Plaintext-Ciphertext Matrix Multiplication and FHE Bootstrapping: Fast and Fused

Youngjin Bae, Jung Hee Cheon, Guillaume Hanrot,
Jai Hyun Park, Damien Stehlé

Summary

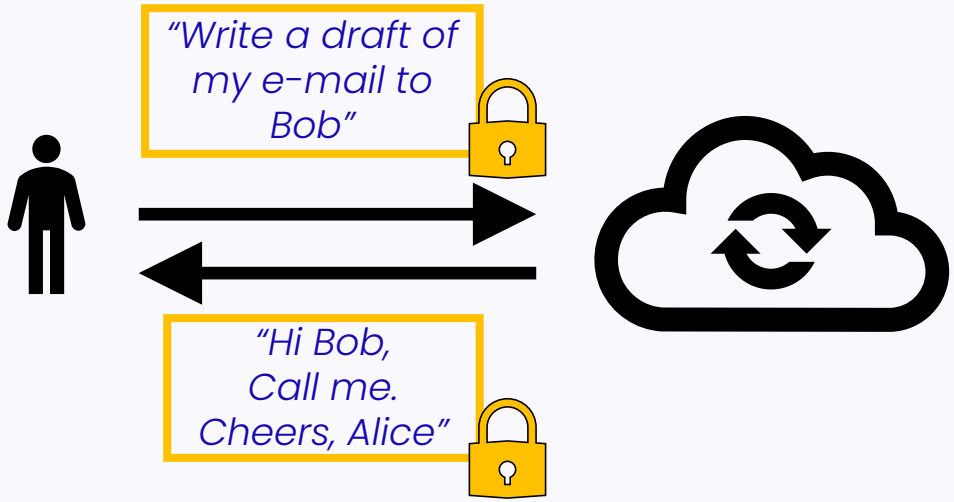
- Fast plaintext-ciphertext matrix multiplication (PCMM)
  - 0.31s for PCMM with 256×256 matrices in a single thread CPU.
  - How? Reduce PCMM to plaintext matrix multiplications.
- Batch bootstrapping (BTS) with high throughput
  - 2x for 32 batches
- Fused PCMM with batch BTS
  - 41% higher throughput than current BTS without PCMM

Matrix Multiplication

- Matrix multiplication is central in high-performance computing
  - highly optimized libraries for basic linear algebra subprograms (BLAS)
  - Can be 10x faster than a naïve implementation for large matrices

What about matrix multiplication on encrypted data?

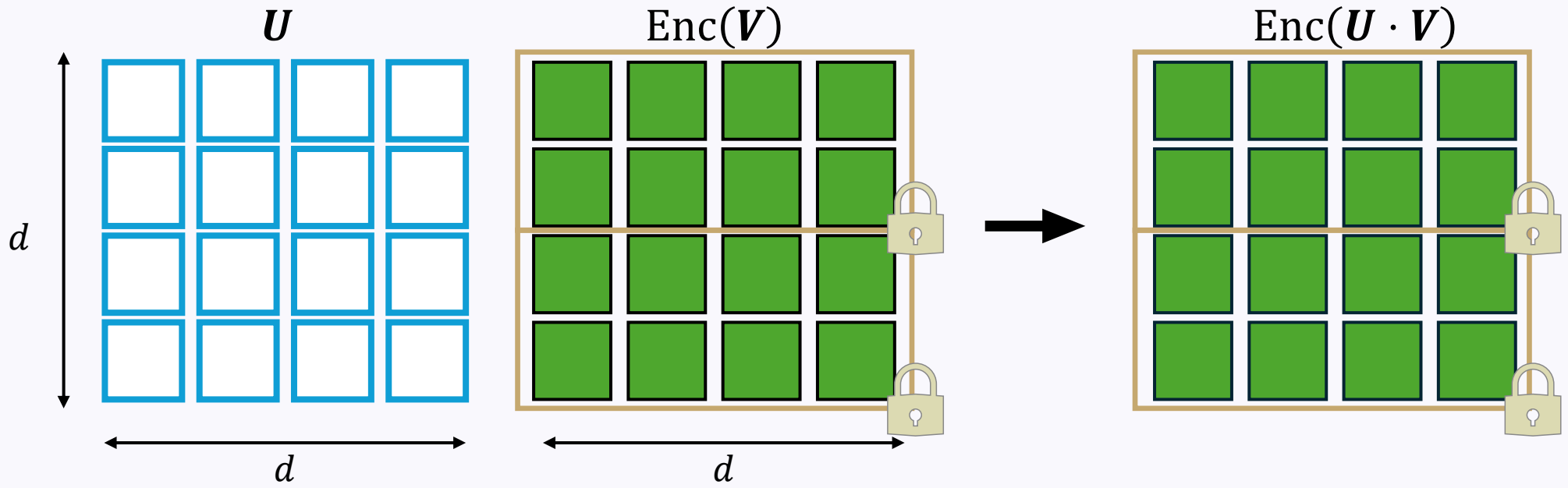
Privacy-preserving machine learning as a service



- PPMM: plaintext-plaintext matrix multiplication
- PCMM: plaintext-ciphertext matrix multiplication
- CCMM: ciphertext-ciphertext matrix multiplication

