Ciphertext-Ciphertext Matrix Multiplication: Fast for Large Matrices

Jai Hyun Park

jaihyunp@gmail.com

HEAAN CRYPTO LAB

FHE.org – June 5, 2025

Summary

- Fast ciphertext-ciphertext matrix multiplication (CCMM)
  - 85.2 s for CCMM of 4096×4096 matrices in a single thread CPU
  - How? Reduce CCMM to plaintext matrix multiplications
- Fast ciphertext matrix transpose (CMT)
  - 0.76 s for CMT of a 2048×2048 matrices in a single thread CPU
- Lightweight CCMM and CMT algorithms with smaller key sizes

Matrix Multiplication

- Matrix multiplication is central in high-performance computing
  - highly optimized libraries for basic linear algebra subprograms (BLAS)
  - Can be 10-30x faster than a naïve implementation for large matrices

Matrix Multiplication

- Matrix multiplication is central in high-performance computing
  - highly optimized libraries for basic linear algebra subprograms (BLAS)
  - Can be 10-30x faster than a naïve implementation for large matrices

What about matrix multiplication on encrypted data?

Matrix Multiplication

- Matrix multiplication is central in high-performance computing
  - highly optimized libraries for basic linear algebra subprograms (BLAS)
  - Can be 10-30x faster than a naïve implementation for large matrices

What about matrix multiplication on encrypted data?

Privacy-preserving machine learning as a service

Matrix Multiplication

- Matrix multiplication is central in high-performance computing
  - highly optimized libraries for basic linear algebra subprograms (BLAS)
  - Can be 10-30x 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

PPMM vs. PCMM vs. CCMM

PPMM vs. PCMM vs. CCMM

- PPMM – BLAS libraries
  - highly optimized open libraries
  - Can be 30x faster than a naïve implementation

For matrix dimension 2^{12}:

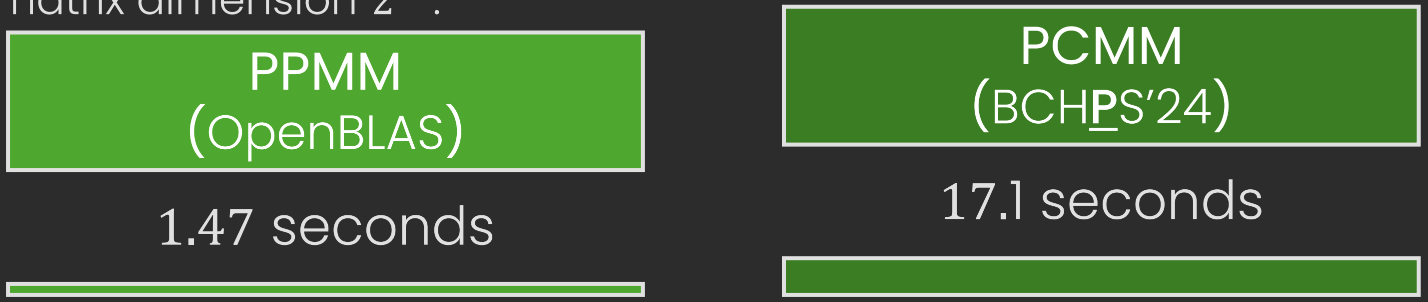
PPMM
(OpenBLAS)

1.47 seconds

PPMM vs. PCMM vs. CCMM

- PPMM – BLAS libraries
  - highly optimized open libraries
  - Can be 30x faster than a naïve implementation
- PCMM – BCHPS'24
  - Reduction from PCMM to PPMM
  - Optimizations with shared-a, truncation, and others

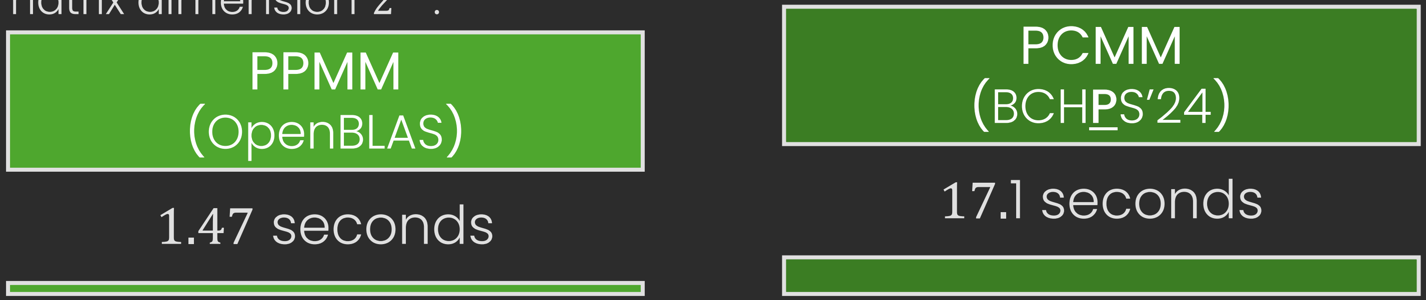
For matrix dimension 2^{12}:



PPMM vs. PCMM vs. CCMM

- PPMM – BLAS libraries
  - highly optimized open libraries
  - Can be 30x faster than a naïve implementation
- PCMM – BCHPS'24
  - Reduction from PCMM to PPMM
  - Optimizations with shared-a, truncation, and others
- CCMM – JKLS'18
  - cubic bit complexity
  - 0.6 seconds for matrix dimension 64

For matrix dimension 2^{12}:



PPMM vs. PCMM vs. CCMM

- PPMM – BLAS libraries
  - highly optimized open libraries
  - Can be 30x faster than a naïve implementation
- PCMM – BCH<u>P</u>S'24
  - Reduction from PCMM to PPMM
  - Optimizations with shared-a, truncation, and others
- CCMM – JKLS'18
  - cubic bit complexity
  - 0.6 seconds for matrix dimension 64

For matrix dimension 2^{12}:

PPMM (OpenBLAS)

1.47 seconds

PCMM (BCH<u>P</u>S'24)

17.1 seconds

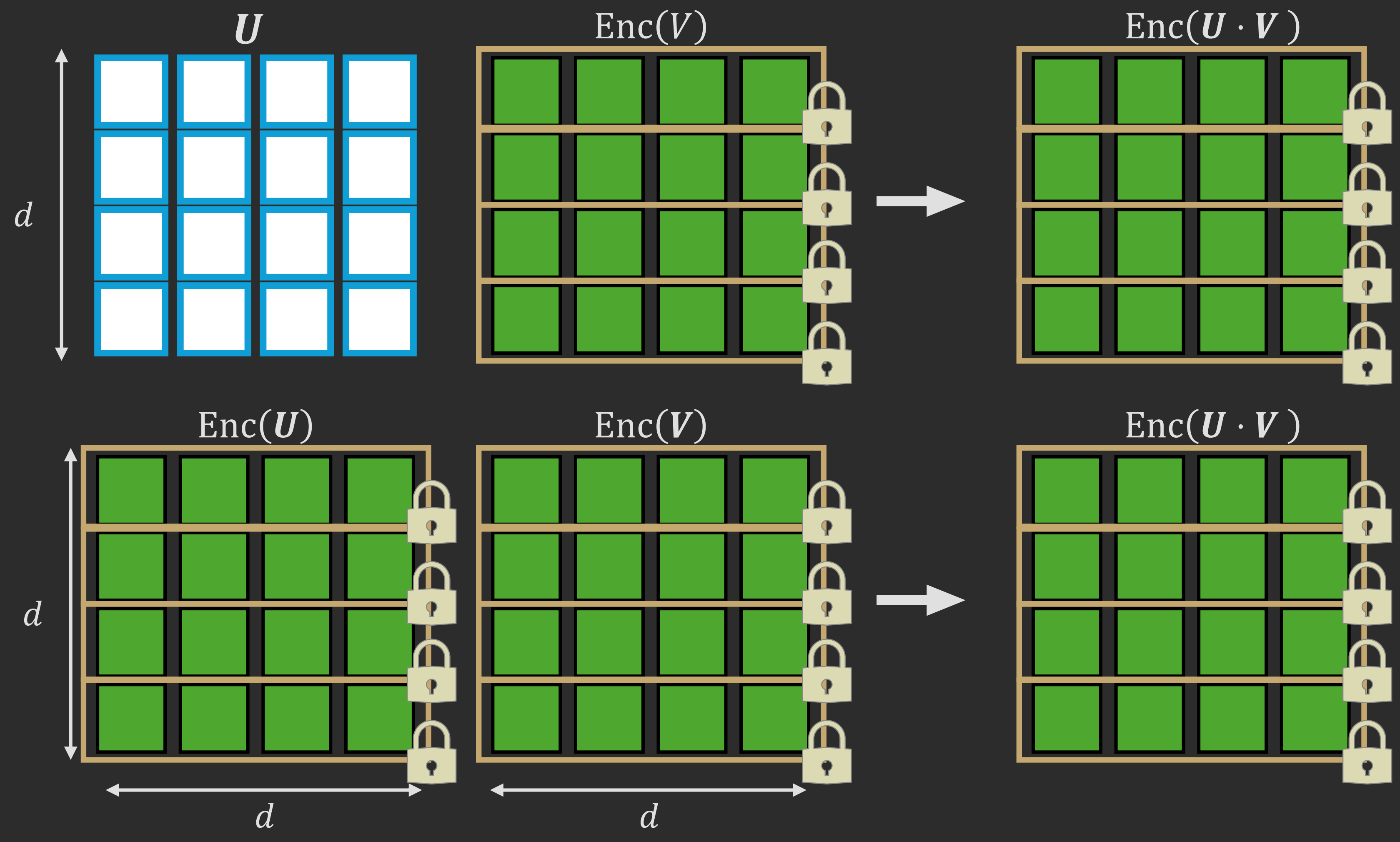
CCMM (JKLS'18, estimated)

> 19 hours

PCMM and CCMM

PCMM

CCMM



PCMM/CCMM with CKKS

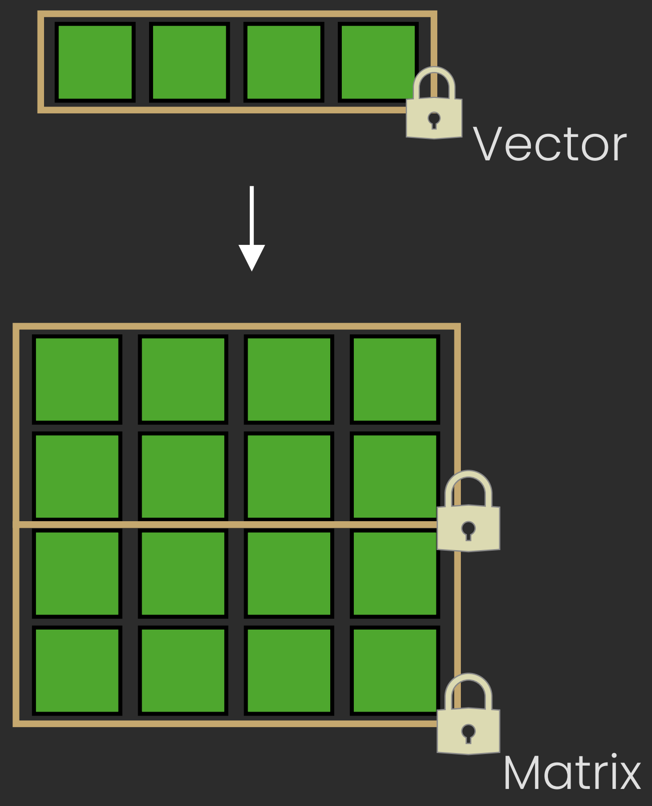
- CKKS
  - Plaintext: vector of real numbers
  - Native operations: // add, // mult, and rotate.

PCMM/CCMM with CKKS

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

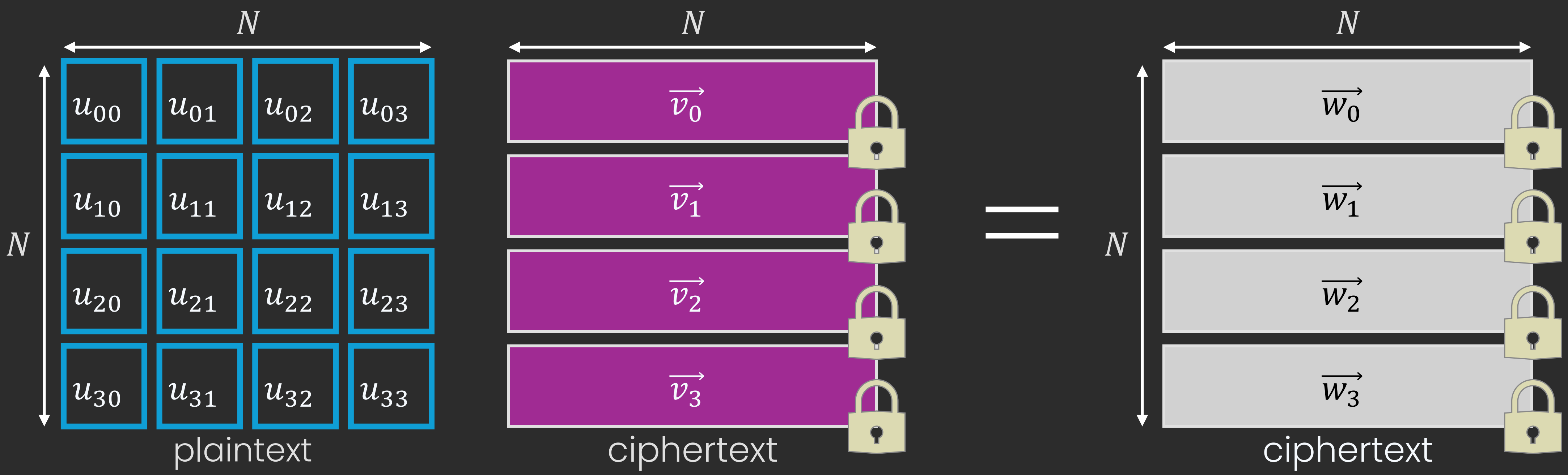
PCMM/CCMM with CKKS

- 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.



