ASPLOS

UCF

ReliaFHE: Resilient Design for Fully Homomorphic Encryption Accelerators

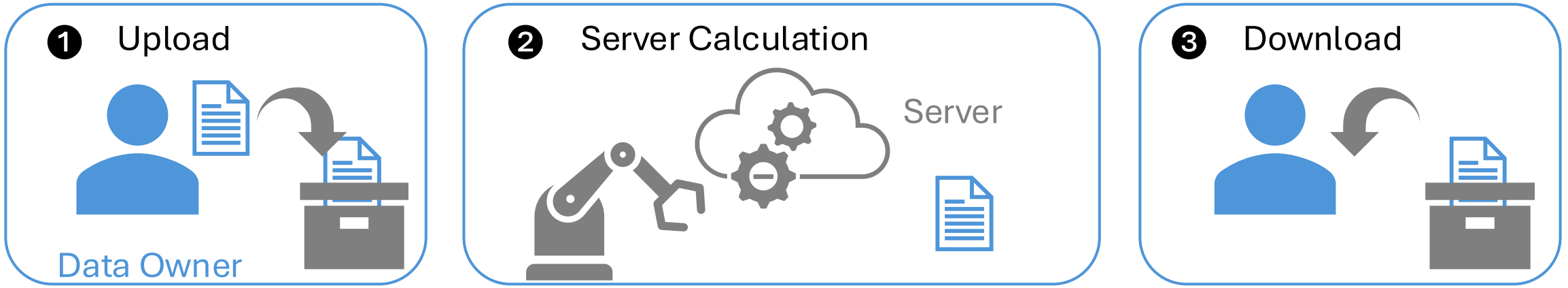
Fan Li, Mayank Kumar, Ruizhi Zhu, Mengxin Zheng, Qian Lou, Xin Xin*

College of Engineering and Computer Science

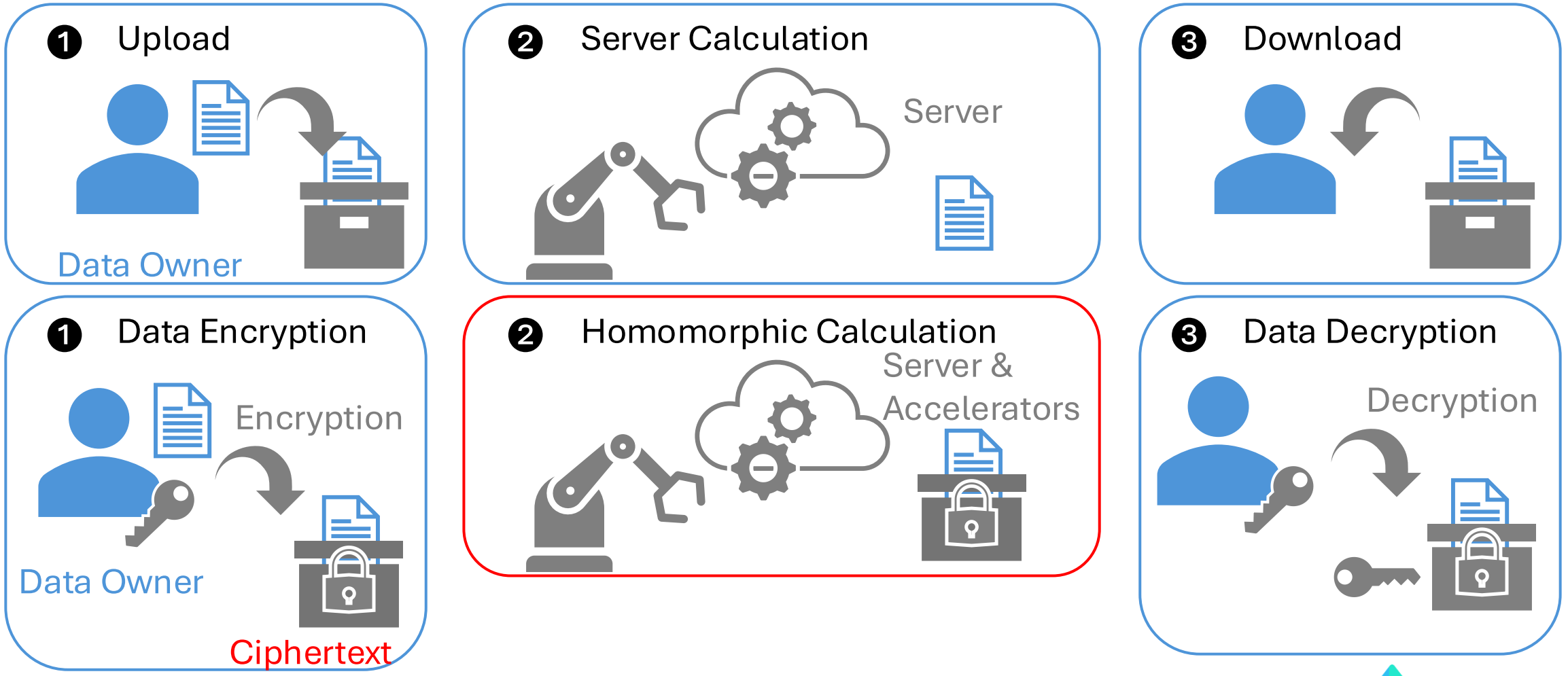
University of Central Florida

International Conference on Architectural Support for Programming Languages and Operating Systems

Fully Homomorphic Encryption (FHE)

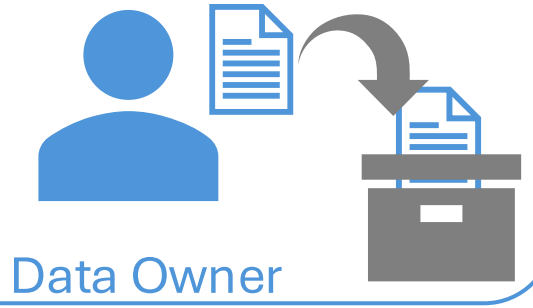


Fully Homomorphic Encryption (FHE)