- PCMMs and CCMMs with diverse dimensions
  - e.g., PCMM of dimension 128 ~ 16384 for GPT-3.5

Plaintext-Ciphertext Matrix Multiplication (PCMM)

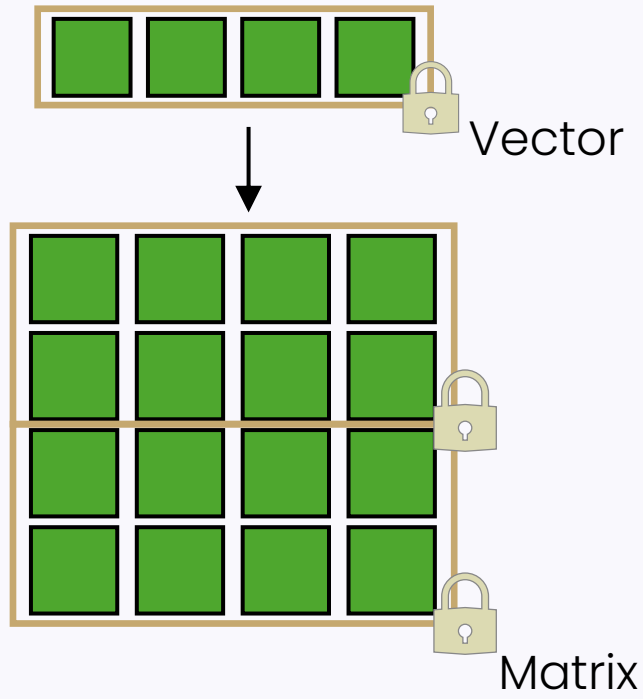


- Multiplication between a plaintext matrix and a ciphertext matrix.
  - $d \times d \times d$ PCMM

- PCMM with RLWE-based (fully) homomorphic encryption schemes (CKKS)
  - Compatibility with the other machine learning tasks
  - High efficiency

PCMM with CKKS

- CKKS
  - Plaintext: vector of real numbers
  - Native operations: // add, // mult, and rotate.



- With the native operations, PCMM requires lots of rotates.
  - For example, [JKLS18] has a cubic bit complexity, but is orders of magnitude slower than PPMM.

- Questions
  1. How to utilize PPMM BLAS libraries?
  2. How to handle multiple ciphertexts?

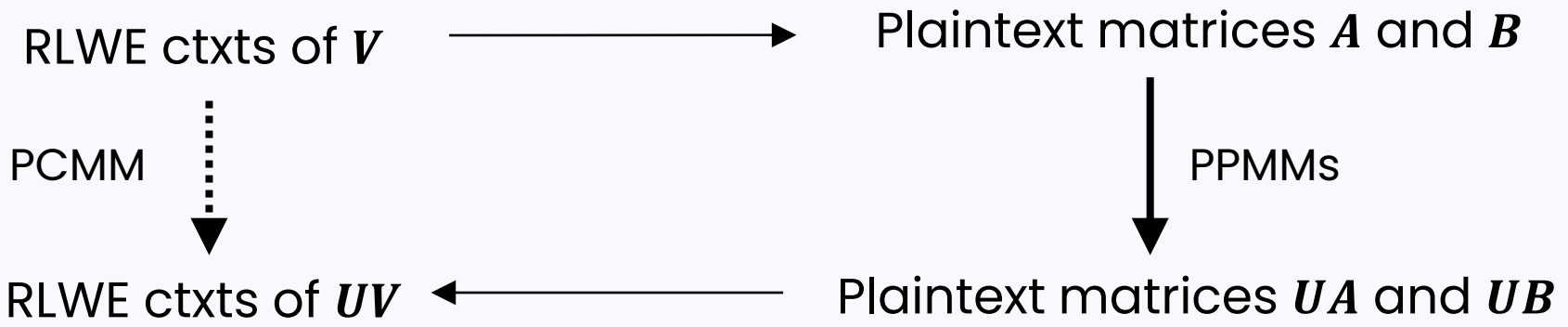
✓Reduction from PCMM to PPMM

✓Batch ciphertext computation

[JKLS18] X. Jiang, M. Kim, K. Lauter, and Y. Song. Secure Outsourced Matrix Computation and Application to Neural Networks. In CCS, 2018.

Reduction from PCMM to PPMMs

- [LZ22] considers verifiable PCMM
  - Performs one PCMM using two PPMMs
  - Restriction: $d \geq N$



- This is a great idea for fast PCMM: we can use BLAS libraries

[LZ22] J. Liu and L. F. Zhang. *Privacy-preserving and publicly verifiable matrix multiplication*. IEEE Trans. on Services Computing, 2022

RLWE-based Encryption of Matrices

- In the ring $\mathbb{Z}_Q[X]/(X^N+1)$, an RLWE ciphertext $(a, b = as + m)$ is:

$$[a_0 \quad a_1 \quad \cdots \quad a_{N-1}] \begin{bmatrix} s_0 & s_1 & \cdots & s_{N-1} \\ -s_{N-1} & s_0 & \cdots & s_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ -s_1 & -s_2 & \cdots & s_0 \end{bmatrix} + [b_0 \quad b_1 \quad \cdots \quad b_{N-1}] = [m_0 \quad m_1 \quad \cdots \quad m_{N-1}]$$