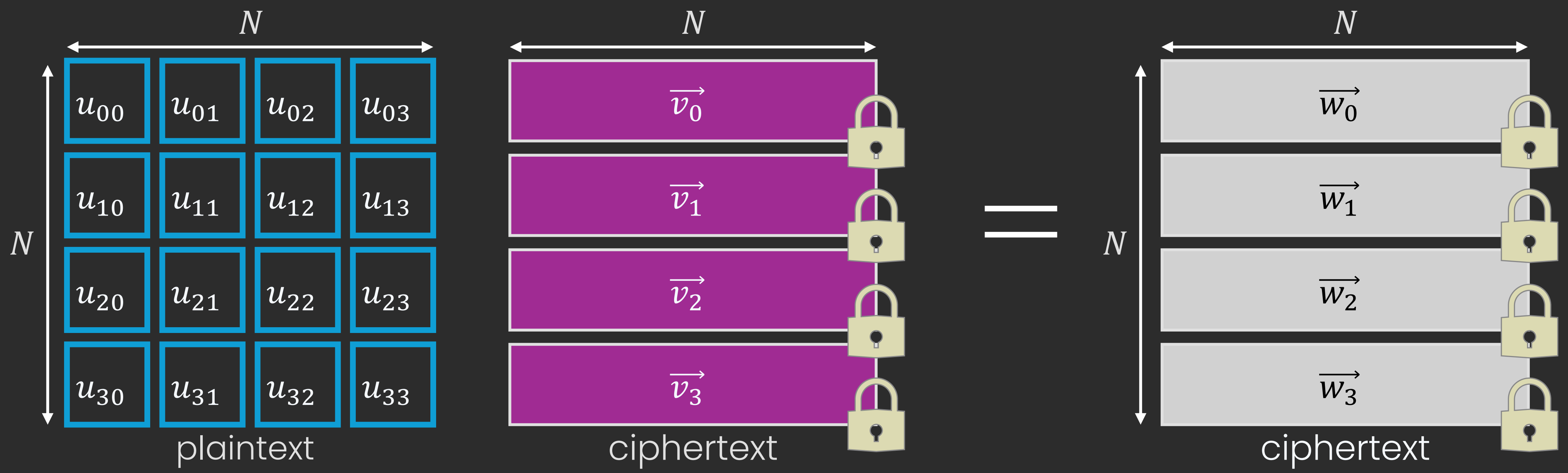
Why BLAS?

- PCMM when the matrix dimension $d = N$



Why BLAS?

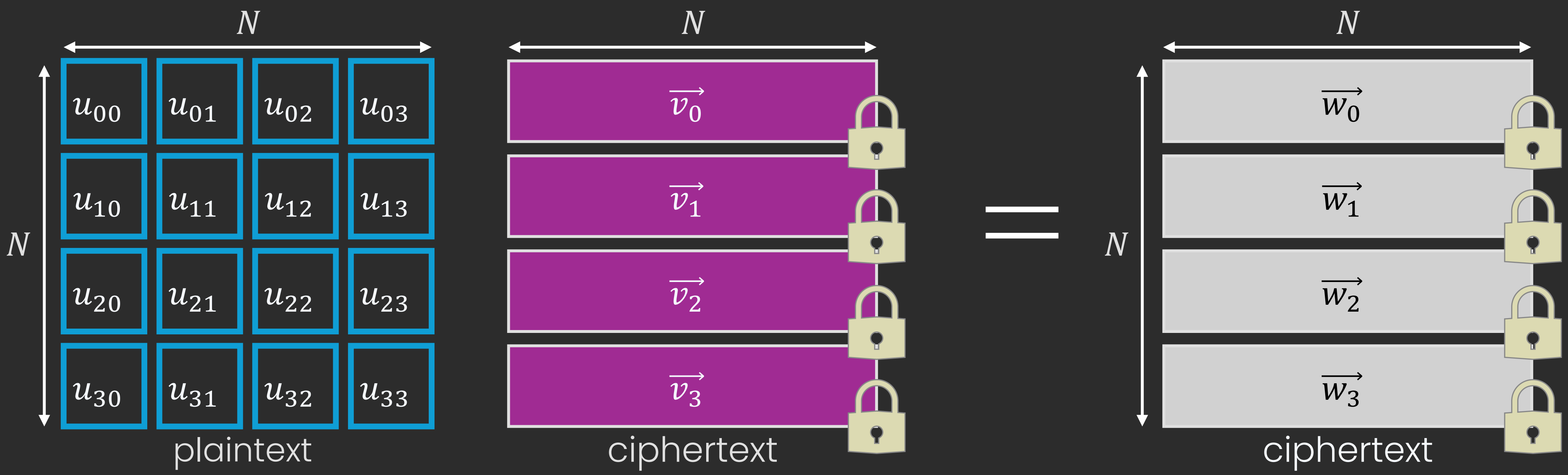
- PCMM when the matrix dimension $d = N$



- Linear algebra: $\vec{w_j} = u_{j0}\vec{v_0} + u_{j1}\vec{v_1} + u_{j2}\vec{v_2} + u_{j3}\vec{v_3} = \sum_i u_{ji}\vec{v_i}$
- Linear HE : $Enc(\vec{w_j}) = \sum_i u_{ji} Enc(\vec{v_i})$

Why BLAS?

- PCMM when the matrix dimension $d = N$



- Linear algebra: $\vec{w_j} = u_{j0}\vec{v_0} + u_{j1}\vec{v_1} + u_{j2}\vec{v_2} + u_{j3}\vec{v_3} = \sum_i u_{ji}\vec{v_i}$
- Linear HE : $Enc(\vec{w_j}) = \sum_i u_{ji} Enc(\vec{v_i})$

$>$ 2500 seconds for $N = 2^{13}$

How to utilize BLAS?

Q. How to utilize PPMM BLAS libraries?

How to utilize BLAS?

Q. How to utilize PPMM BLAS libraries?

A. Reduction from PCMM/CCMM to PPMM

How to utilize BLAS?

Q. How to utilize PPMM BLAS libraries?

A. Reduction from PCMM/CCMM to PPMM

RLWE ctxts of $\boldsymbol{U}, \boldsymbol{V}$

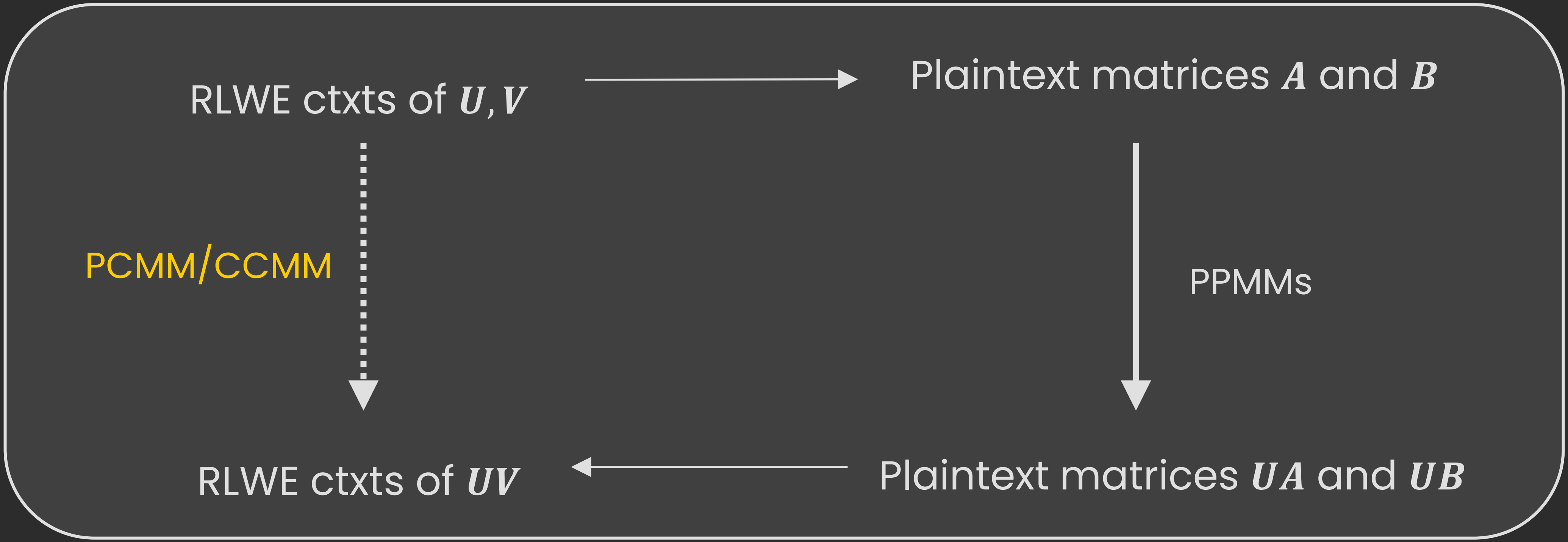
PCMM/CCMM ↓

RLWE ctxts of $\boldsymbol{UV}$

How to utilize BLAS?

Q. How to utilize PPMM BLAS libraries?

A. Reduction from PCMM/CCMM to PPMM



RLWE-based Encryption of Matrices

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

RLWE-based Encryption of Matrices

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

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

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

RLWE-based Encryption of Matrices

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

RLWE-based Encryption of Matrices

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

$$\begin{bmatrix} - & \boldsymbol{a}_1^t & - \\ - & \boldsymbol{a}_2^t & - \\ & \vdots & \\ - & \boldsymbol{a}_N^t & - \end{bmatrix} \begin{bmatrix} s_0 & s_1 & \cdots & s_{N-1} \\ -s_{N-1} & s_0 & \cdots & s_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ -s_1 & -s_2 & \cdots & s_0 \end{bmatrix} + \begin{bmatrix} - & \boldsymbol{b}_1^t & - \\ - & \boldsymbol{b}_2^t & - \\ & \vdots & \\ - & \boldsymbol{b}_N^t & - \end{bmatrix} = \begin{bmatrix} - & \boldsymbol{m}_1^t & - \\ - & \boldsymbol{m}_2^t & - \\ & \vdots & \\ - & \boldsymbol{m}_N^t & - \end{bmatrix}$$

RLWE-based Encryption of Matrices

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

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

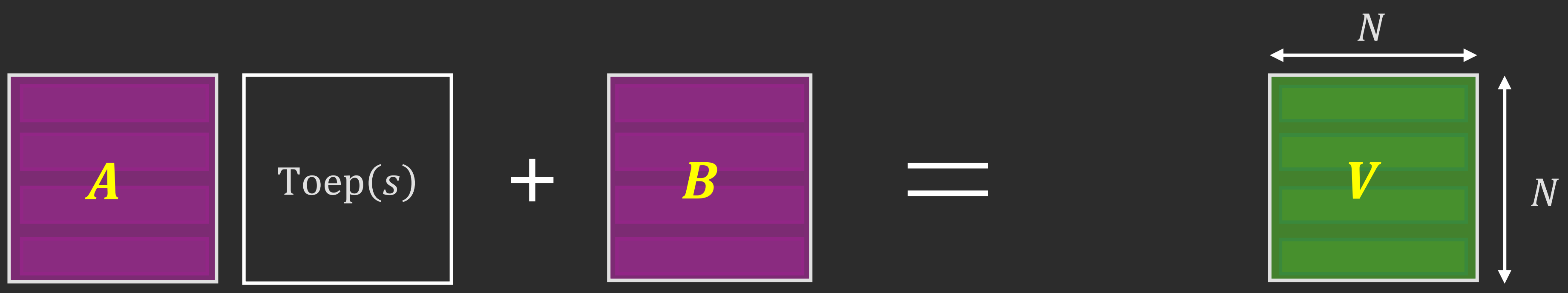
$\mathrm{Toep}(s)$

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

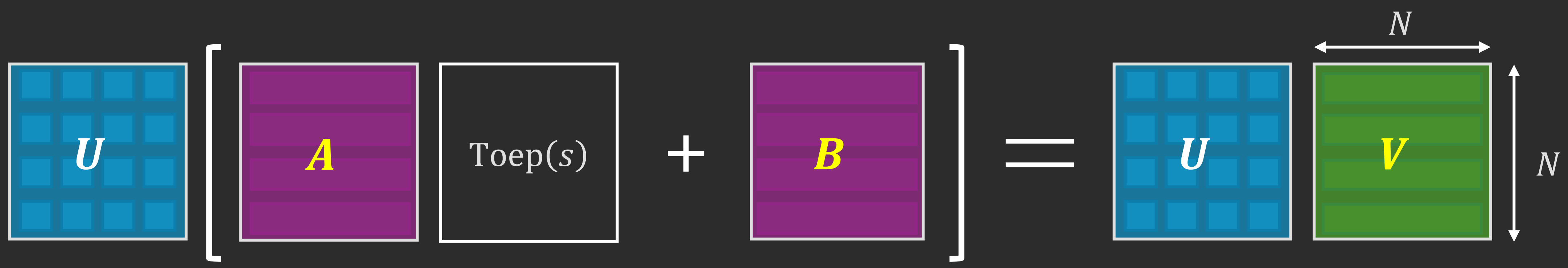
- N RLWE ciphertexts are:

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

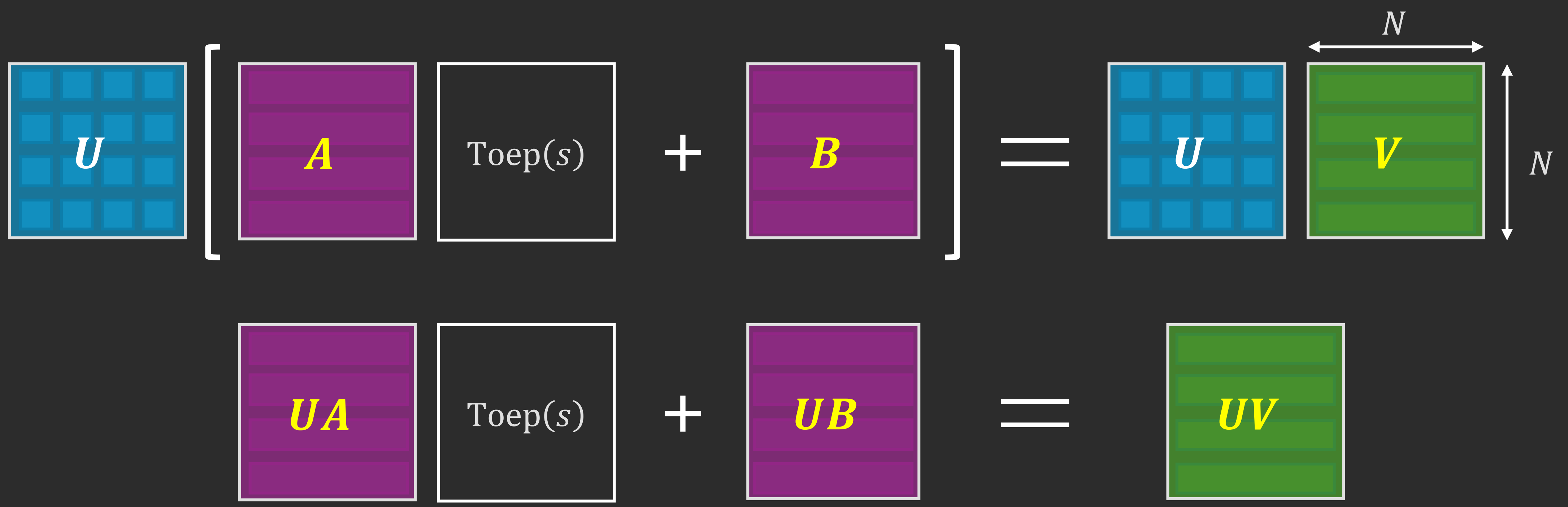
PCMM ≤ PPMMs (BCHPS'24, LZ'22)



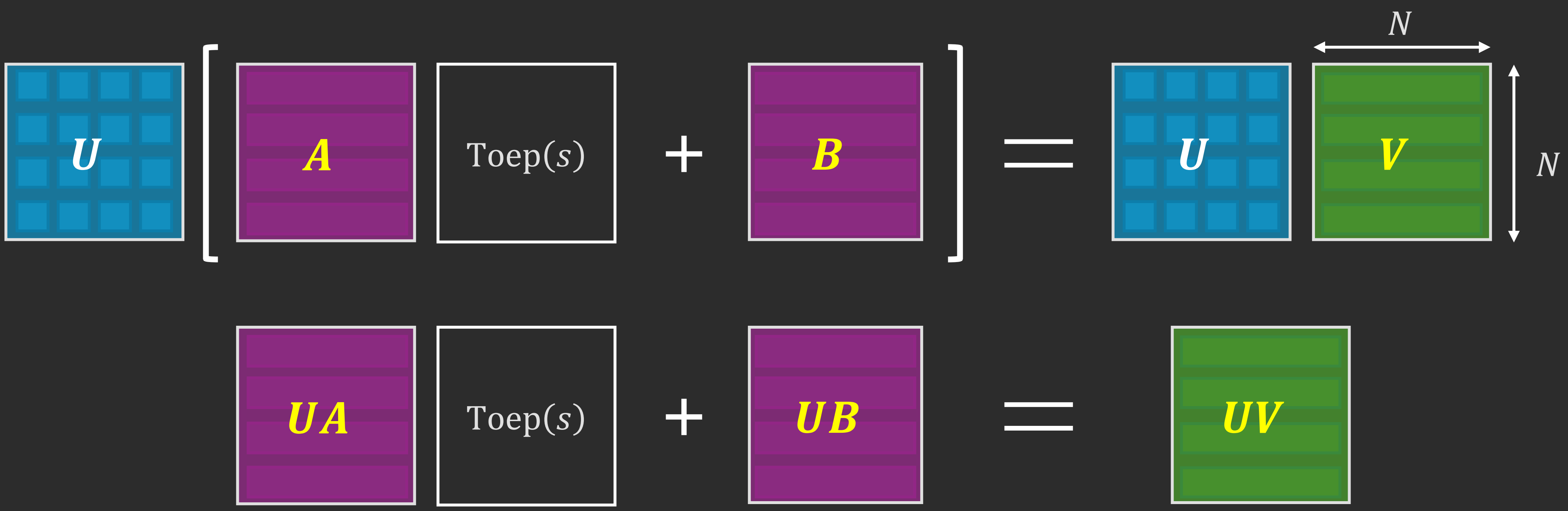
PCMM ≤ PPMMs (BCHPS'24, LZ'22)



PCMM $\leq$ PPMMs (BCHPS'24, LZ'22)

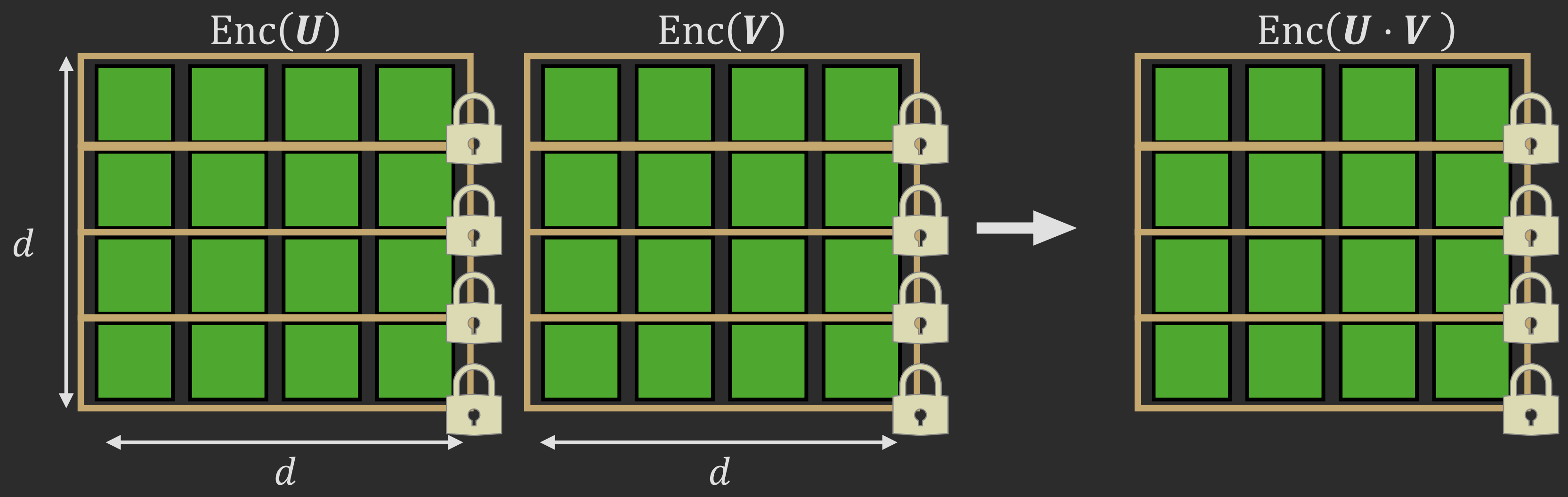


PCMM ≤ PPMMs (BCHPS'24, LZ'22)

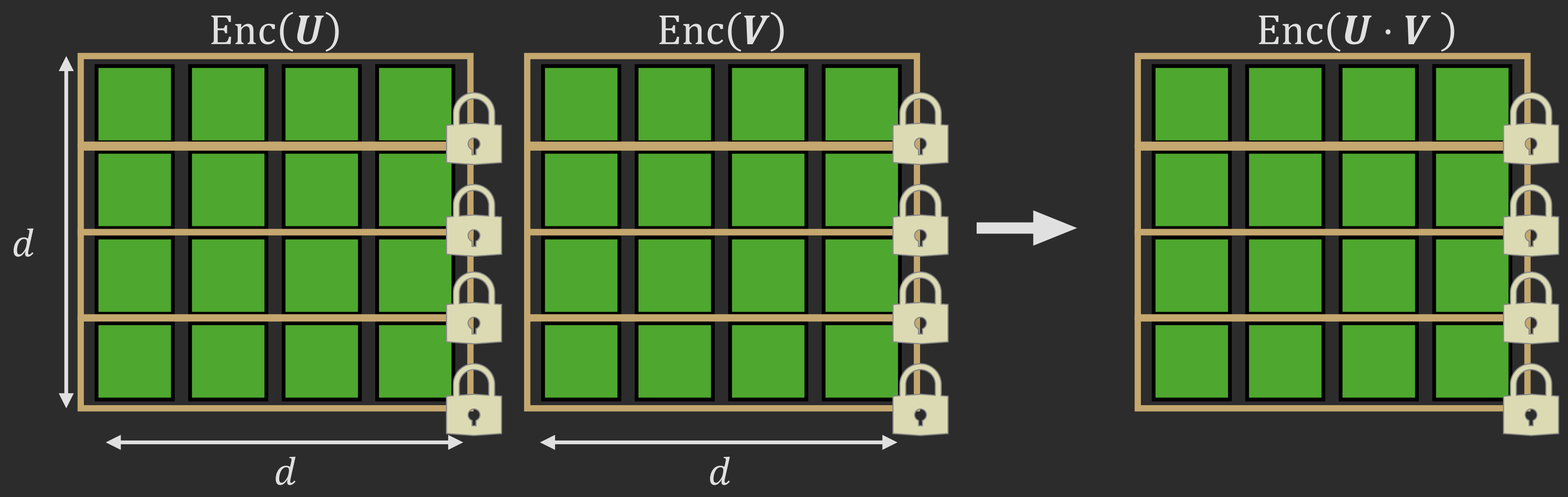


- $N \times N \times N$ PCMM ≤ *two* $N \times N \times N$ PPMMs *modulo* Q
- We use fast PPMM BLAS libraries for $N \times N \times N$ PCMM
- For PCMMs with other dimensions, see BCHPS'24

CCMM for Large Matrices



CCMM for Large Matrices



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

CCMM?

$$\boldsymbol{U}\,\boldsymbol{V} = \left[\boldsymbol{A}\,\mathrm{Toep}(s) + \boldsymbol{B}\right]\left[\boldsymbol{A'}\,\mathrm{Toep}(s) + \boldsymbol{B'}\right]$$

CCMM?

$$UV = \left[A\,\mathrm{Toep}(s) + B\right]\left[A'\,\mathrm{Toep}(s) + B'\right]$$

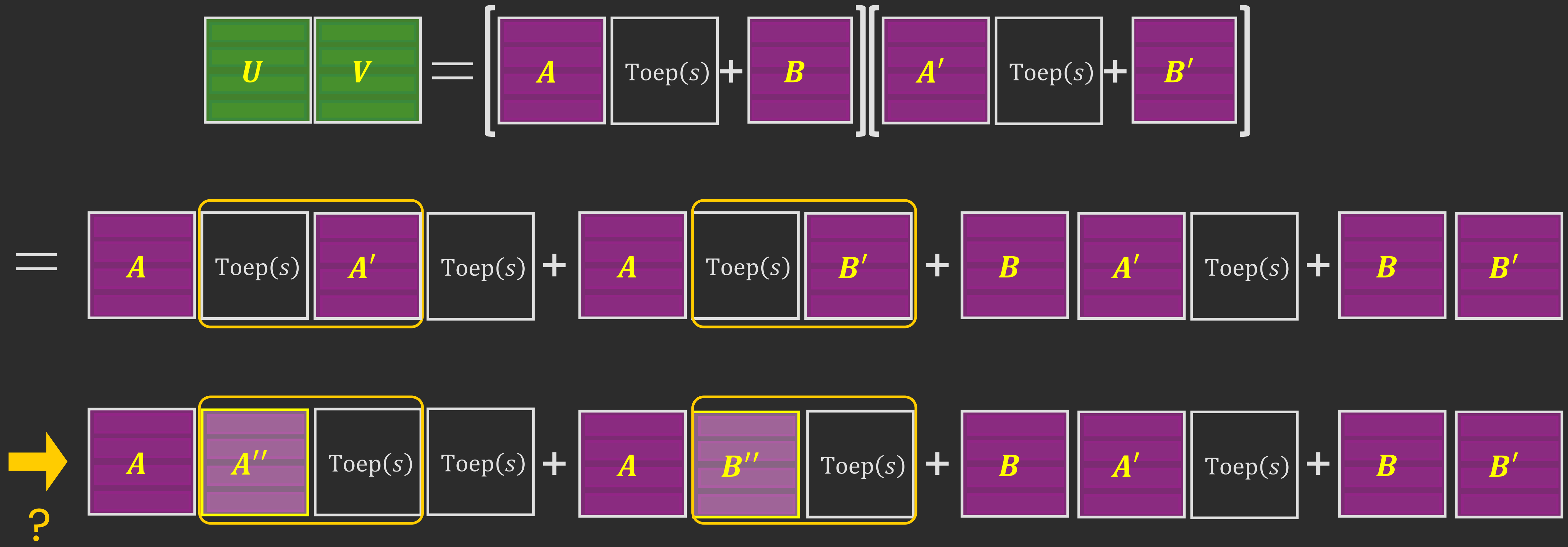
$$= A\,\mathrm{Toep}(s)\,A'\,\mathrm{Toep}(s) + A\,\mathrm{Toep}(s)\,B' + B\,A'\,\mathrm{Toep}(s) + B\,B'$$

CCMM?

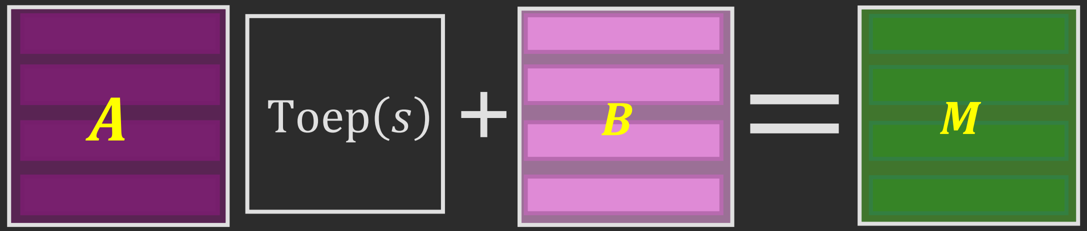
$$UV = \left[A\,\mathrm{Toep}(s) + B\right]\left[A'\,\mathrm{Toep}(s) + B'\right]$$

$$= A\,\mathrm{Toep}(s)\,A'\,\mathrm{Toep}(s) + A\,\mathrm{Toep}(s)\,B' + B\,A'\,\mathrm{Toep}(s) + B\,B'$$

CCMM?