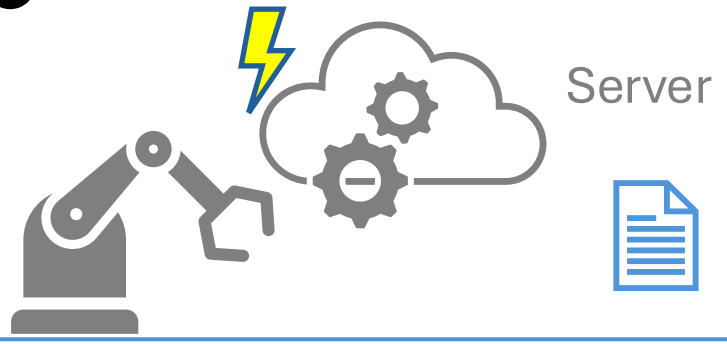


Fully Homomorphic Encryption (FHE)

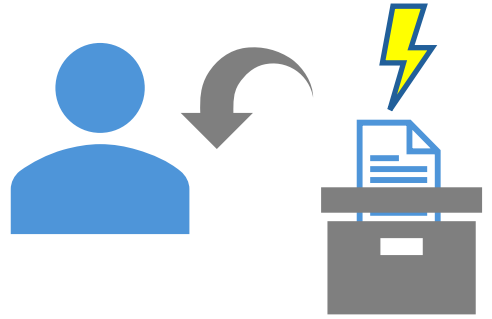
❶ Upload



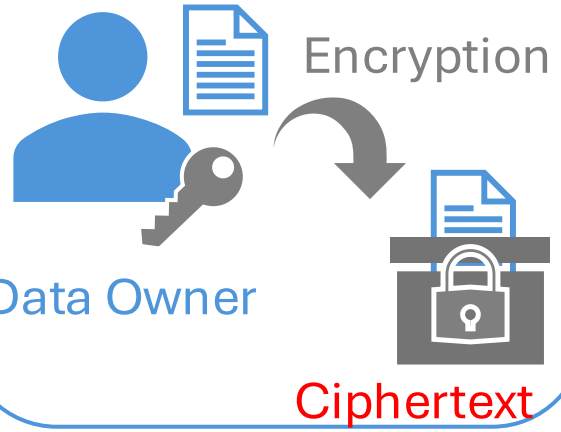
❷ Server Calculation



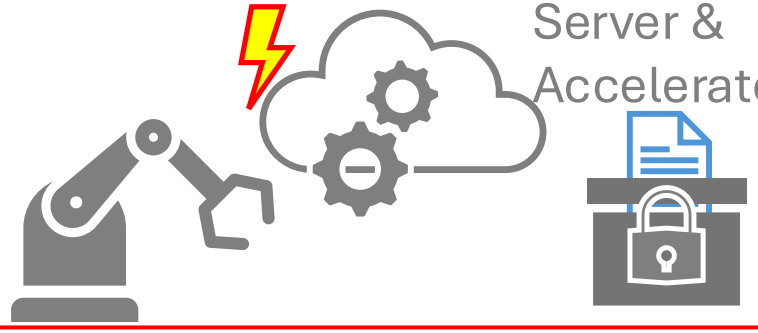
❸ Download



❶ Data Encryption

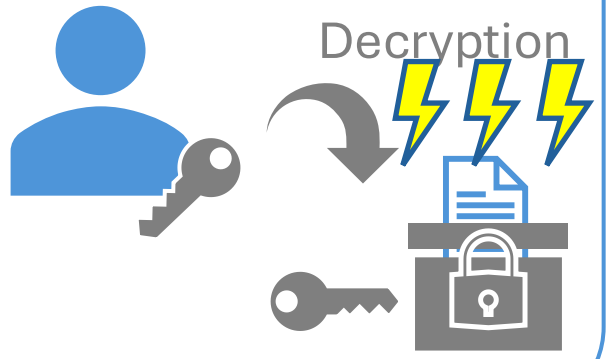


❷ Homomorphic Calculation



