, a_{N-1})]\!]$ and a marix M (square matrix of order N), we can compute

$$[\![\vec{a} \cdot M]\!] = [\![\vec{a}]\!] \cdot M$$

  via homomorphic linear transformation.

- When a square matrix of order N does not need to be encrypted, homomorphic linear transformation canperform by about $\sqrt{2N}$ times automorphisms (via the baby-step giant-step technique).

- Then, homomorphic linear transformation for a square matrix of order N must be faster than the evaluation of a degree-$(N-1)$ polynomial (which requires about $2\sqrt{N}$ times homomorphic multiplications)

Algorithm 5 HLT

Input: $\boldsymbol{c} = [\![\boldsymbol{a}]\!], M \in \mathbb{F}_t^{N \times N}$
Output: $[\![\boldsymbol{a} \cdot M]\!]$

1: (Precomputing) : For a matrix M, compute $\overline{\boldsymbol{m}}_{5^{\alpha+A\cdot\beta}}, \underline{\boldsymbol{m}}_{(-1)\cdot 5^{\alpha+A\cdot\beta}}$ $(0 \le \alpha \le A-1, 0 \le \beta \le B-1)$.
2: $\boldsymbol{d}_{\text{digits}} \leftarrow \mathbf{Ext}_{Q\to PQ}(\mathcal{D}(c_1))$
3: Put $\boldsymbol{d}_{\text{digits}}$ in NTT format
4: $\boldsymbol{c}_L \leftarrow [0], \boldsymbol{c}_R \leftarrow [0]$
5: **for** $\beta = 0$ to $B-1$ **do**
6: $\quad \boldsymbol{c}_{L_{\text{BS}}} \leftarrow \boldsymbol{c} \cdot [\overline{\boldsymbol{m}}_{5^{A\cdot\beta}}], \boldsymbol{c}_{R_{\text{BS}}} \leftarrow \boldsymbol{c} \cdot [\underline{\boldsymbol{m}}_{-5^{A\cdot\beta}}]$ // $\alpha = 0$
7: $\quad$ **for** $\alpha = 1$ to $A-1$ **do**
8: $\quad\quad \tilde{\boldsymbol{c}}_1 \leftarrow [\boldsymbol{d}_{\text{digits}} \cdot \mathbf{ksk}_{s\to\sigma_{5^\alpha}^{-1}(s)}]_{PQ}$
9: $\quad\quad \sigma_{5^\alpha}(\boldsymbol{c}) \leftarrow (\sigma_{5^\alpha}(c_0) + \lfloor \sigma_{5^\alpha}(\tilde{c}_{1,0})/P \rceil, \lfloor \sigma_{5^\alpha}(\tilde{c}_{1,1})/P \rceil)$
10: $\quad\quad \boldsymbol{c}_{L_{\text{BS}}} \leftarrow \boldsymbol{c}_{L_{\text{BS}}} \oplus \sigma_{5^\alpha}(\boldsymbol{c}) \cdot [\overline{\boldsymbol{m}}_{5^{\alpha+A\cdot\beta}}]$
11: $\quad\quad \boldsymbol{c}_{R_{\text{BS}}} \leftarrow \boldsymbol{c}_{R_{\text{BS}}} \oplus \sigma_{5^\alpha}(\boldsymbol{c}) \cdot [\underline{\boldsymbol{m}}_{-5^{\alpha+A\cdot\beta}}]$
12: $\quad \boldsymbol{c}_L \leftarrow \boldsymbol{c}_L \oplus \sigma_{5^{A\cdot\beta}}(\boldsymbol{c}_{L_{\text{BS}}})$
13: $\quad \boldsymbol{c}_R \leftarrow \boldsymbol{c}_R \oplus \sigma_{5^{A\cdot\beta}}(\boldsymbol{c}_{R_{\text{BS}}})$
14: $\tilde{\boldsymbol{c}} \leftarrow \boldsymbol{c}_L \oplus \sigma_{-1}(\boldsymbol{c}_R)$
15: **return** $\tilde{\boldsymbol{c}} (= [\![\boldsymbol{a} \cdot M]\!])$