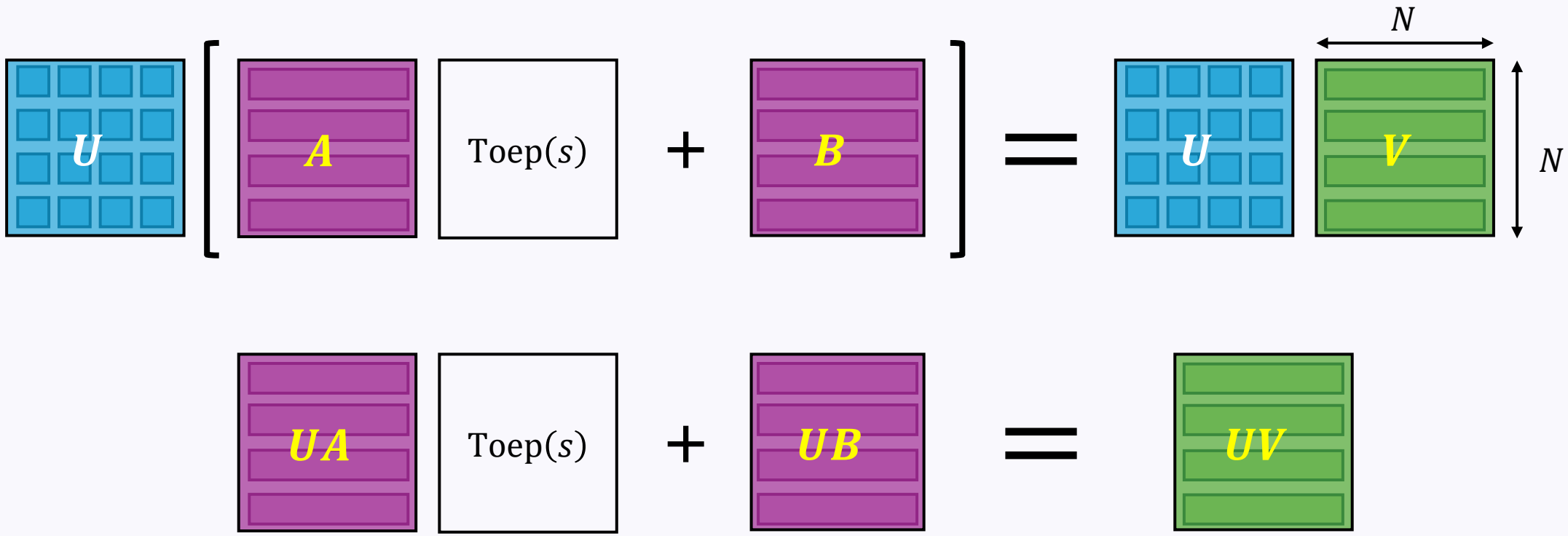
$\mathrm{Toep}(s)$

✓ a_i, b_i, s_i, m_i are coeffs of a, b, s, m

- N RLWE ciphertexts (with a shared secret) are:

$$\begin{bmatrix} — & a_1^t & — \\ — & a_2^t & — \\ & \boldsymbol{A} & \\ & \vdots & \\ — & a_N^t & — \end{bmatrix} \begin{bmatrix} s_0 & s_1 & \cdots & s_{N-1} \\ -s_{N-1} & s_0 & \cdots & s_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ -s_1 & -s_2 & \cdots & s_0 \end{bmatrix} + \begin{bmatrix} — & b_1^t & — \\ — & b_2^t & — \\ & \boldsymbol{B} & \\ & \vdots & \\ — & b_N^t & — \end{bmatrix} = \begin{bmatrix} — & m_1^t & — \\ — & m_2^t & — \\ & \boldsymbol{M} & \\ & \vdots & \\ — & m_N^t & — \end{bmatrix}$$

RLWE PCMM ≤ PPMMs

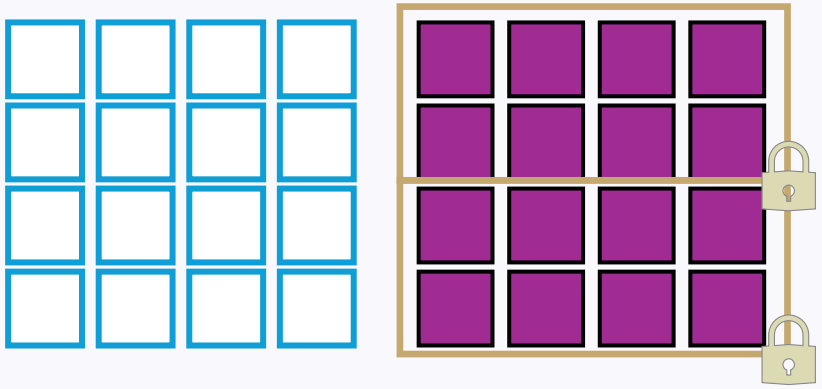


- $N \times N \times N$ PCMM $\leq$ *two* $N \times N \times N$ PPMMs *modulo* Q
- We use PPMM BLAS libraries for PCMM