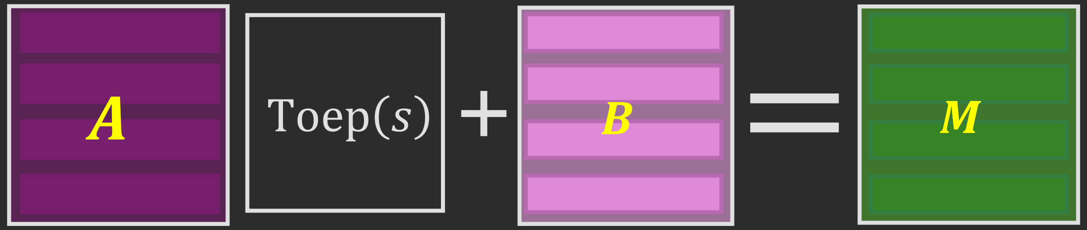

U
V
=
A
Toep(s)
+
B
A′
Toep(s)
+
B′
=
A
Toep(s)
A′
Toep(s)
+
A
Toep(s)
B′
+
B
A′
Toep(s)
+
B
B′
A
A′′
Toep(s)
Toep(s)
+
A
B′′
Toep(s)
+
B
A′
Toep(s)
+
B
B′
?

A.Toep(s) vs. Toep(s).A

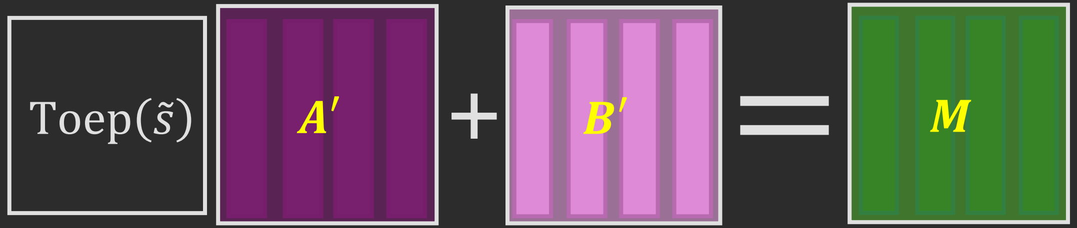


Encrypting each row: $a_i s + b_i = m_i$

A.Toep(s) vs. Toep(s).A

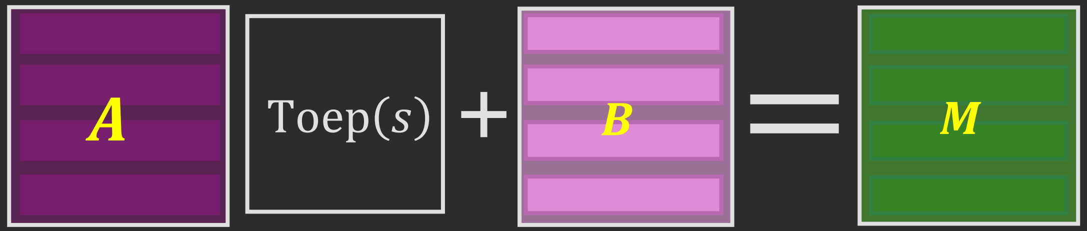


Encrypting each row: $a_i s + b_i = m_i$

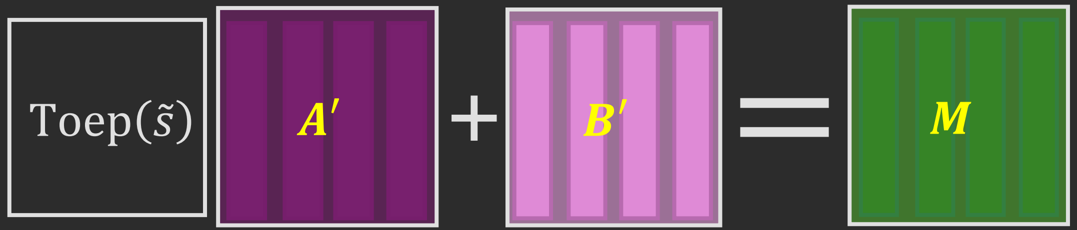


Encrypting each column: $a_j s + b_j = m'_j$

A.Toep(s) vs. Toep(s).A



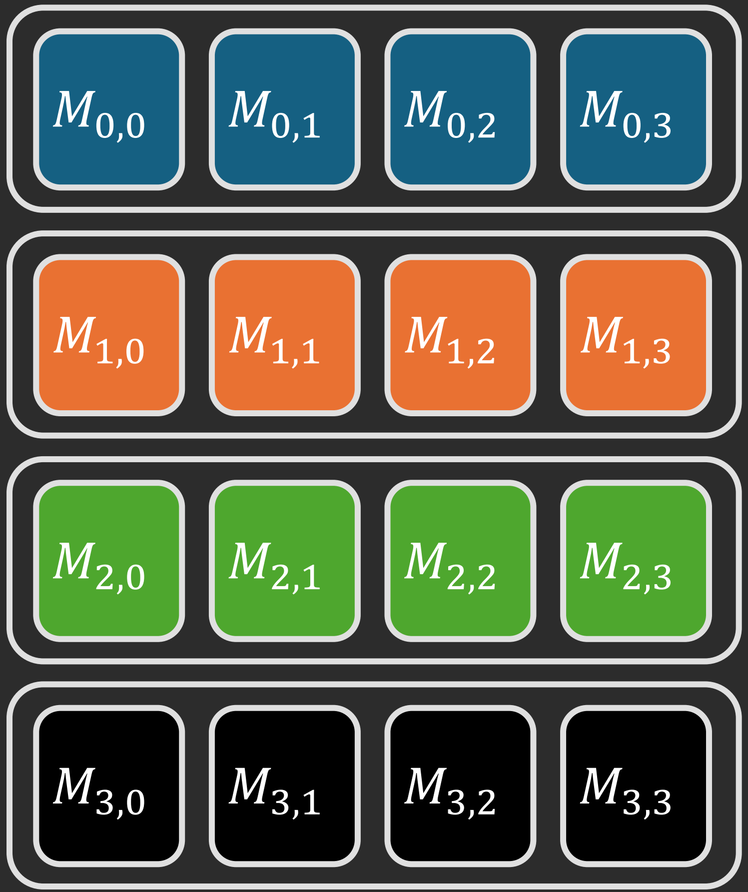
Encrypting each row: $a_i s + b_i = m_i$



Encrypting each column: $a_j s + b_j = m_j'$

✓ N RLWE ciphertexts to encrypt $N \times N$ matrix $\boldsymbol{M}$
- Row: $\boldsymbol{A} \cdot \mathrm{Toep}(s) + \boldsymbol{B} = \boldsymbol{M}$
- Column: $\mathrm{Toep}(\tilde{s}) \cdot \boldsymbol{A}' + \boldsymbol{B}' = \boldsymbol{M}$

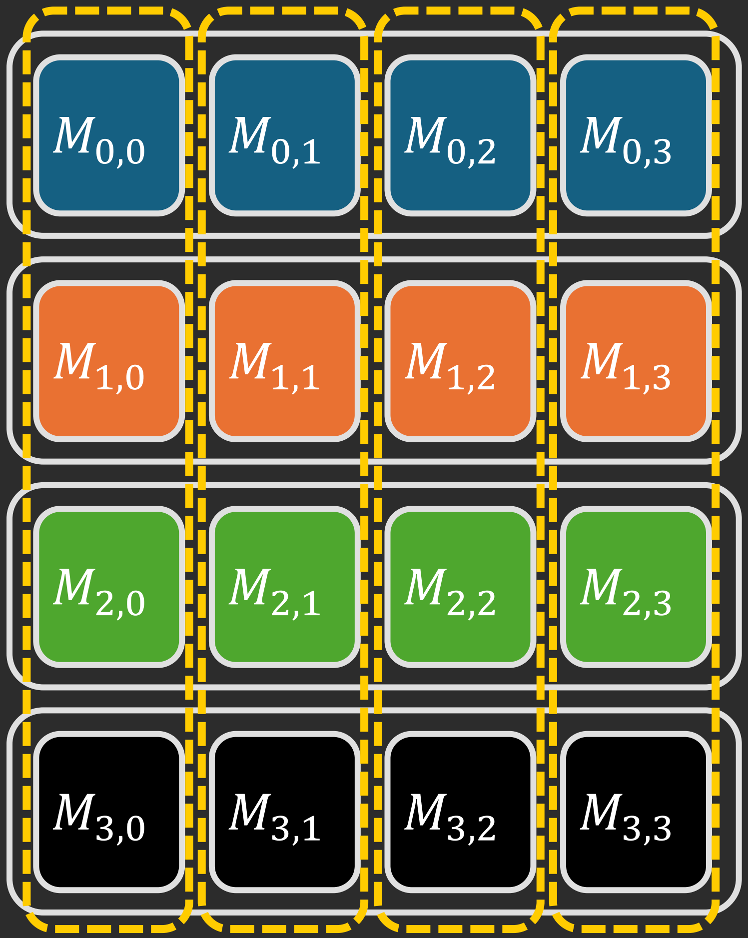
Ciphertext Matrix Transpose (CMT)



N ciphertexts

CMT: $$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

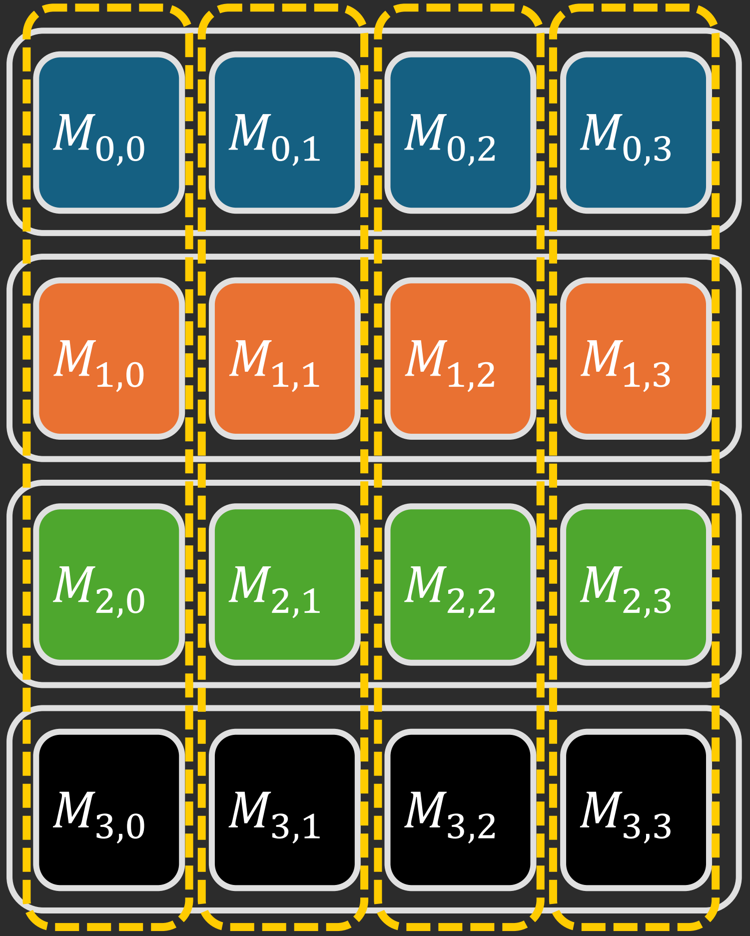
Ciphertext Matrix Transpose (CMT)



N ciphertexts → N ciphertexts

CMT: $m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$ → $m'_j(X) = \sum_{i \in [N]} M_{i,j} X^i$

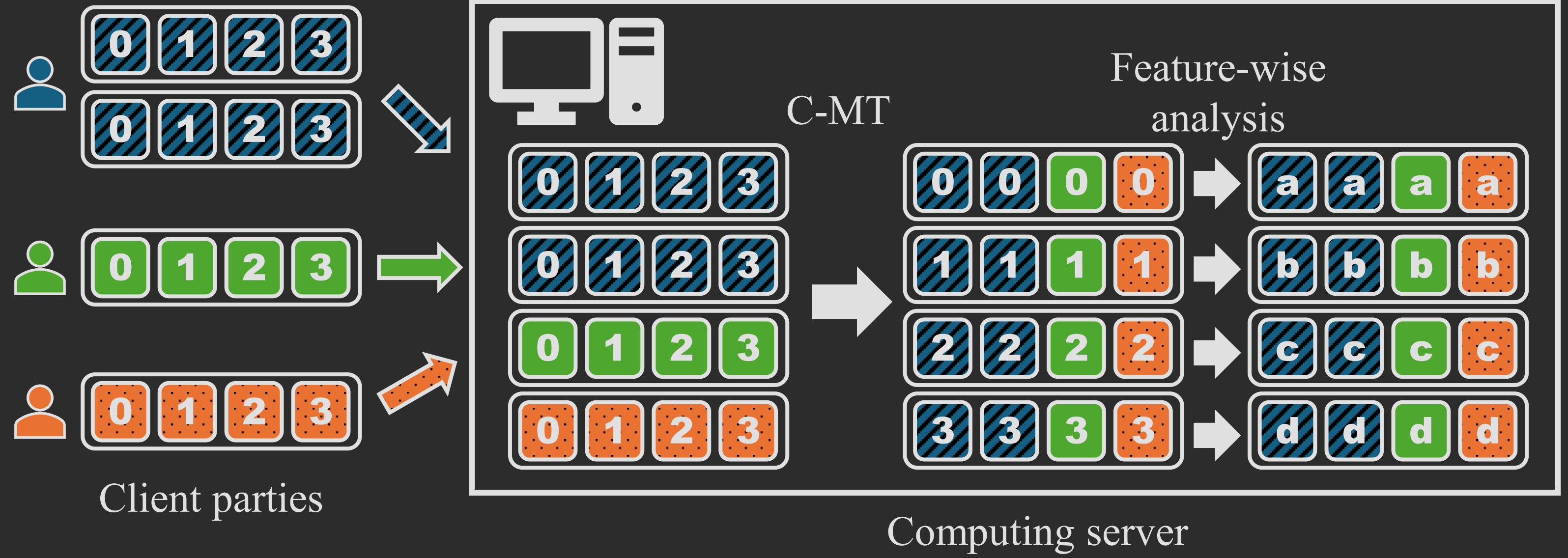
Ciphertext Matrix Transpose (CMT)