FHE is more error sensitive

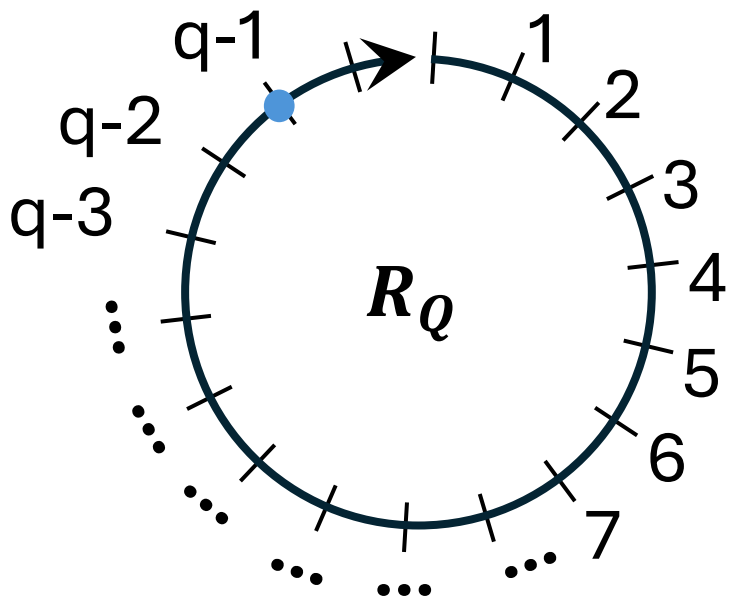
❸ Data Decryption



Error-Sensitivity Analysis

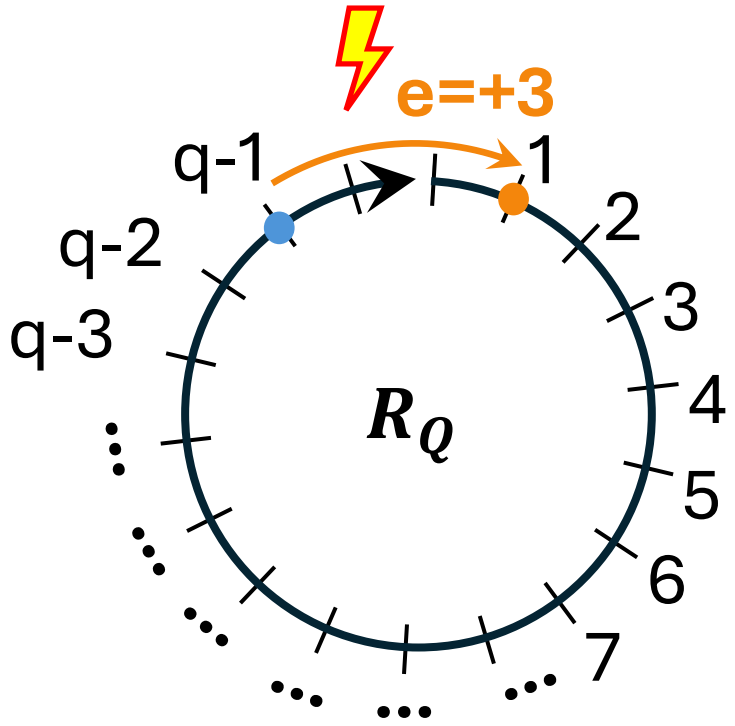
- Modulus Operations
- NTT Representation
- Packing Algorithm

Error-Sensitivity Analysis



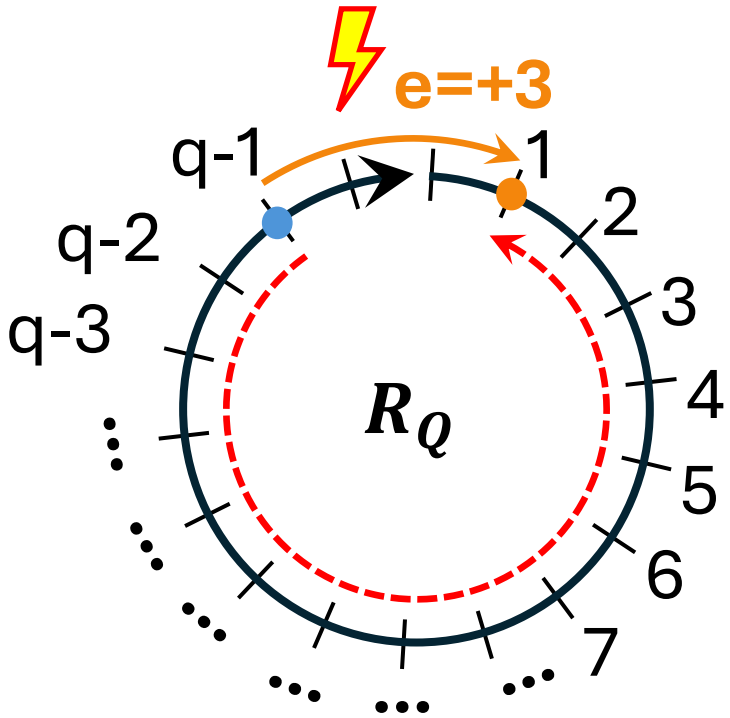
- **Modulus Operations**
- NTT Representation
- Packing Algorithm