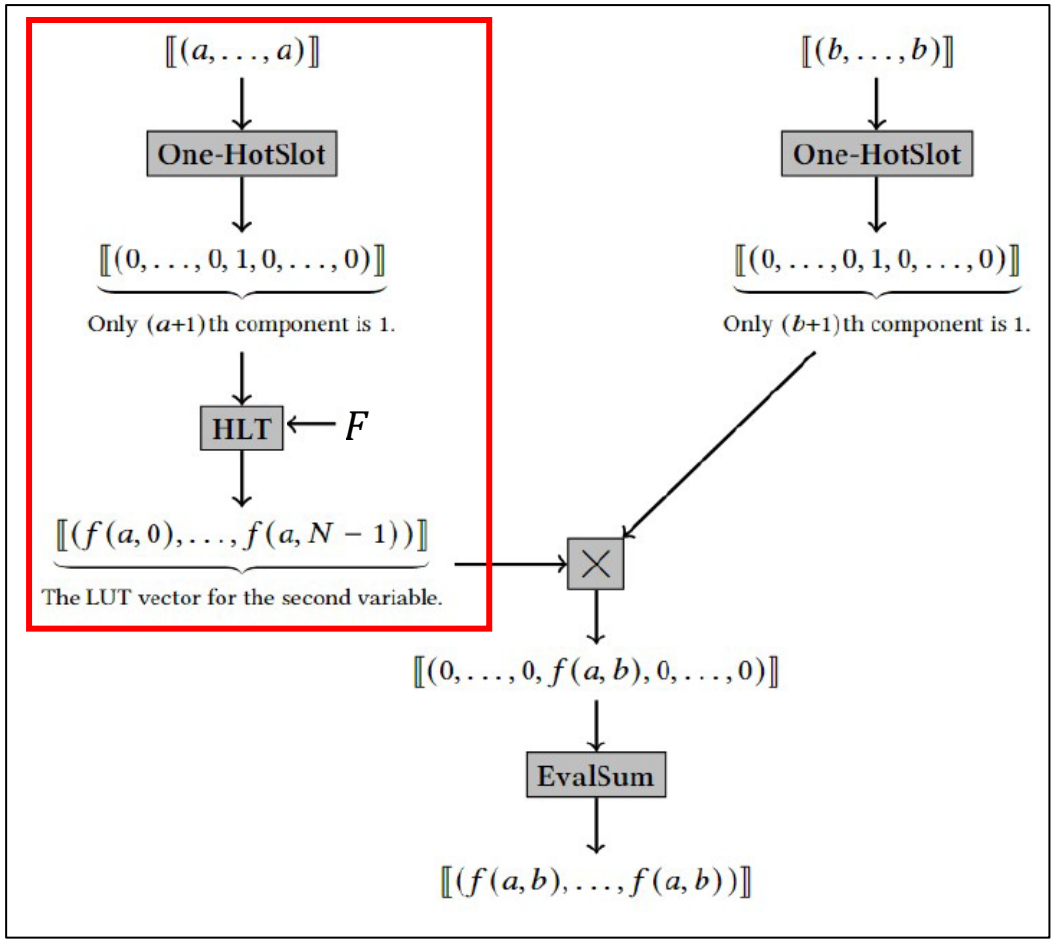
[HS18] S. Halevi and V. Shoup. 2018. Faster Homomorphic Linear Transformations in HElib. In Advances in Cryptology – CRYPTO 2018 (Lecture Notes in Computer Science). Springer, 93–120.

Our Proposal

- We can efficiently translate $[\![(a, \ldots, a)]\!]$ to a ciphertext of a one-hot vector $[\![(0, \ldots, 1, \ldots, 0)]\!]$ for the first variable.
- The one-hot vector for the first variable a can be used to extract the $(a+1)$th row of the matrix F by multiplying it homomorphically.

$$\underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{Only } (a+1)\text{th component is 1.}} \cdot \begin{pmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(a,0) & \cdots & f(a,N-1) \\ \vdots & \ddots & \vdots \\ f(N-1,0) & \cdots & f(N-1,N-1) \end{pmatrix}$$

- Eliminate the polynomial evaluation process, which is the most time-consuming part.



Comparison

Complexity

- Maeda et al.'s method requires
  - $O(\sqrt{N})$ homomorphic multiplications (for the polynomial evaluation and One-HotSlot)
  - $O(\log(N))$ automorphisms (for EvalSum)

- Our proposal requires:
  - $O(\log(N))$ homomorphic multiplications (for One-HotSlot × 2)
  - $O(\sqrt{N})$ automorphisms (for homomorphic linear transformation and EvalSum)

- This leads to an efficiency gap between these methods.