N ciphertexts → N ciphertexts

CMT: $m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$ → $m'_j(X) = \sum_{i \in [N]} M_{i,j} X^i$

Application



CMT with N keyswitchings

N ciphertexts

$$m_i(X) = \sum_{j\in[N]} M_{i,j}\, X^j$$

CMT

$$m'_j(X) = \sum_{i\in[N]} M_{i,j}\, X^i$$

N ciphertexts

CMT with N keyswitchings

Trace $\forall i, j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$

N ciphertexts

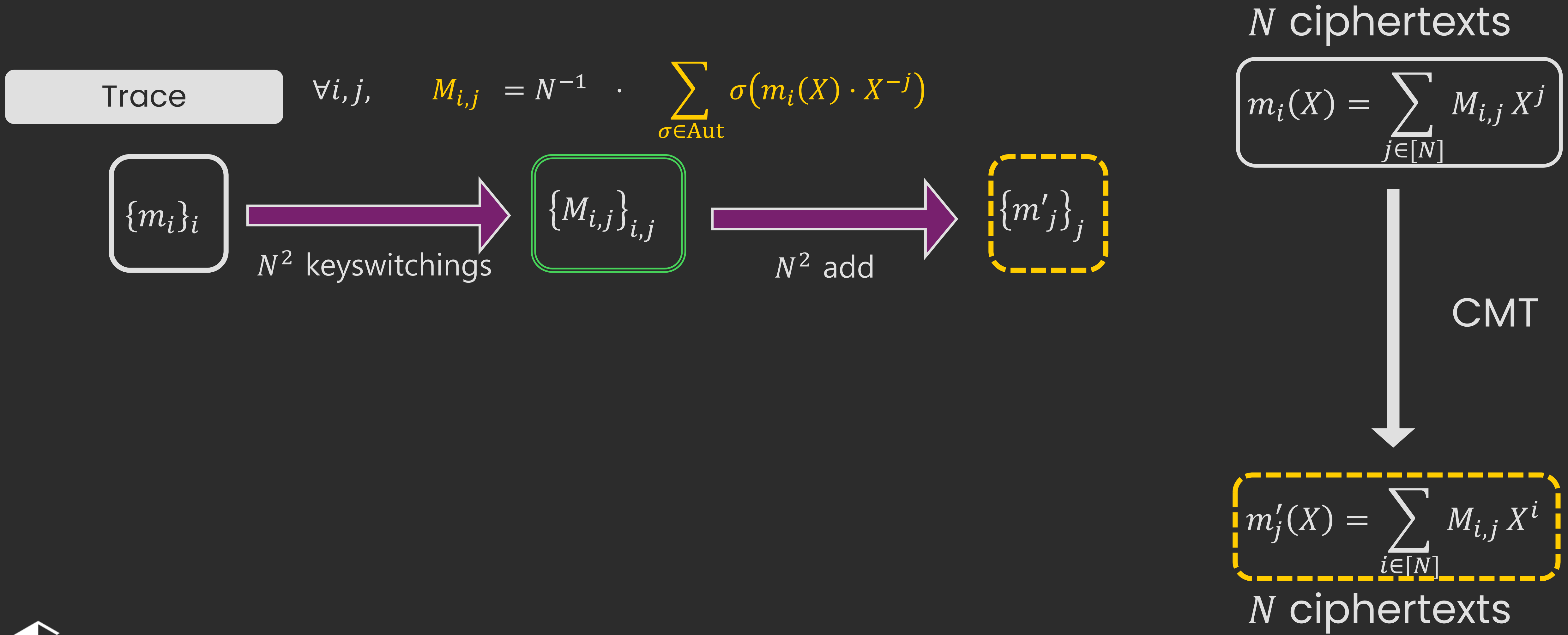
$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

CMT

$$m_j'(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

CMT with N keyswitchings



CMT with N keyswitchings

Trace

$$\forall i,j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$$

$$\forall j, \quad m'_j(X) = N^{-1} \cdot \sum_{i \in [N]} \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j} \cdot m_i(X)\right) \cdot X^i$$

N ciphertexts

$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

CMT

$$m'_j(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

CMT with N keyswitchings

Trace

$$\forall i,j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$$

$$\forall j, \quad m_j'(X) = N^{-1} \cdot \sum_{i \in [N]} \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j} \cdot m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sum_{i \in [N]} \sigma\left(X^{-j}\right) \cdot \sigma\left(m_i(X)\right) \cdot X^i$$

N ciphertexts

$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

↓ CMT

$$m_j'(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

CMT with N keyswitchings

Trace

$$\forall i,j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$$

$$\forall j, \quad m'_j(X) = N^{-1} \cdot \sum_{i \in [N]} \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j} \cdot m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sum_{i \in [N]} \sigma\left(X^{-j}\right) \cdot \sigma\left(m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j}\right) \cdot \sigma\left(\sum_{i \in [N]} m_i(X) \cdot \sigma^{-1}\left(X^i\right)\right)$$

N ciphertexts

$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

CMT

$$m'_j(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

CMT with N keyswitchings

Trace

$$\forall i,j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$$

Sharing automorphisms

$$\forall j, \quad m_j'(X) = N^{-1} \cdot \sum_{i \in [N]} \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j} \cdot m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sum_{i \in [N]} \sigma\left(X^{-j}\right) \cdot \sigma\left(m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j}\right) \cdot \sigma\left(\sum_{i \in [N]} m_i(X) \cdot \sigma^{-1}\left(X^i\right)\right)$$

$\widetilde{m_\sigma}(X)$

N ciphertexts

$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

CMT

$$m_j'(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

CMT with N keyswitchings

Trace

$$\forall i,j, \quad M_{i,j} = N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(m_i(X) \cdot X^{-j}\right)$$

Sharing automorphisms

$$\forall j, \quad m'_j(X) = N^{-1} \cdot \sum_{i \in [N]} \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j} \cdot m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sum_{i \in [N]} \sigma\left(X^{-j}\right) \cdot \sigma\left(m_i(X)\right) \cdot X^i$$

$$= N^{-1} \cdot \sum_{\sigma \in \mathrm{Aut}} \sigma\left(X^{-j}\right) \cdot \sigma\left(\sum_{i \in [N]} m_i(X) \cdot \sigma^{-1}\left(X^i\right)\right)$$

$\widetilde{m_\sigma}(X)$

$\{m_i\}_i$ → N^2 add → $\{\widetilde{m_\sigma}\}_\sigma$ → N key-switchings → $\{\sigma(\widetilde{m_\sigma})\}_\sigma$ → N^2 add → $\{m'_j\}_j$

N ciphertexts

$$m_i(X) = \sum_{j \in [N]} M_{i,j} X^j$$

↓ CMT

$$m'_j(X) = \sum_{i \in [N]} M_{i,j} X^i$$

N ciphertexts

Tweak Algorithm

- ✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}\left(X^i\right) \cdot m_i$
- ✓ $m_j'(X) = \sum_\sigma \sigma\left(X^{-j}\right) \cdot \sigma(\widetilde{m_\sigma})$

Tweak Algorithm

- ✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$
- ✓ $m_j'(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\mathrm{Tweak}\left(\{m_i\}_{i\in[n]}\right) \mapsto \left\{\sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i\right\}_{j\in[n]}$$

N is the ring degree of RLWE

Tweak Algorithm

✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$
✓ $m_j'(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\text{Tweak}(\{m_i\}_{i\in[n]}) \mapsto \left\{\sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i\right\}_{j\in[n]}$$

N is the ring degree of RLWE

- $\text{Tweak}(\{m_i\}_{i\in[n]})$ can be done with
  - $\text{Tweak}(\{m_{2i}\}_{i\in[n/2]})$
  - $\text{Tweak}(\{m_{2i+1}\}_{i\in[n/2]})$
  - n ring additions

Tweak Algorithm

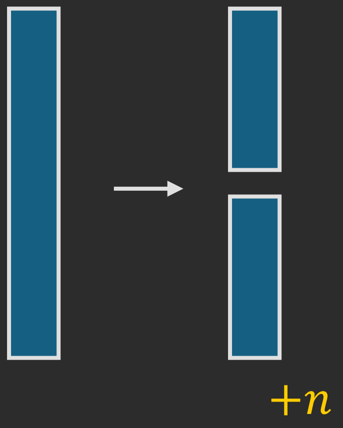
✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$

✓ $m_j'(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\mathrm{Tweak}(\{m_i\}_{i\in[n]}) \mapsto \left\{\sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i\right\}_{j\in[n]}$$

N is the ring degree of RLWE

- $\mathrm{Tweak}(\{m_i\}_{i\in[n]})$ can be done with
  - $\mathrm{Tweak}(\{m_{2i}\}_{i\in[n/2]})$
  - $\mathrm{Tweak}(\{m_{2i+1}\}_{i\in[n/2]})$
  - n ring additions



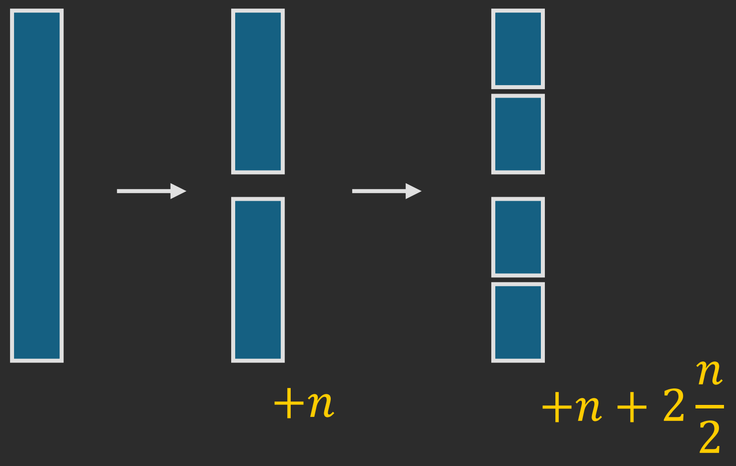
Tweak Algorithm

✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$
✓ $m'_j(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\mathrm{Tweak}(\{m_i\}_{i\in[n]}) \mapsto \left\{ \sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i \right\}_{j\in[n]}$$

N is the ring degree of RLWE

- $\mathrm{Tweak}(\{m_i\}_{i\in[n]})$ can be done with
  - $\mathrm{Tweak}(\{m_{2i}\}_{i\in[n/2]})$
  - $\mathrm{Tweak}(\{m_{2i+1}\}_{i\in[n/2]})$
  - n ring additions



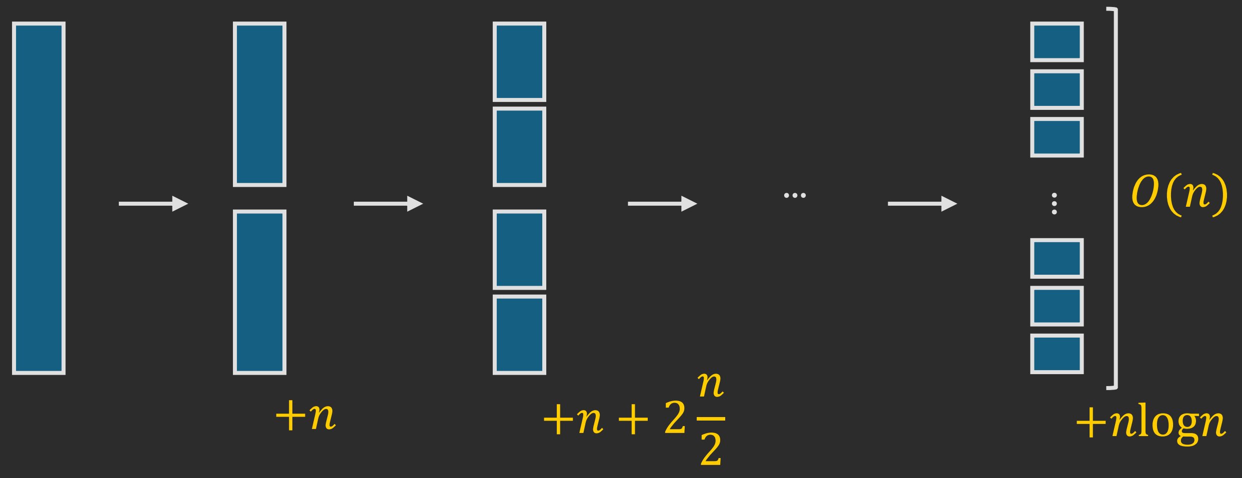
Tweak Algorithm

✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$
✓ $m'_j(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\mathrm{Tweak}(\{m_i\}_{i\in[n]}) \mapsto \left\{ \sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i \right\}_{j\in[n]}$$

N is the ring degree of RLWE

- $\mathrm{Tweak}(\{m_i\}_{i\in[n]})$ can be done with
  - $\mathrm{Tweak}(\{m_{2i}\}_{i\in[n/2]})$
  - $\mathrm{Tweak}(\{m_{2i+1}\}_{i\in[n/2]})$
  - n ring additions