Error-Sensitivity Analysis



- **Modulus Operations**
- NTT Representation
- Packing Algorithm

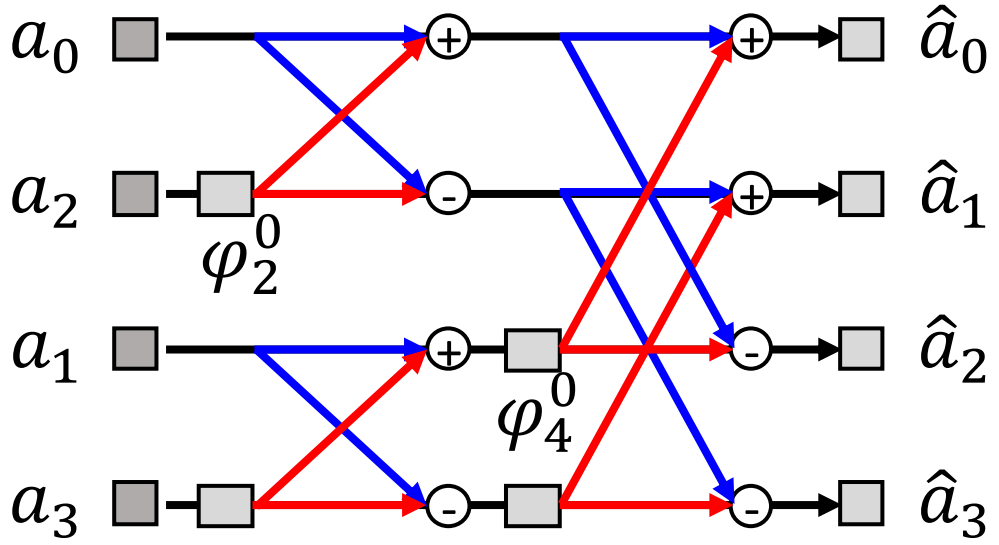
Error-Sensitivity Analysis



Error amplification

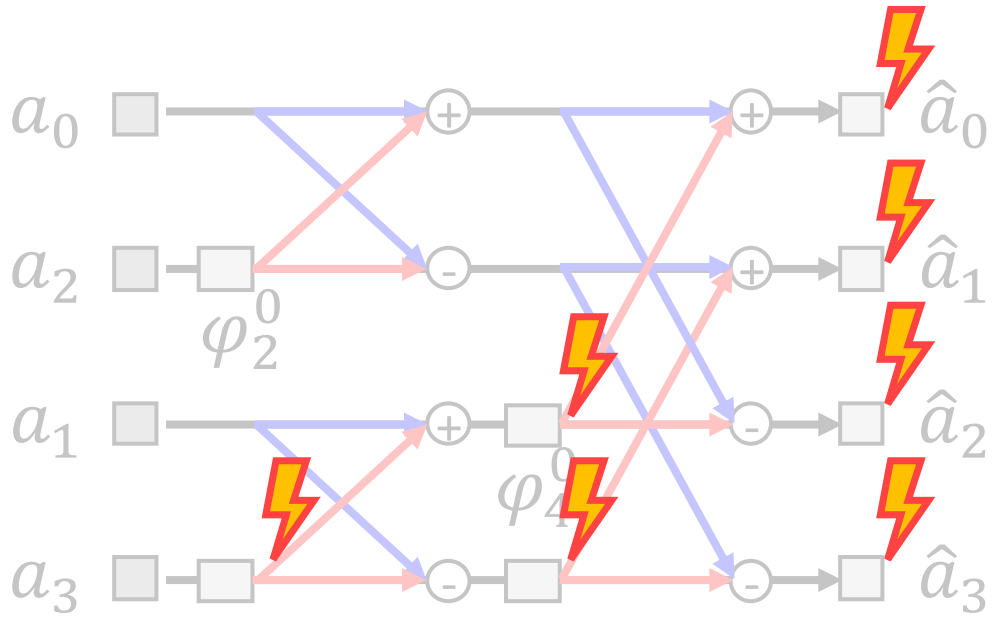
- **Modulus Operations**
- NTT Representation
- Packing Algorithm

Error-Sensitivity Analysis



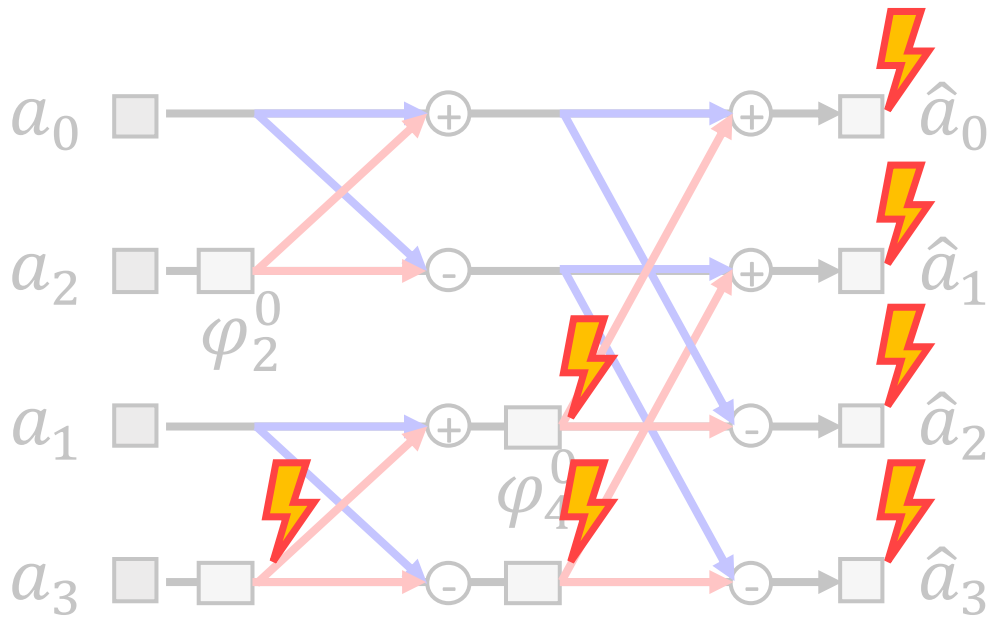
- Modulus Operations
- **NTT Representation**
- Packing Algorithm

Error-Sensitivity Analysis



- Modulus Operations
- **NTT Representation**
- Packing Algorithm

Error-Sensitivity Analysis

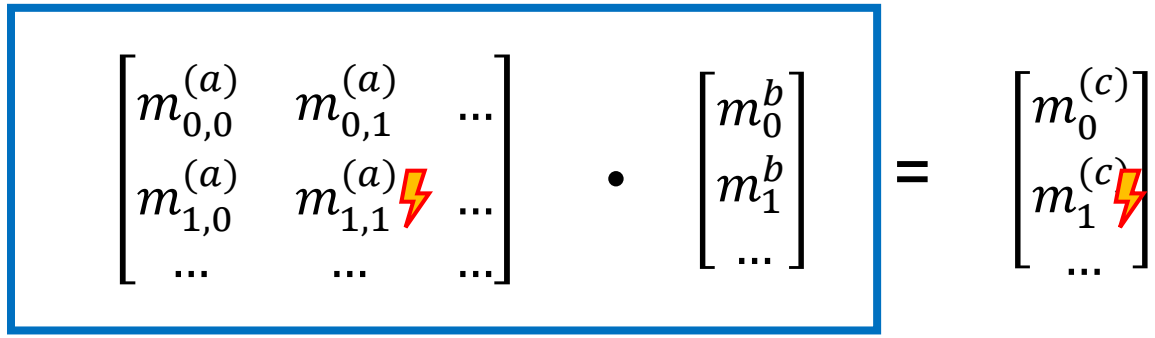


Error broadcast

- Modulus Operations
- **NTT Representation**
- Packing Algorithm

Error-Sensitivity Analysis

Plaintext



- Modulus Operations
- NTT Representation
- **Packing Algorithm**

Error-Sensitivity Analysis

Plaintext

$$\begin{bmatrix} m_{0,0}^{(a)} & m_{0,1}^{(a)} & \cdots \\ m_{1,0}^{(a)} & m_{1,1}^{(a)} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} m_0^b \\ m_1^b \\ \cdots \end{bmatrix} = \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