PCMM with Small/Large Matrices

Small matrices

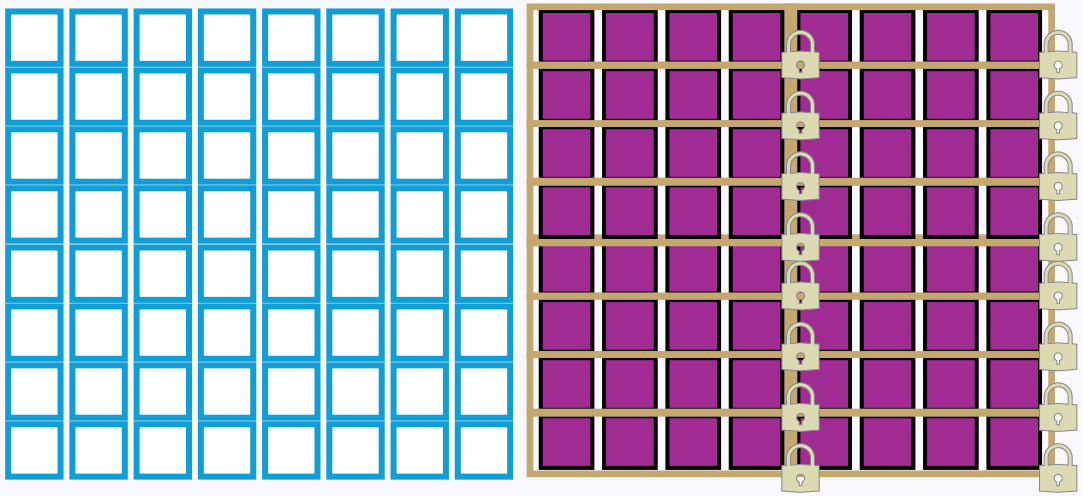
- Data moves within ciphertexts



Can we reduce PCMM to PPMMs?

Large matrices

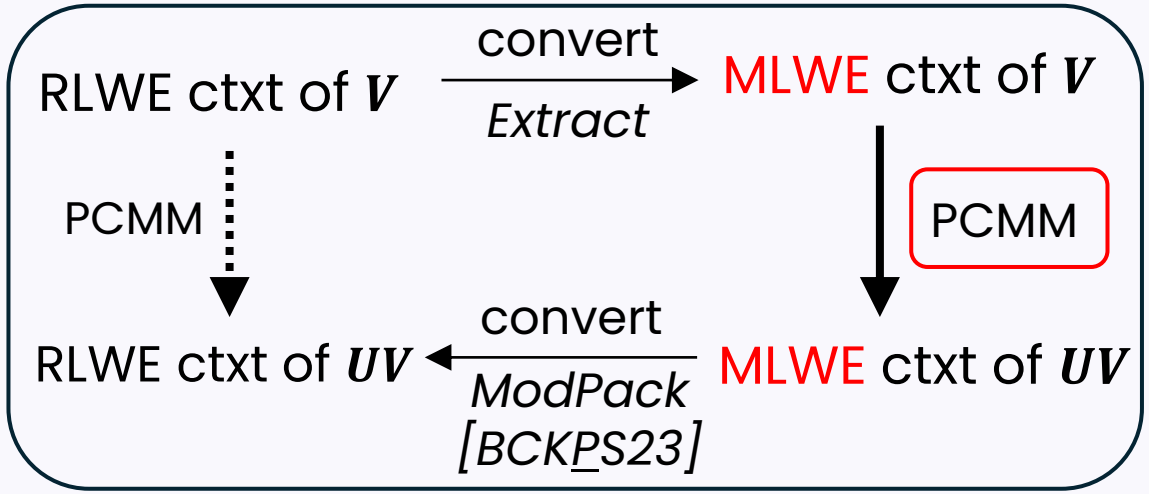
- Latency of 2 PPMMs is not satisfactory



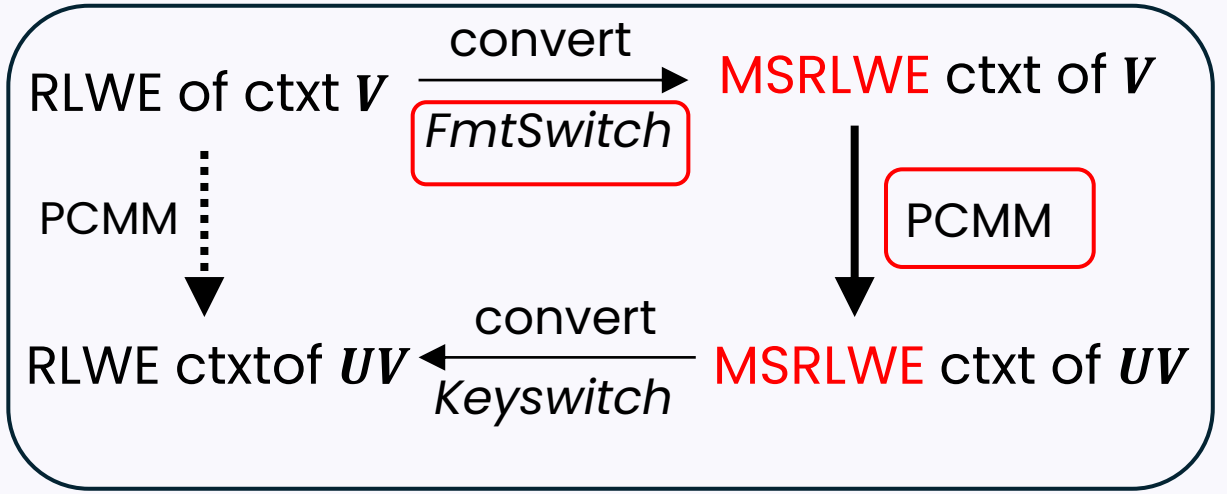
Can we do better than 2 PPMMs?