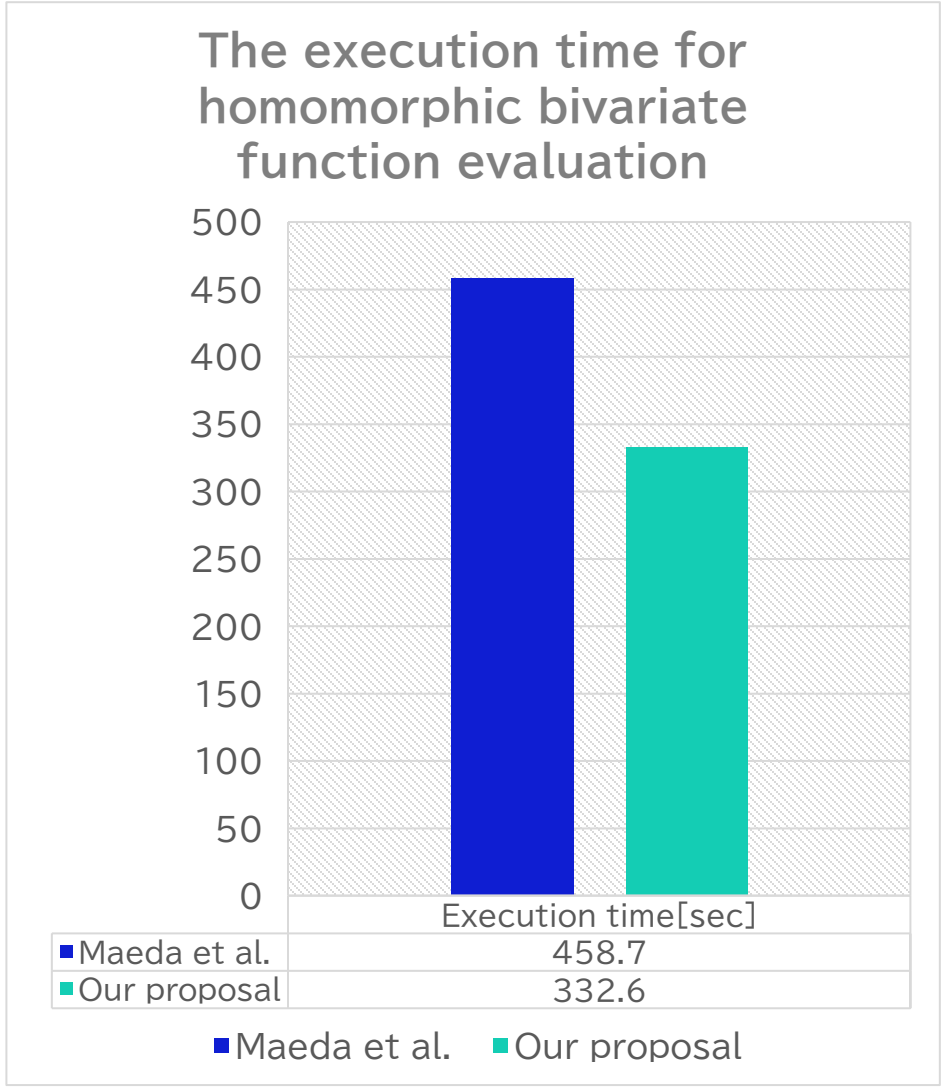
- As a drawback, our proposal requires more key-switching keys : $O(\log(N)) \rightarrow O(\sqrt{N})$

Method	#-Homomorphic multiplications	#-Automorphisms
Maeda et al.	$O(\sqrt{N})$	$O(\log(N))$
Ours	$O(\log(N))$	$O(\sqrt{N})$

[MMNFT24] D. Maeda, K. Morimura, S. Narisada, K. Fukushima, and T. Nishide. 2024. Efficient Homomorphic Evaluation of Arbitrary Uni/Bivariate Integer Functions and Their Applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E107.A, 3 (2024), 234–247.

Implementation Result

- Experimental Environment:
  - CPU: Intel Xeon Silver 4114@2.20GHz
  - Memories: 96GB
  - Library: OpenFHE
  - Scheme: BFVrns
  - Security Parameter: HEStd_128_classic
- The input domain is $\{0,1,\dots,N-1\} \times \{0,1,\dots,N-1\}$ for $N = 2^{15}$.
- We choose the division function $f(x,y) = \lfloor x/y \rfloor$ as a representative non-linear bivariate function.
- Our proposed method reduces the execution time to **72.5%**, which decreases the total time by **126.1 seconds**.



Conclusions

- We proposed a method for evaluating arbitrary bivariate functions, which is faster than the method proposed by Maeda et al.
  - Less homomorphic multiplications ($O(\sqrt{N}) \to O(\log(N))$), more automorphisms ($O(\log(N)) \to O(\sqrt{N})$)
    - It decreases the computation time
    - But it requires more key-switching keys… ($O(\log(N)) \to O(\sqrt{N})$)
  - Through the implementation by OpenFHE, we confirm that our method achieved a speed-up compared to the previous method.

- Future work:
  - Extend input domain from N to t ($> 2N$) by using multiple ciphertexts similar to [MMNFT24], and implement it.

[MMNFT24] D. Maeda, K. Morimura, S. Narisada, K. Fukushima, and T. Nishide. 2024. Efficient Homomorphic Evaluation of Arbitrary Uni/Bivariate Integer Functions and Their Applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E107.A, 3 (2024), 234–247.

\Orchestrating a brighter world

Estimated Memory Usage

- For the parameters in our experiment, the size of each key switching key ksk is estimated 20.6 MB.
  - The method by Maeda et al. requires 289.0 MB
  - The proposed method requires 6874.4 MB (≒ 6.9 GB)
    - Almost 24× memory usage.

[MMNFT24] D. Maeda, K. Morimura, S. Narisada, K. Fukushima, and T. Nishide. 2024. Efficient Homomorphic Evaluation of Arbitrary Uni/Bivariate Integer Functions and Their Applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E107.A, 3 (2024), 234–247.