Ciphertext

$$\left(\begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \cdots \right) \cdot \begin{bmatrix} ct_0^{(b)} \\ ct_1^{(b)} \\ \cdots \end{bmatrix} \xrightarrow{Dec} \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

- Modulus Operations
- NTT Representation
- **Packing Algorithm**

Error-Sensitivity Analysis

Plaintext

$$\begin{bmatrix} m_{0,0}^{(a)} & m_{0,1}^{(a)} & \cdots \\ m_{1,0}^{(a)} & m_{1,1}^{(a)} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} m_0^b \\ m_1^b \\ \cdots \end{bmatrix} = \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

Ciphertext

$$\left(\begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \cdots \right) \cdot \begin{bmatrix} ct_0^{(b)} \\ ct_1^{(b)} \\ \cdots \end{bmatrix} \xrightarrow{Dec} \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

- Modulus Operations
- NTT Representation
- **Packing Algorithm**

Error-Sensitivity Analysis

Plaintext

$$\begin{bmatrix} m_{0,0}^{(a)} & m_{0,1}^{(a)} & \cdots \\ m_{1,0}^{(a)} & m_{1,1}^{(a)} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \cdot \begin{bmatrix} m_0^b \\ m_1^b \\ \cdots \end{bmatrix} = \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

Ciphertext

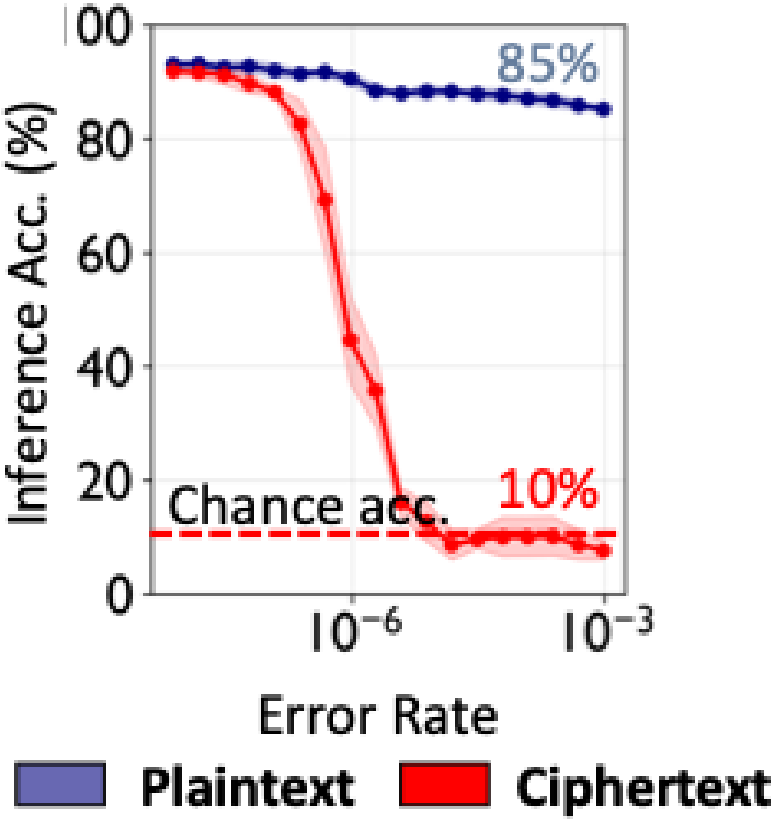
$$\left(\begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \begin{bmatrix} ct_0^{(a)} \\ ct_1^{(a)} \\ \cdots \end{bmatrix} + \cdots \right) \cdot \begin{bmatrix} ct_0^{(b)} \\ ct_1^{(b)} \\ \cdots \end{bmatrix} \xrightarrow{Dec} \begin{bmatrix} m_0^{(c)} \\ m_1^{(c)} \\ \cdots \end{bmatrix}$$