Different Formats for Various Dimensions

- Module LWE (MLWE) and Multi-secret RLWE (MSRLWE) to reduce various dimensional PCMMs to PPMMs.



Small matrices

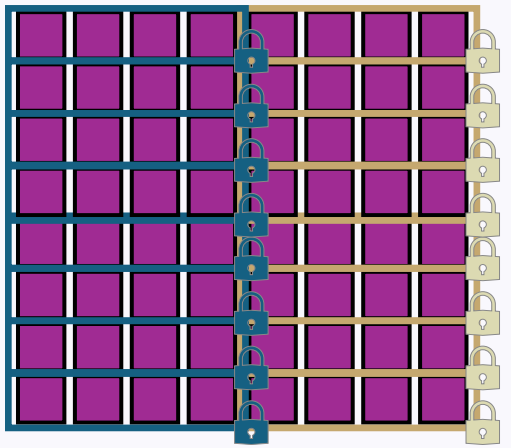


Large matrices

- Format conversions are negligible unless matrices are small.

[BCKPS23] Y. Bae, J. H. Cheon, J. Kim, J. H. Park, and D. Stehlé. HERMES: efficient ring packing using MLWE ciphertexts and application to transciphering. In CRYPTO, 2023.

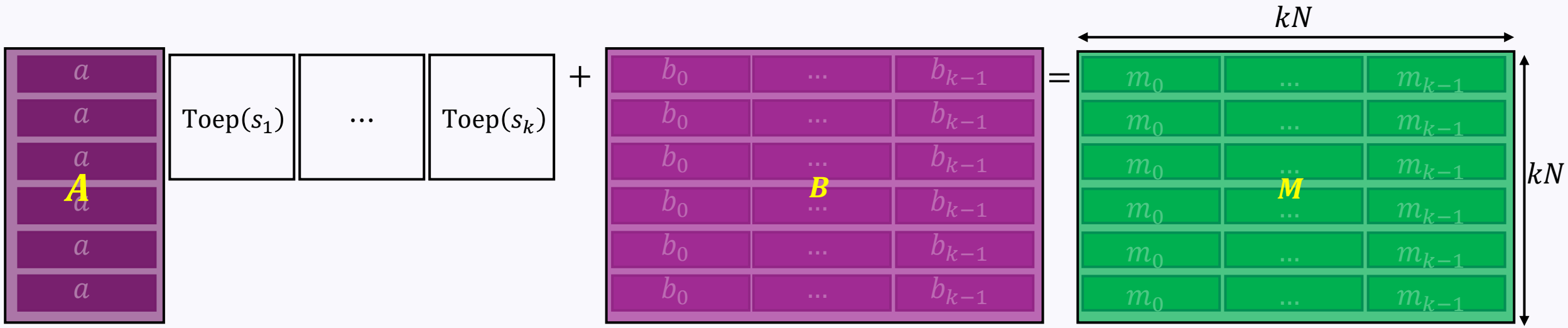
MSRLWE for Large Matrices



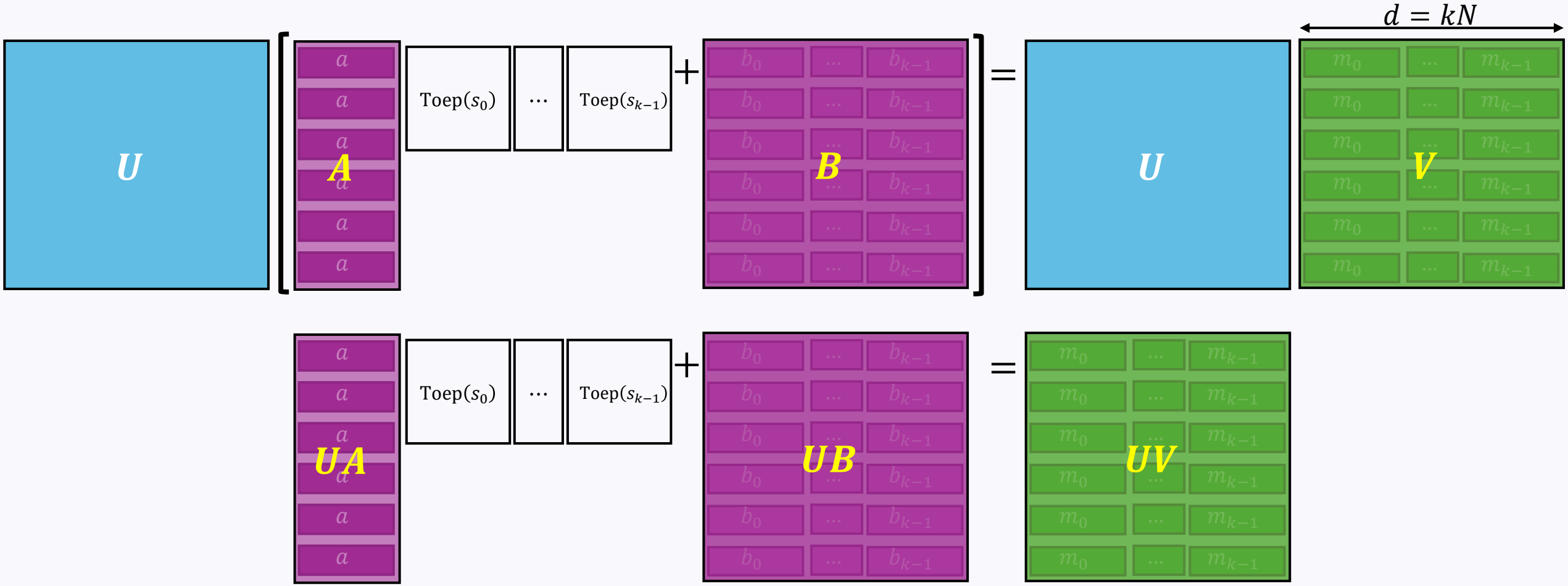
- MSRLWE ciphertexts share a-part rather than secret.

$$as_j + b_j = m_j \quad \forall j \in [k]$$

where s_j is a different secret for each j.

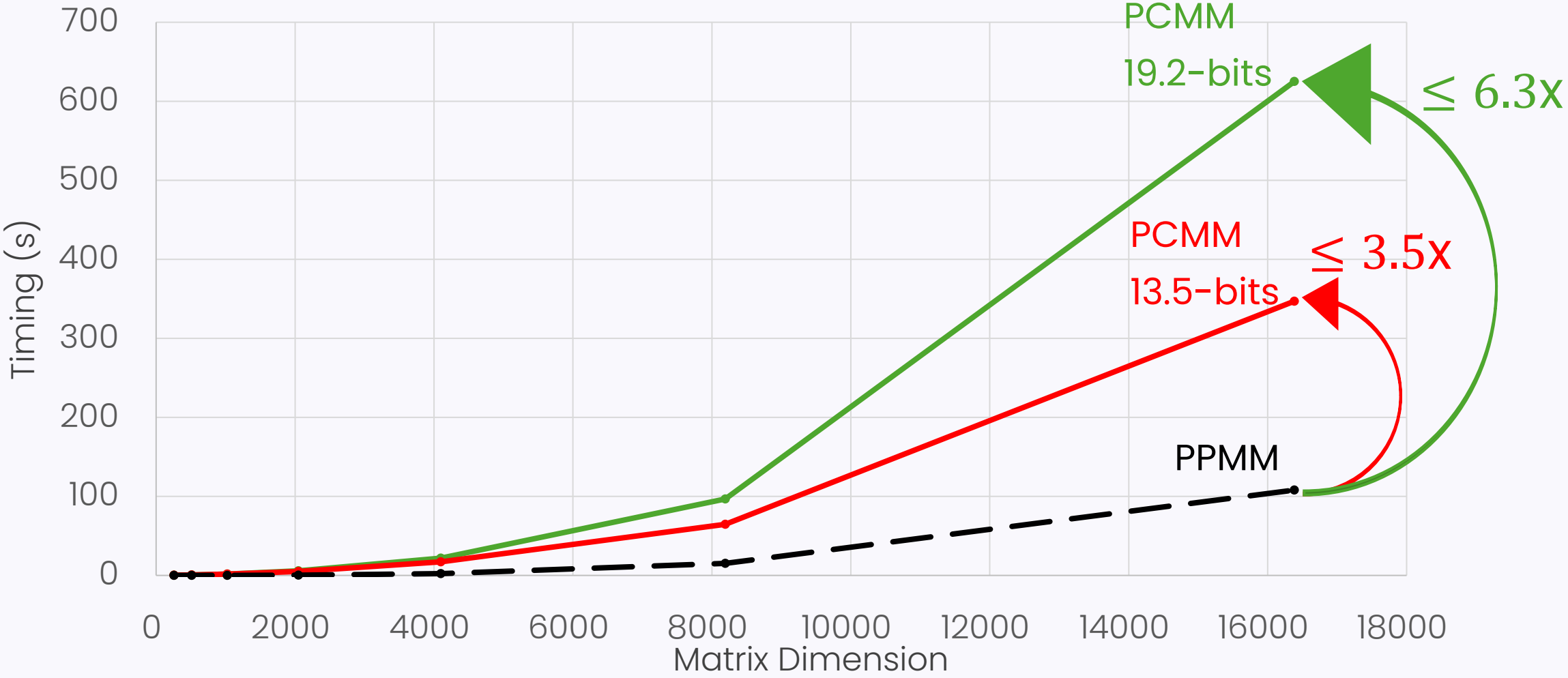


MSRLWE PCMM ≤ PPMMs



- $(d > N)$ $d \times d \times d$ *MSRLWE PCMM* ≤ *two PPMMs modulo* Q
- PPMM UA is easier than UV