Tweak Algorithm

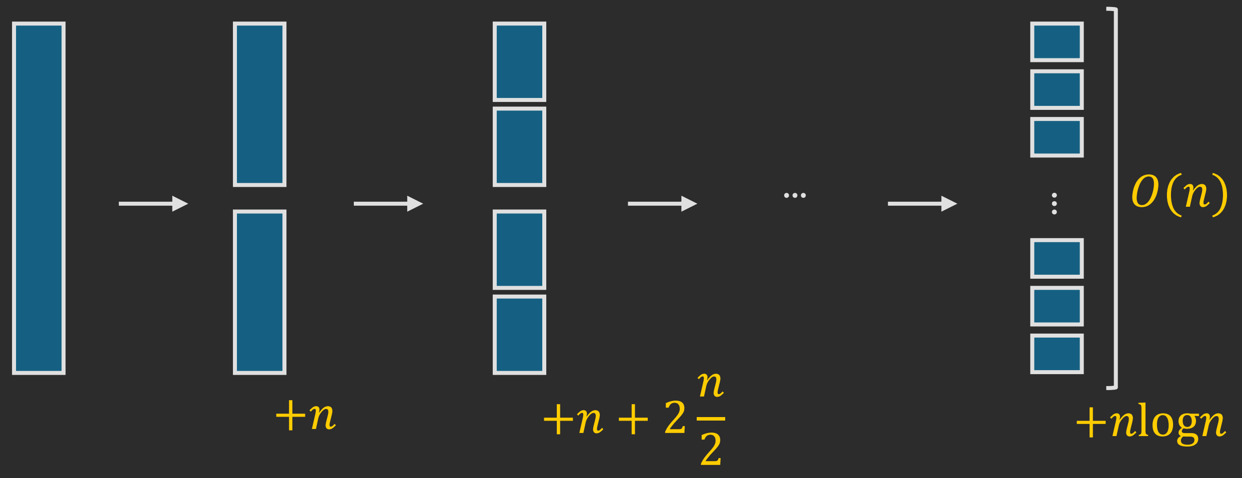
✓ $\widetilde{m_\sigma}(X) = \sum_i \sigma^{-1}(X^i) \cdot m_i$

✓ $m'_j(X) = \sum_\sigma \sigma(X^{-j}) \cdot \sigma(\widetilde{m_\sigma})$

$$\mathrm{Tweak}(\{m_i\}_{i\in[n]}) \mapsto \left\{ \sum_{i\in[n]} X^{\frac{2N}{n}ij} \cdot m_i \right\}_{j\in[n]}$$

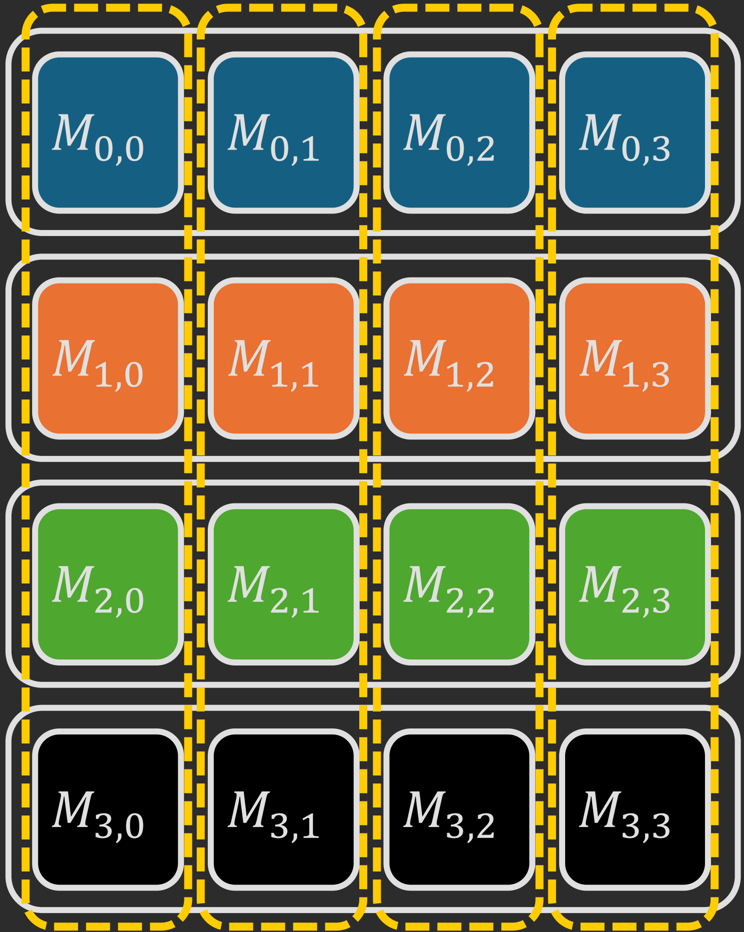
N is the ring degree of RLWE

- $\mathrm{Tweak}(\{m_i\}_{i\in[n]})$ can be done with
  - $\mathrm{Tweak}(\{m_{2i}\}_{i\in[n/2]})$
  - $\mathrm{Tweak}(\{m_{2i+1}\}_{i\in[n/2]})$
  - n ring additions



❖ The cost of $\mathrm{Tweak}(\{m_i\}_{i\in[n]})$ is $Nn\log n$

CMT Algorithm

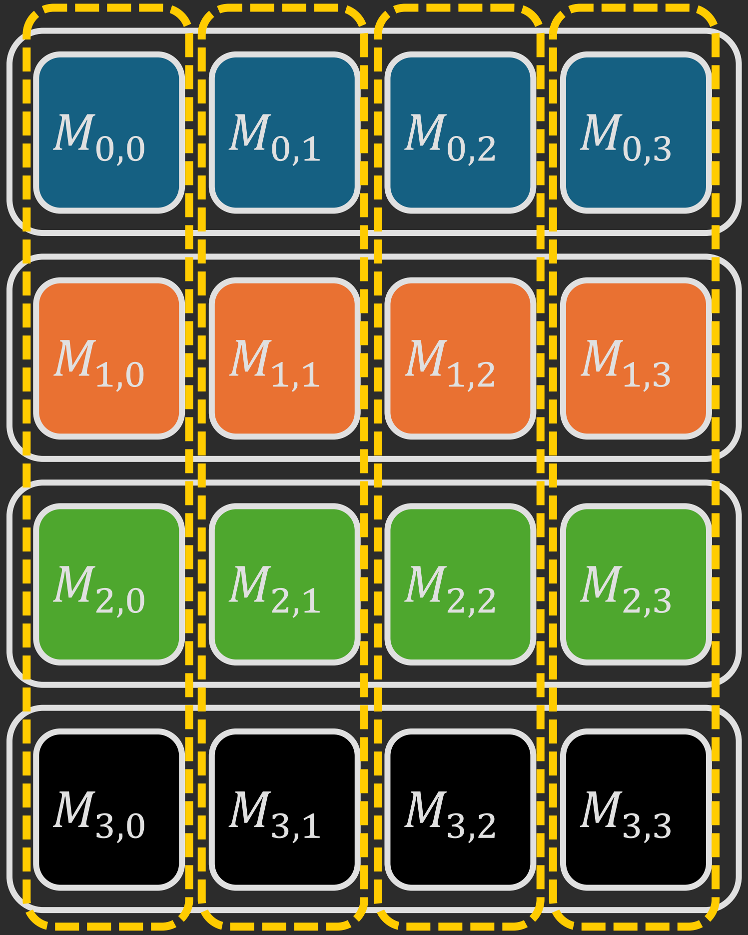


$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

CMT

$$m'_j(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

CMT Algorithm



$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

$\{m_i\}_i$

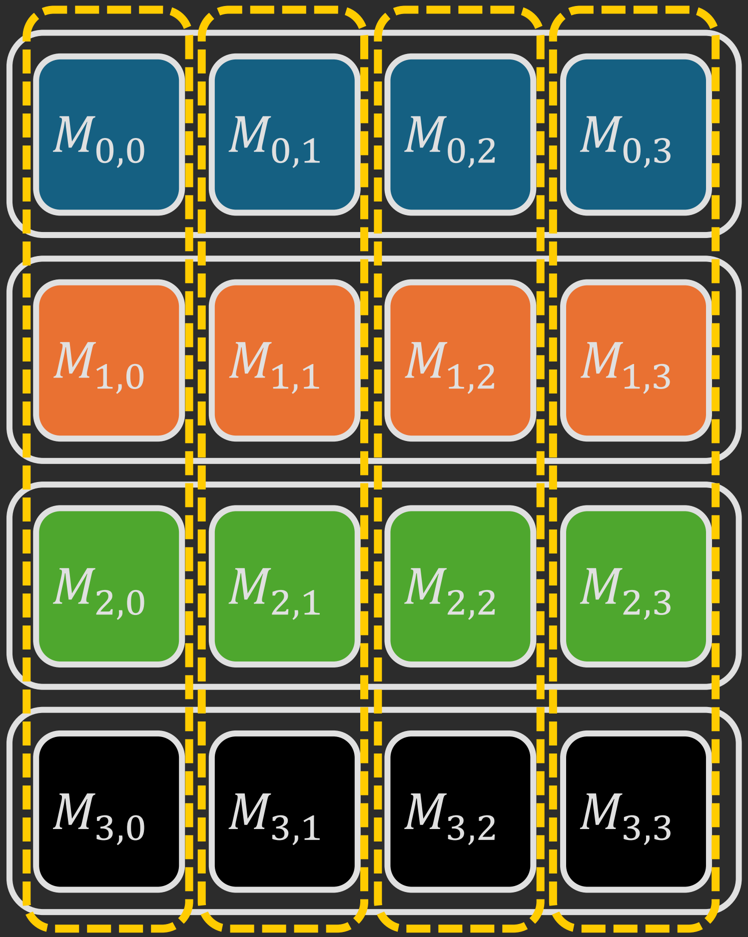
Tweak

$\{\widetilde{m_\sigma}\}_\sigma$

CMT

$$m'_j(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

CMT Algorithm



$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

CMT

$$\{m_i\}_i$$

↓ Tweak

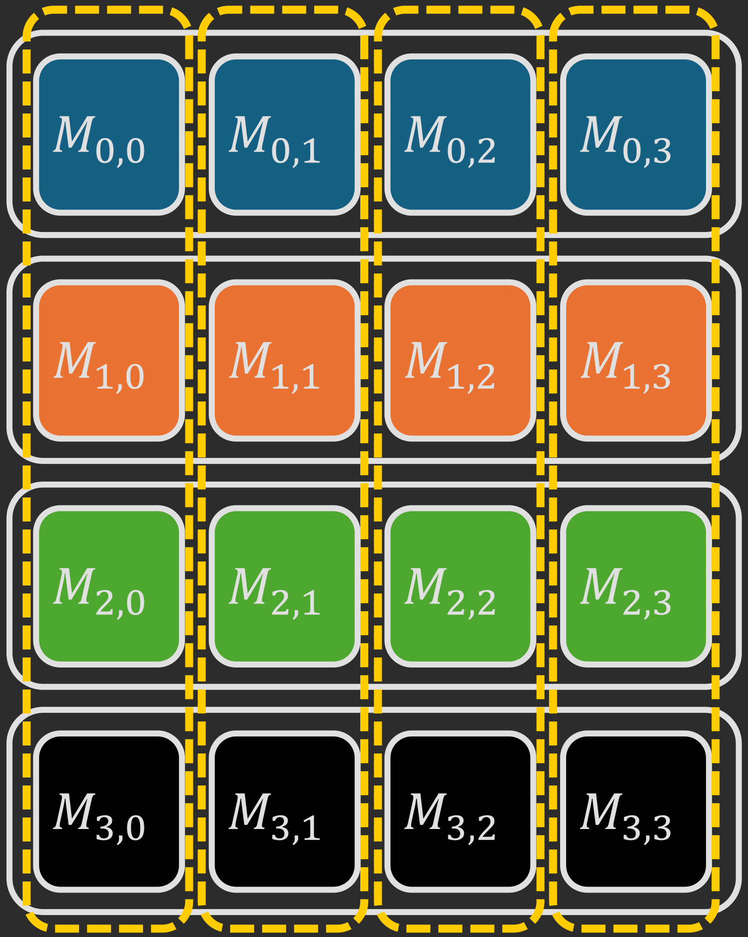
$$\{\widetilde{m_\sigma}\}_\sigma$$

↓ N Keyswitchings

$$\{\sigma(\widetilde{m_\sigma})\}_\sigma$$

$$m_j'(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

CMT Algorithm



$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

CMT:

$\{m_i\}_i$

↓ Tweak

$\{\widetilde{m_\sigma}\}_\sigma$

↓ N Keyswitchings

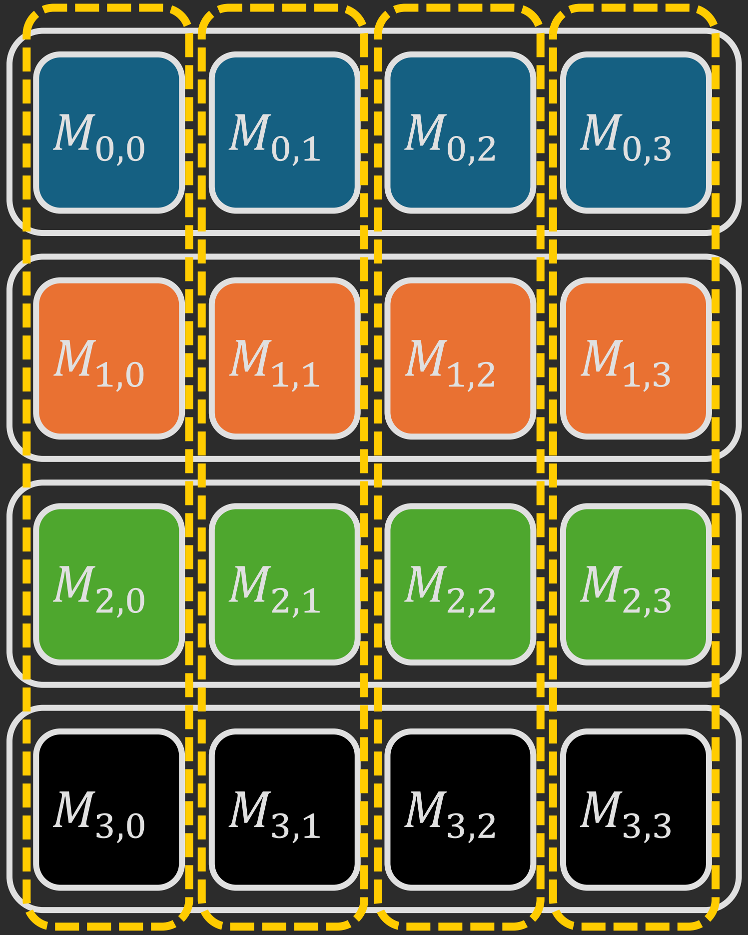
$\{\sigma(\widetilde{m_\sigma})\}_\sigma$