Error broadcast

- Modulus Operations
- NTT Representation
- **Packing Algorithm**

Error-Sensitivity Analysis

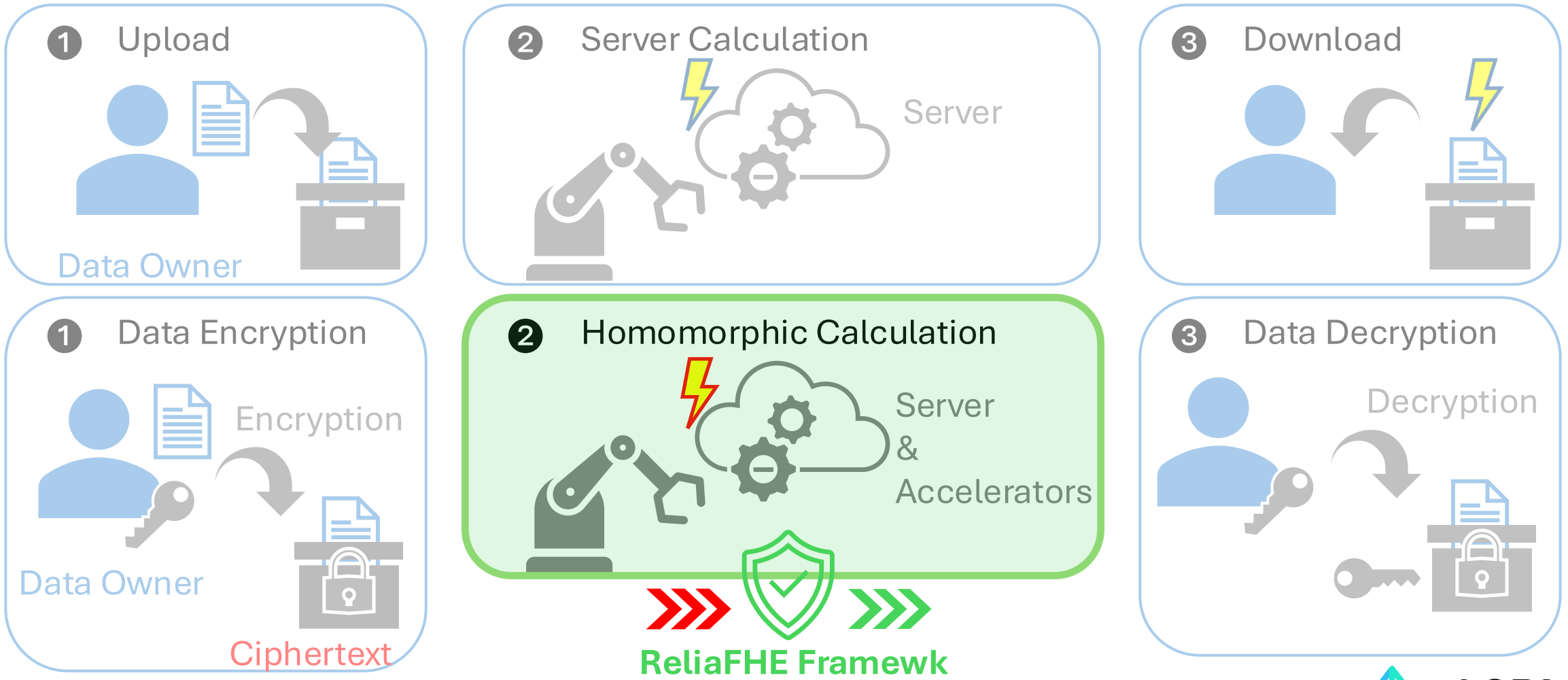
MNIST classification



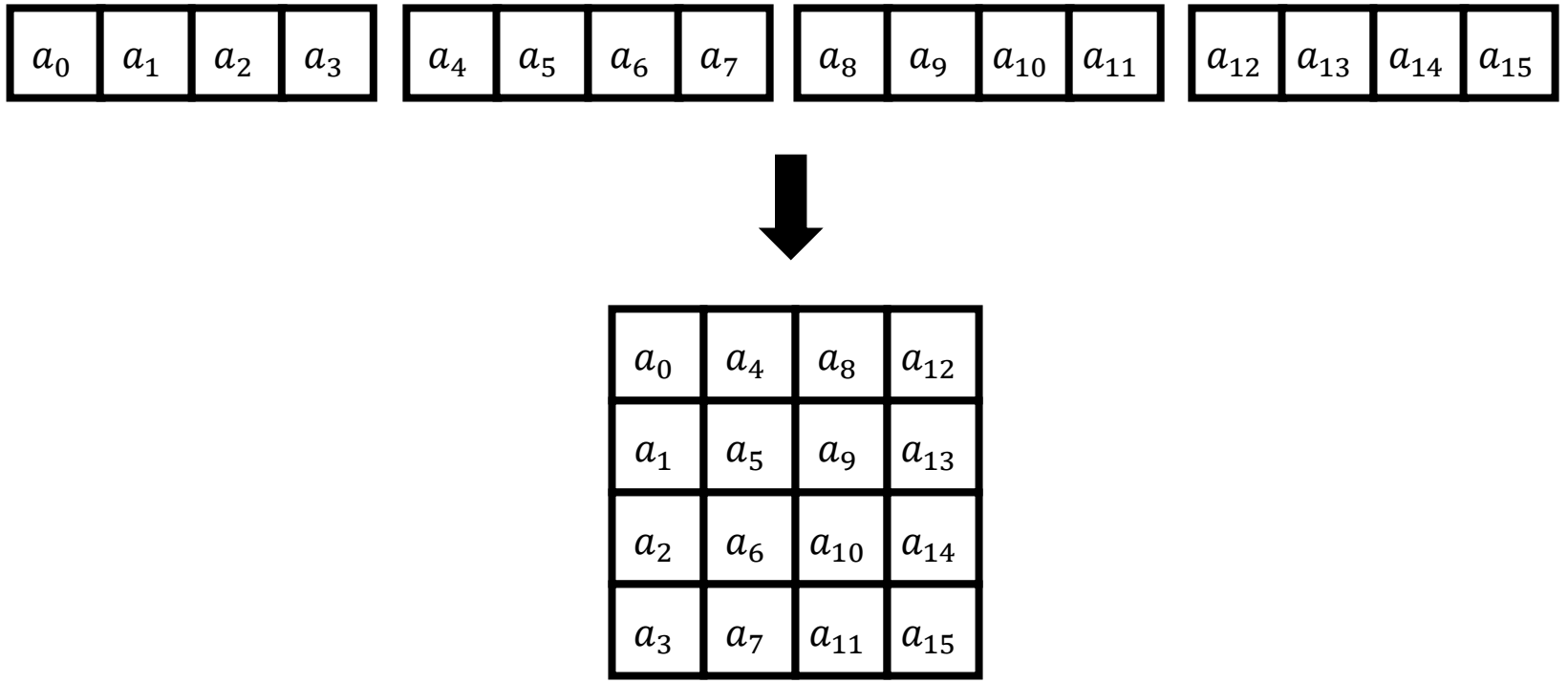
FHE-based model is more error sensitive

- Modulus Operations
- NTT Representation
- Packing Algorithm

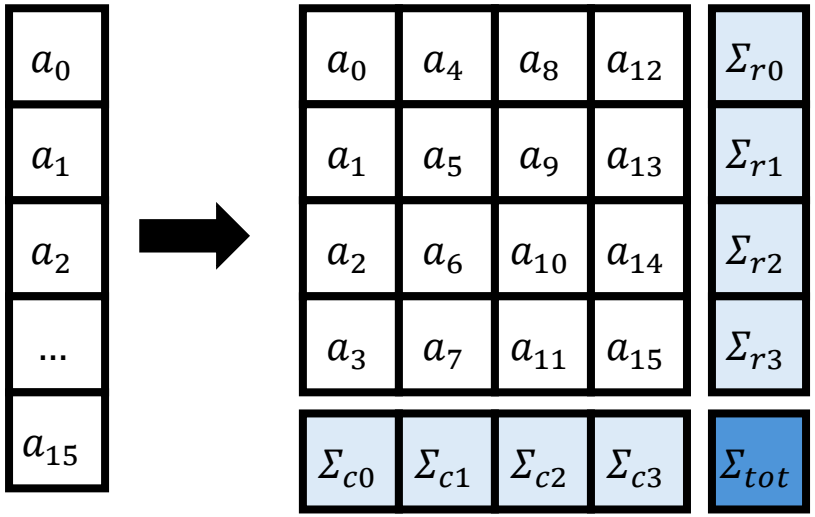
ReliaFHE Framework



Storage Protection(e.g. coefficient storage)