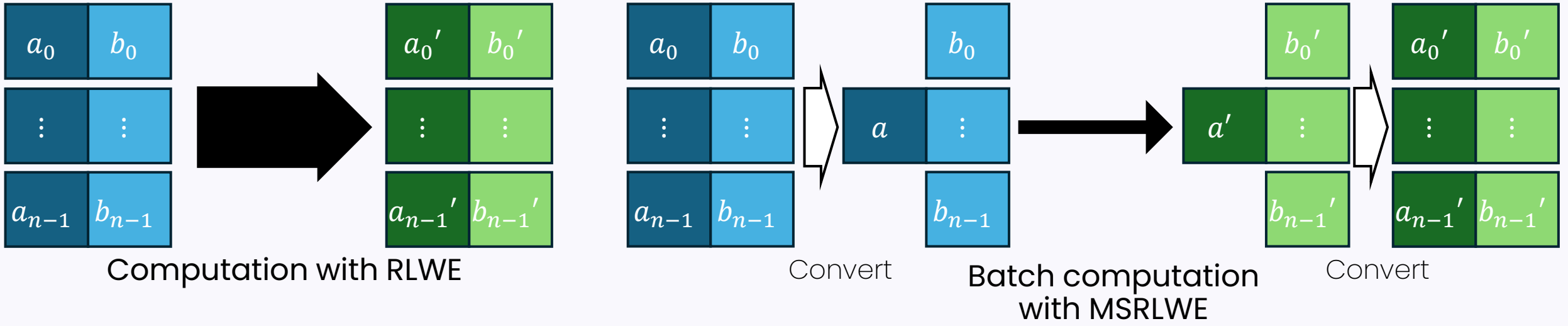
Experimental Results



Intel® Xeon® Gold 6242 CPU at 2.80GHz, single thread
HEaaN library for HE, OpenBLAS for PPMM.

MSRLWE for Batch Computation

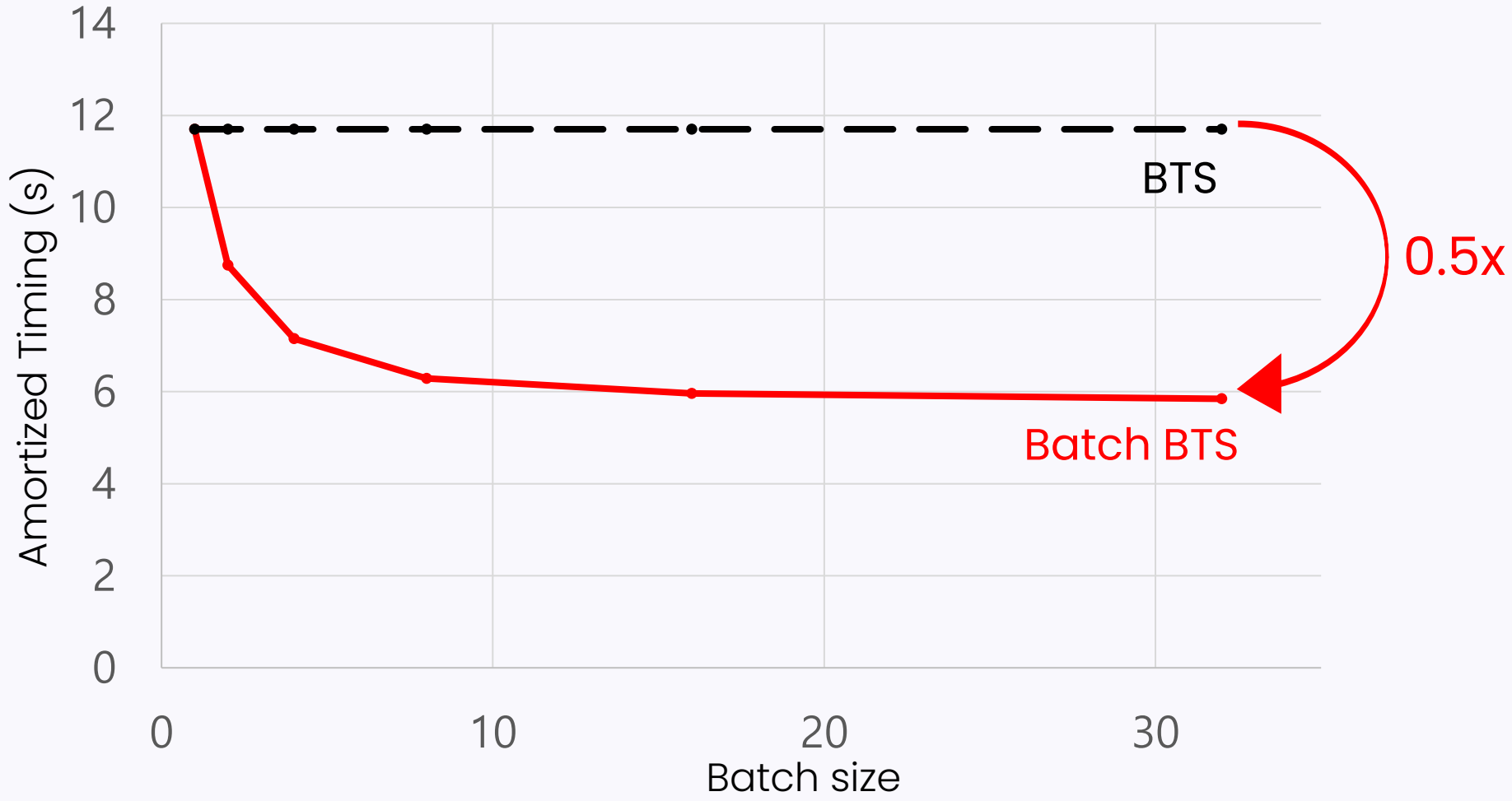
- Another question: how to manage the multiple ciphertexts involved in PCMM?