↓ Tweak

$\{m'_j\}_j$

$$m'_j(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

CMT Algorithm



$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

CMT:

$\{m_i\}_i$

↓ Tweak

$\{\widetilde{m_\sigma}\}_\sigma$

↓ N Keyswitchings

$\{\sigma(\widetilde{m_\sigma})\}_\sigma$

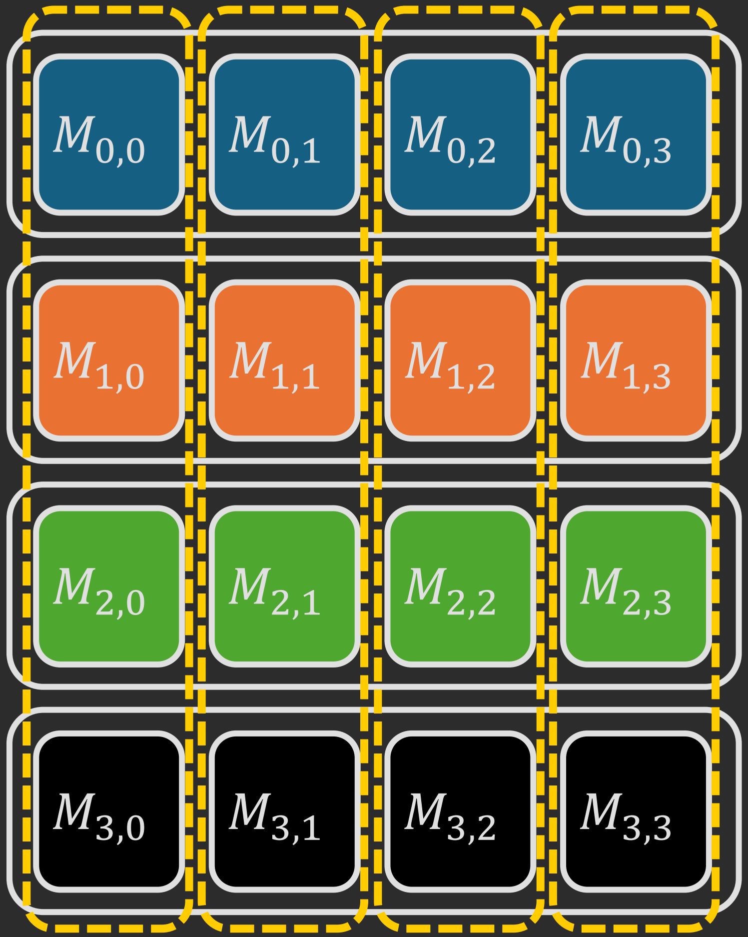
↓ Tweak

$\{m'_j\}_j$

$$m'_j(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

❖ $\tilde{O}(N^2)$ operations

CMT Algorithm



$$m_i(X) = m_{i,0} + m_{i,1}X + m_{i,2}X^2 + m_{i,3}X^3$$

CMT

$\{m_i\}_i$

↓ Tweak

$\{\widetilde{m_\sigma}\}_\sigma$

↓ N Keyswitchings

$\{\sigma(\widetilde{m_\sigma})\}_\sigma$

↓ Tweak

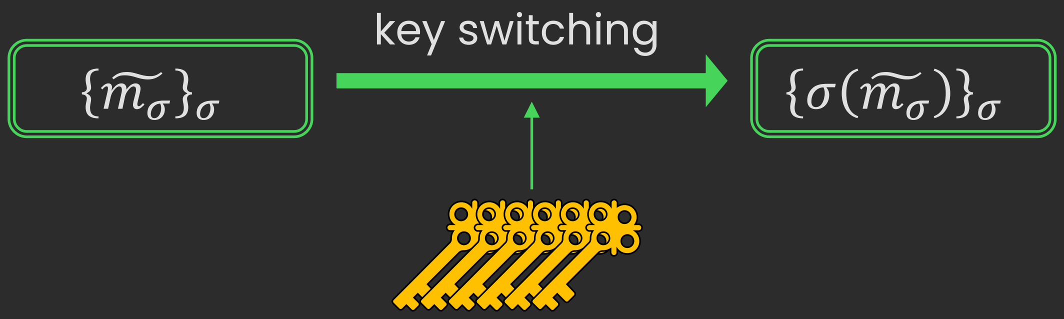
$\{m'_j\}_j$

$$m'_j(X) = m_{0,j} + m_{1,j}X + m_{2,j}X^2 + m_{3,j}X^3$$

❖ $\tilde{O}(N^2)$ operations

Lightweight CMT Algorithm

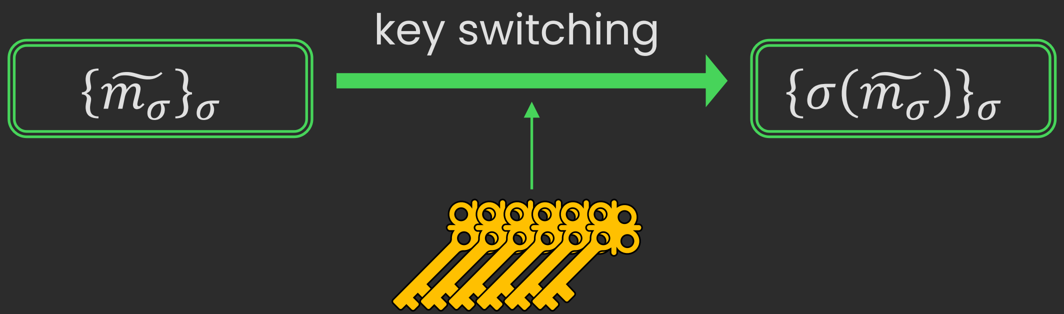
Basic algorithm



: N keys for all automorphisms

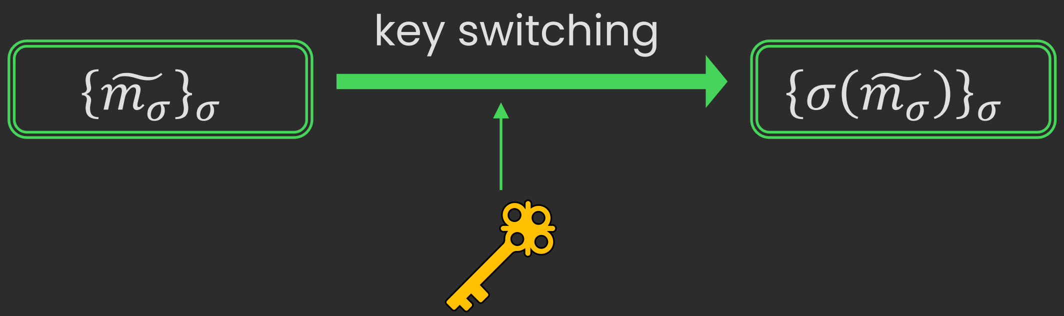
Lightweight CMT Algorithm

Basic algorithm



: N keys for all automorphisms

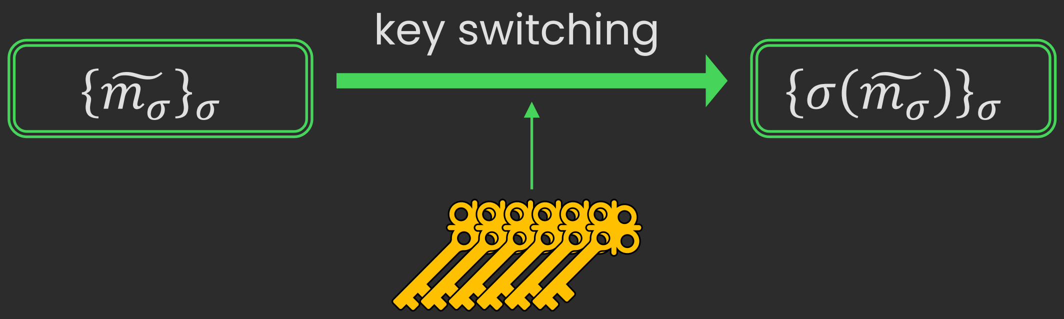
Lightweight algorithm



: a single key for automorphism

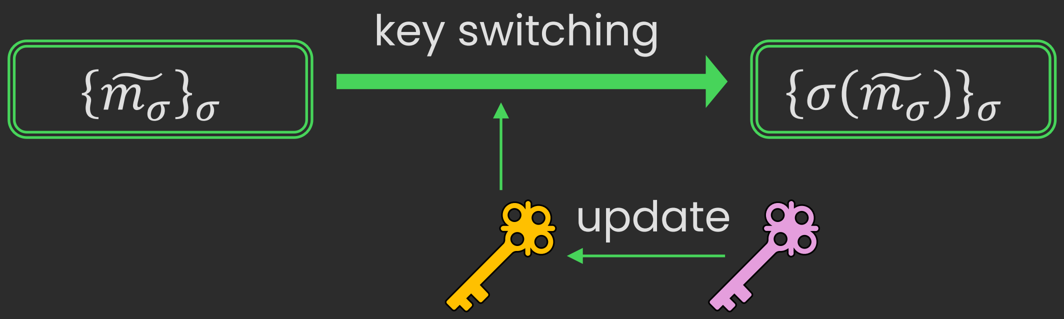
Lightweight CMT Algorithm

Basic algorithm



: N keys for all automorphisms

Lightweight algorithm

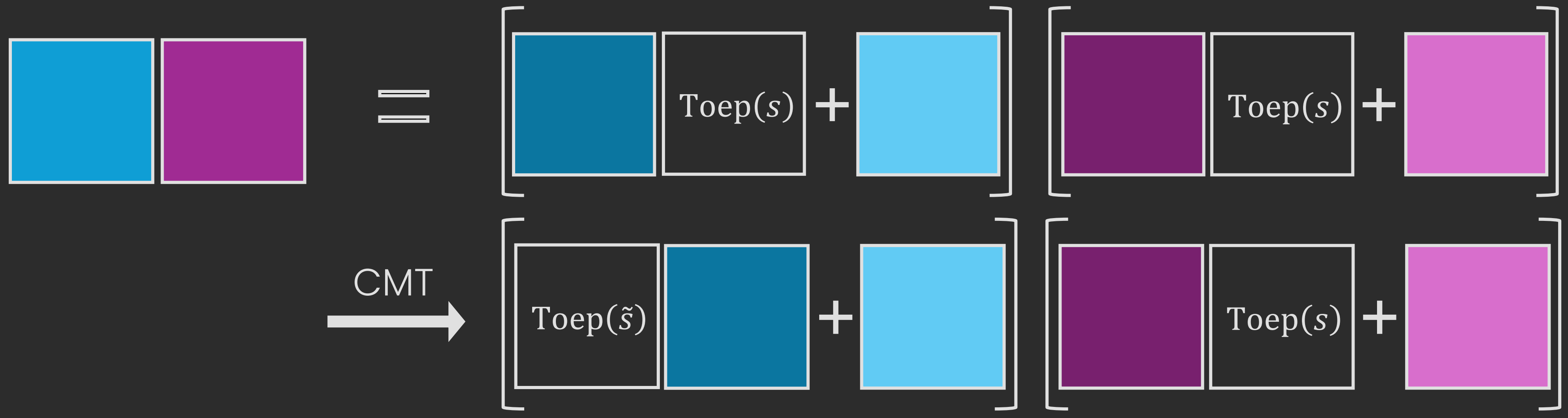


: a single key for automorphism

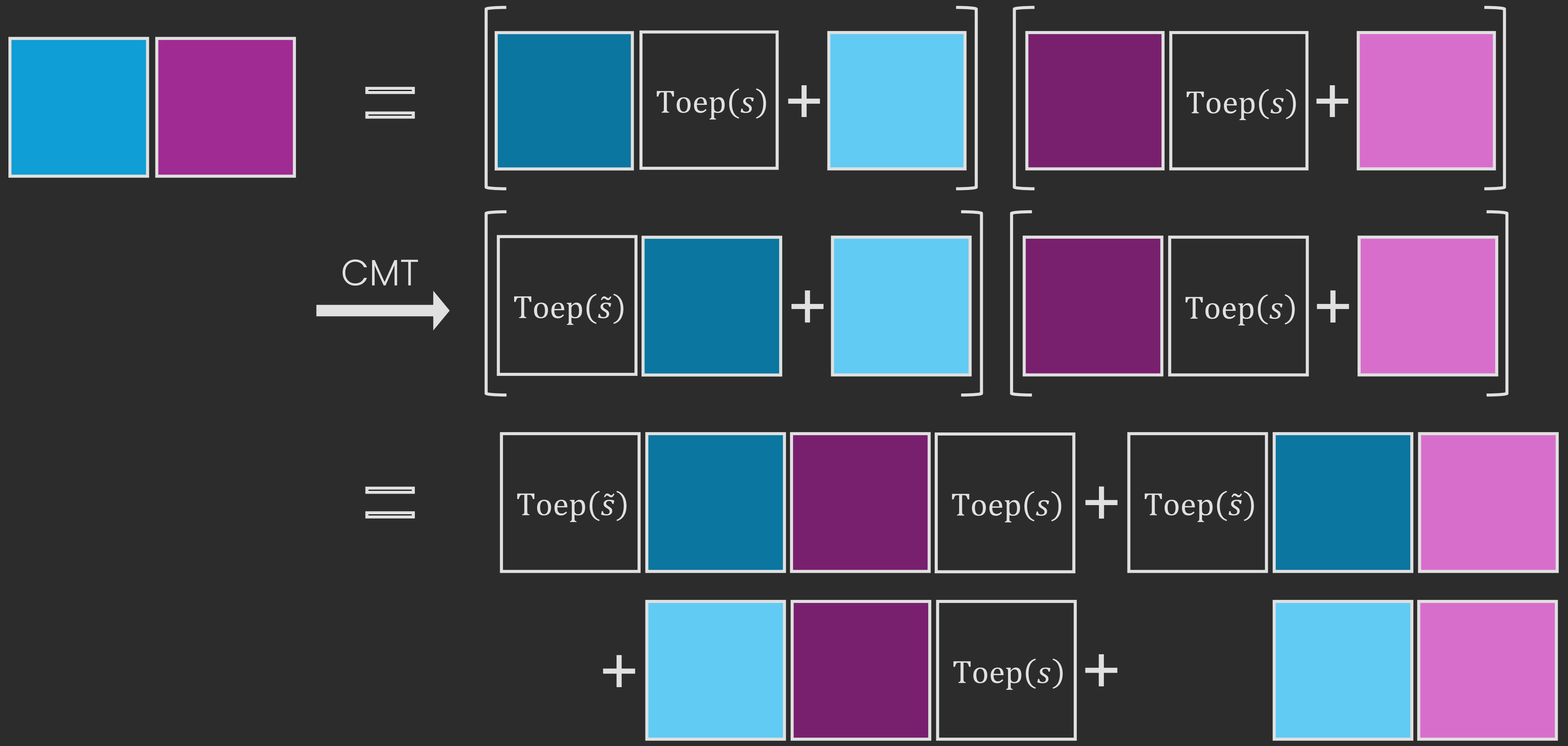
+ two master keys for the “key updates”