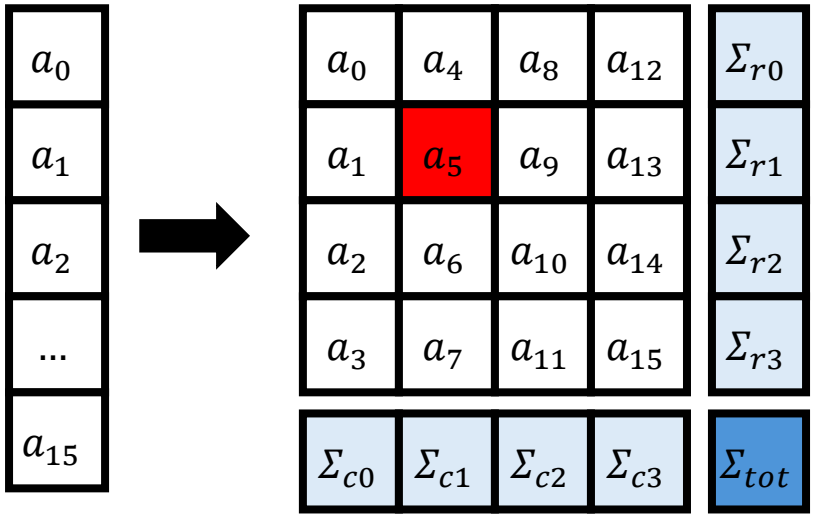
Storage Protection(e.g. coefficient storage)



Σ_r, Σ_c : row and col-wise Checksum

Σ_{tot}: total Checksum

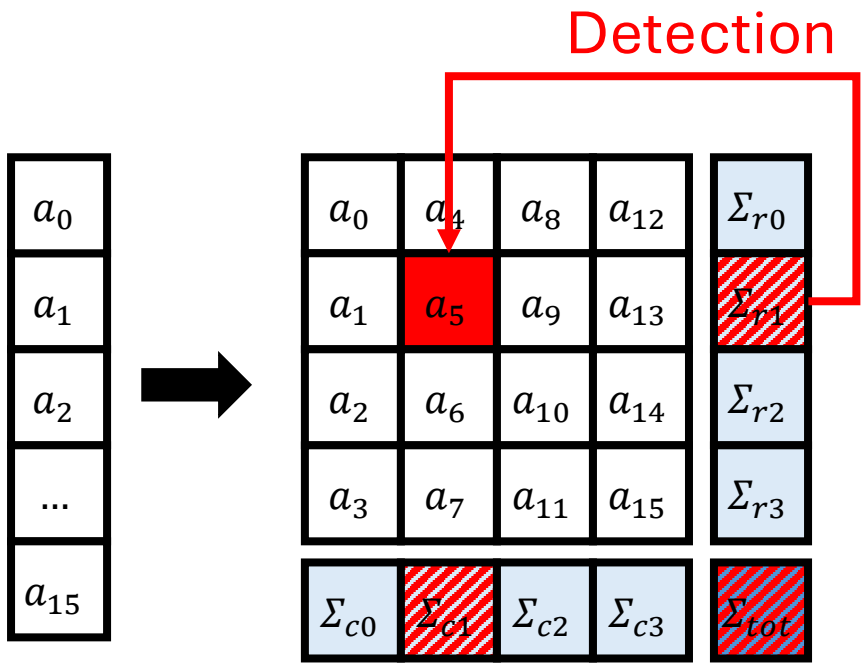
Storage Protection(e.g. coefficient storage)



Σ_r, Σ_c : row and col-wise Checksum

Σ_{tot}: total Checksum

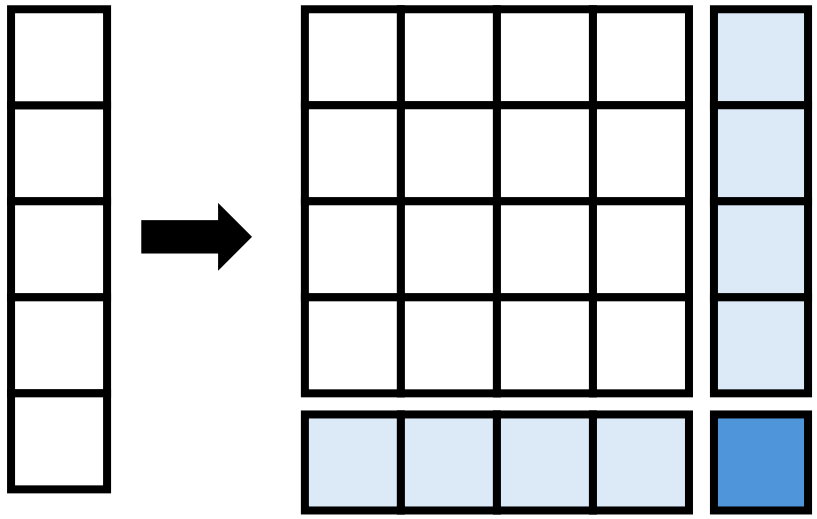
Storage Protection(e.g. coefficient storage)