- Native batch operations using MSRLWE
  - // add, // ptxt-ctxt mult, rotate
  - No // ctxt-ctxt mult

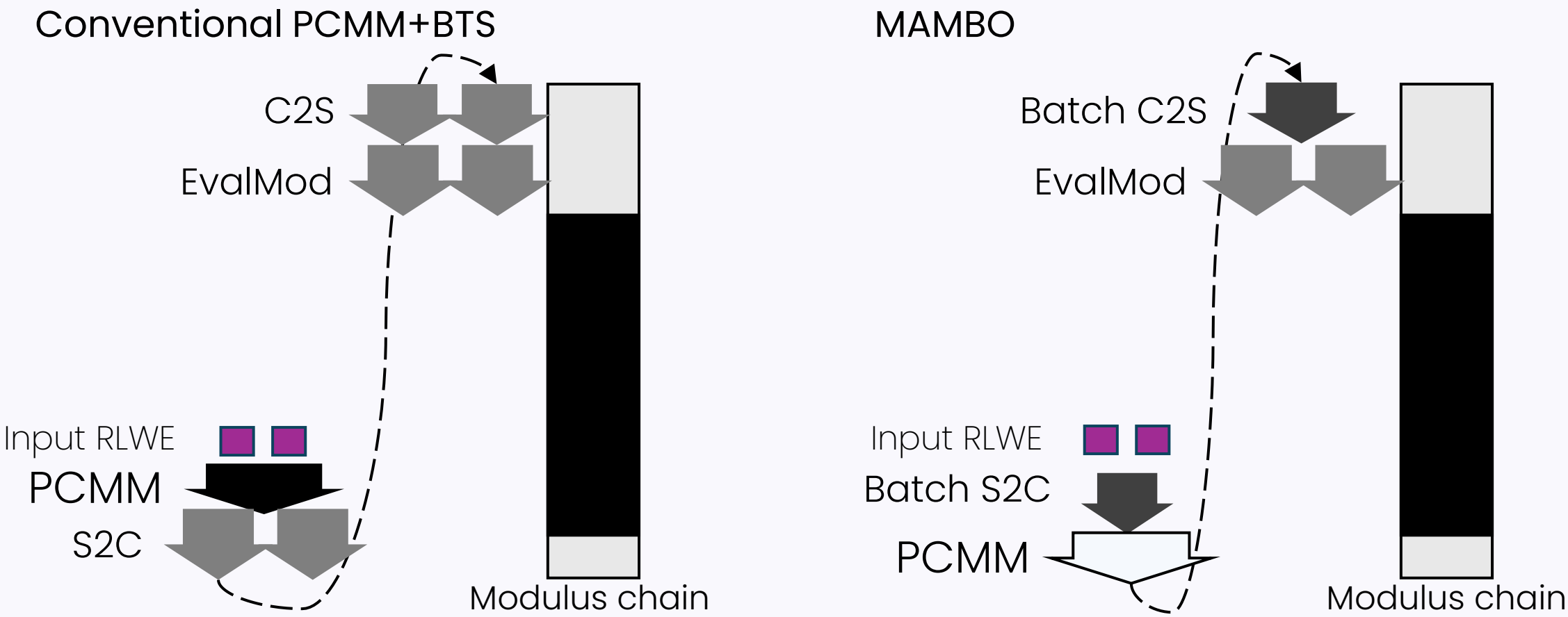
✓**Batch FHE bootstrapping:** batch C2S and S2C during bootstrapping
  - S2C and C2S do not require ctxt-ctxt mult.

Experimental Results



Intel® Xeon® Gold 6242 CPU at 2.80GHz, single thread

MAMBO: Fused PCMM and Batch Bootstrapping



✓ For large matrices, MAMBO is 41% faster than current bootstrapping without PCMM.

Wrapping Up!

- **Fast PCMM**
  - Uses various ciphertext formats for various dimensions
  - Exploits efficiency of BLAS libraries
  - Our PCMM is only ≤ 3.5x slower than PPMM
- **Even faster PCMM with a fixed plaintext matrix** (see paper)
- **Conversion from RLWE formats to MSRLWE formats** (see paper)
- **Batch FHE bootstrapping**
  - MAMBO: Fused PCMM and Batch Bootstrapping
  - 41% faster than current bootstrapping without PCMM

eprint: 2024/1284

Thank you!