Reduction from CCMM to PPMM (1/2)

Reduction from CCMM to PPMM (1/2)

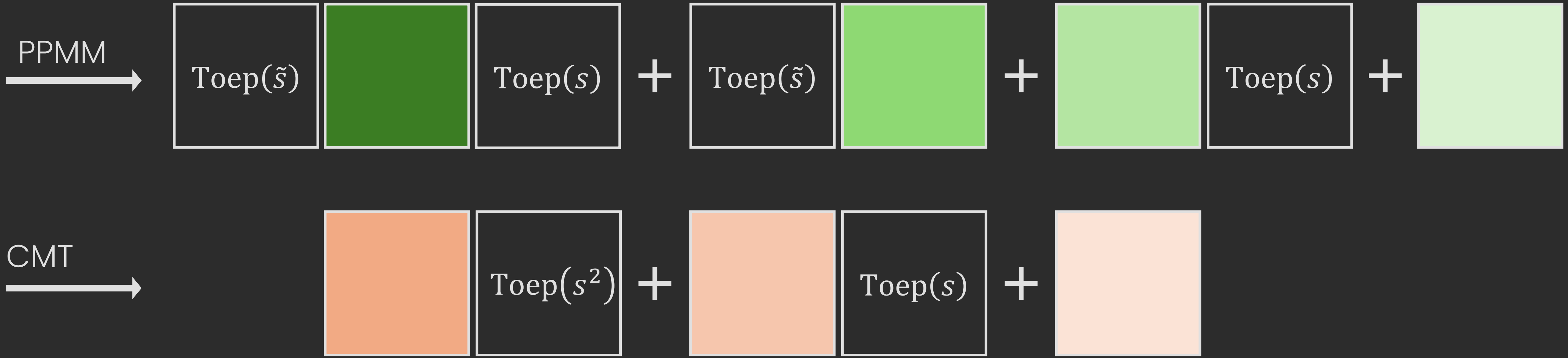


Reduction from CCMM to PPMM (1/2)

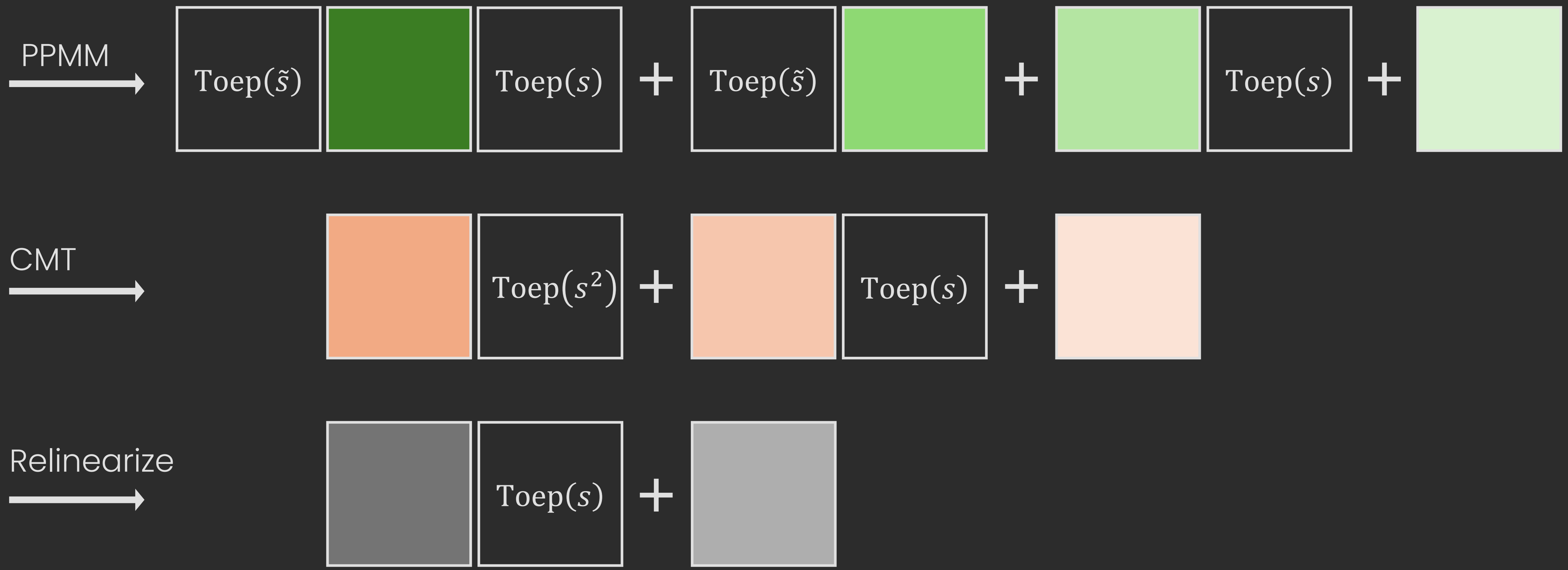


Reduction from CCMM to PPMM (2/2)

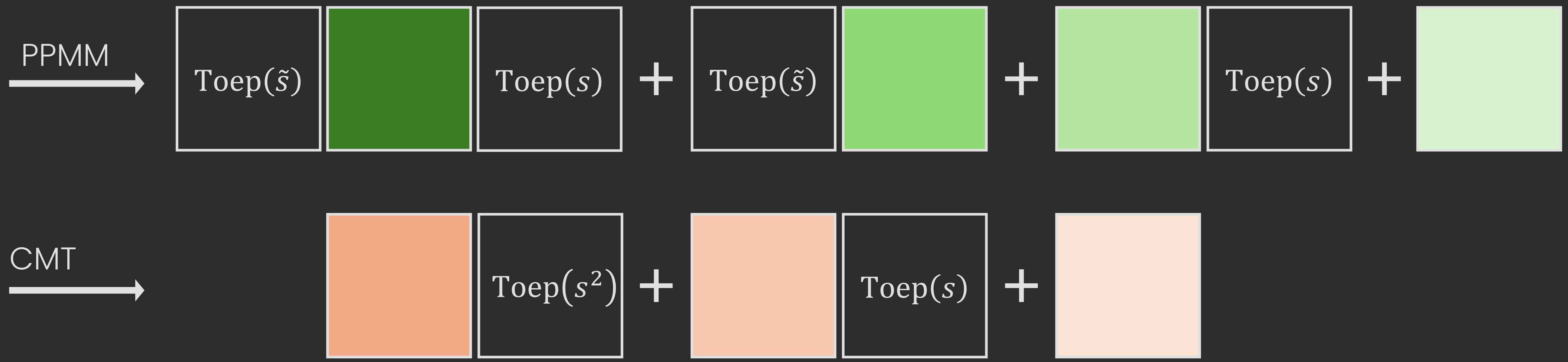
Reduction from CCMM to PPMM (2/2)



Reduction from CCMM to PPMM (2/2)



Reduction from CCMM to PPMM (2/2)



- $N \times N \times N$ CCMM by 4 PPMMs and 3 CMTs
- For PPMM, we use BLAS libraries.
- The others than PPMM are subcubic.

Experimental Results on CCMM

Algorithm	Matrix Dimension	$(\log N, \log Q)$	CMTs	PPMMs	Relin. & Resc.	Total (s)	Prec. (bit)	Key size (MB)
Basic	4096	$(12, 36 + 28)$	25.5	57.1	2.58	85.2	18.7	436
Basic	8192	$(13, 38 + 28)$	104	481	11.8	596	18.5	1960
Lightweight	8192	$(13, 38 + 28)$	186	474	11.8	672	18.5	1.57

All experiments are measured on Intel® Xeon® Gold 6242 CPU at 2.80GHz with a single-thread

All parameters are 128-bit secure

HEaaN library for HE, FLINT library (based on OpenBLAS) for PPMM

Experimental Results on CMT

Algorithm	Matrix Dimension	$(\log N, \log Q)$	Latency (s)	Prec. (bit)	Key size (MB)
Basic	2048	$(11, 26)$	0.764	10.7	27.3
Basic	4096	$(12, 28)$	3.04	16.3	134
Lightweight	4096	$(12, 28)$	4.92	14.2	0.246

All experiments are measured on Intel® Xeon® Gold 6242 CPU at 2.80GHz with a single-thread

All parameters are 128-bit secure

HEaaN library for HE

Follow-up Works

- BCHPS'25. "Encrypted Linear Algebra with BLAS"
  arxiv/2503.16080
  - CC-MM / PC-MM / CC-Mv / PC-Mv with preprocessing using GSW
  - Flexible dimensional CC-MM and PC-MM

- Gentry. "Reducing Encrypted Matrix Multiplication to Plaintext Matrix Multiplication"
  Presented at FHE.org conference 2025
  - C-MT using multi-variate polynomials
  - No published paper or experimental results available yet

Wrapping up!

- Fast CCMM
  - Leverage efficiency of BLAS libraries

Wrapping up!

- Fast CCMM
  - Leverage efficiency of BLAS libraries
- Fast CMT
  - Useful beyond being as a tool for CCMM

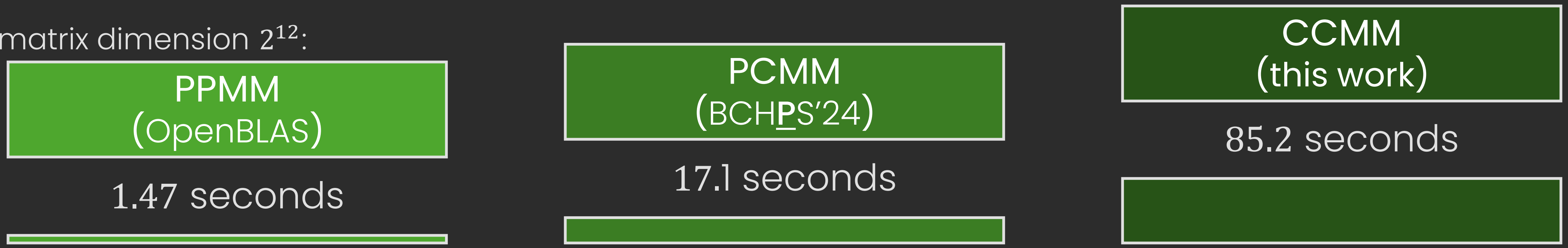
Wrapping up!

- Fast CCMM
  - Leverage efficiency of BLAS libraries
- Fast CMT
  - Useful beyond being as a tool for CCMM
- Lightweight algorithms
  - CCMM with keys less than 2 MB

Wrapping up!

- Fast CCMM
  - Leverage efficiency of BLAS libraries
- Fast CMT
  - Useful beyond being as a tool for CCMM
- Lightweight algorithms
  - CCMM with keys less than 2 MB

For matrix dimension 2^{12}:



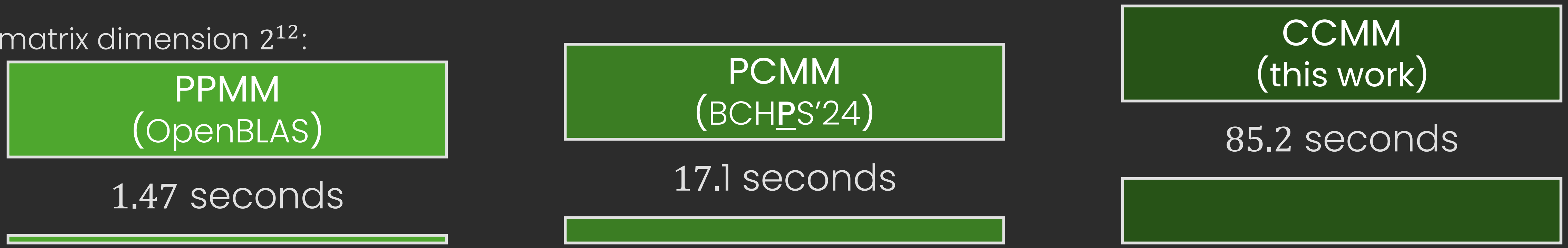
Wrapping up!

- Fast CCMM
  - Leverage efficiency of BLAS libraries
- Fast CMT
  - Useful beyond being as a tool for CCMM
- Lightweight algorithms
  - CCMM with keys less than 2 MB

eprint : 2025/448

Thank you!

For matrix dimension 2^{12}:



References

[Park'25] J. H. Park. "Ciphertext-Ciphertext Matrix Multiplication: Fast for Large Matrices." Eurocrypt 2025

[BCHPS'24] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, D. Stehlé. *"Plaintext-Ciphertext Matrix Multiplication and FHE Bootstrapping: Fast and Fused."* Crypto 2024

[BCHPS'25] Y. Bae, J. H. Cheon, G. Hanrot, J. H. Park, D. Stehlé. *"Encrypted Linear Algebra with BLAS."* arxiv, 2025

[JKLS'18] X. Jiang, M. Kim, K. Lauter, Y. Song. *"Secure Outsourced Matrix Computation and Application to Neural Networks."* CCS 2018

[LZ'22] J. Liu, L. F. Zhang. *"Privacy-preserving and publicly verifiable matrix multiplication."* IEEE Trans. on Services Computing, 2022