Σ_r, Σ_c : row and col-wise Checksum

Σ_{tot}: total Checksum

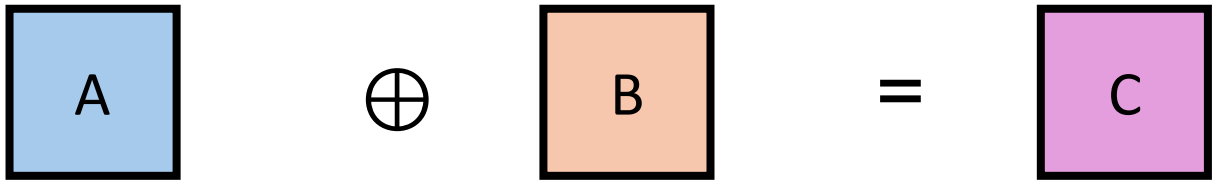
Storage Protection(e.g. coefficient storage)



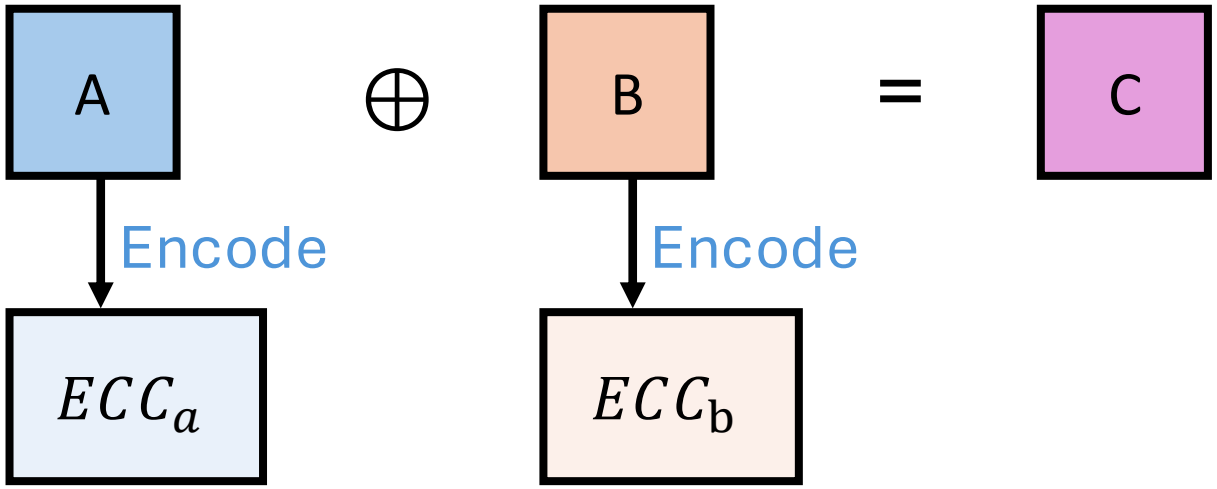
Redundancy: $O(\sqrt{n})$

e.g.
|v|=65536
|R|= 256
0.3% Redundancy

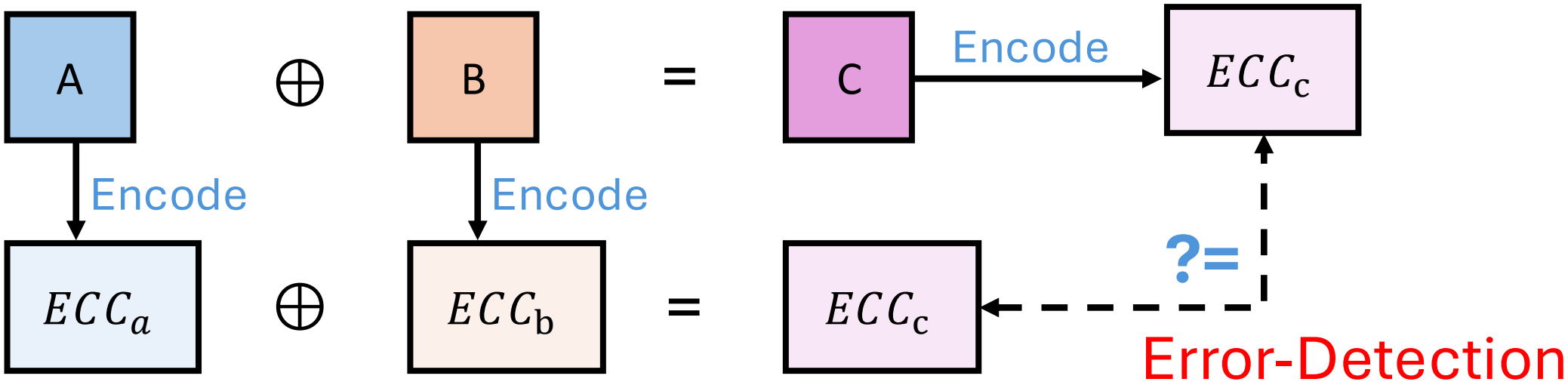
Computation Protection



Computation Protection



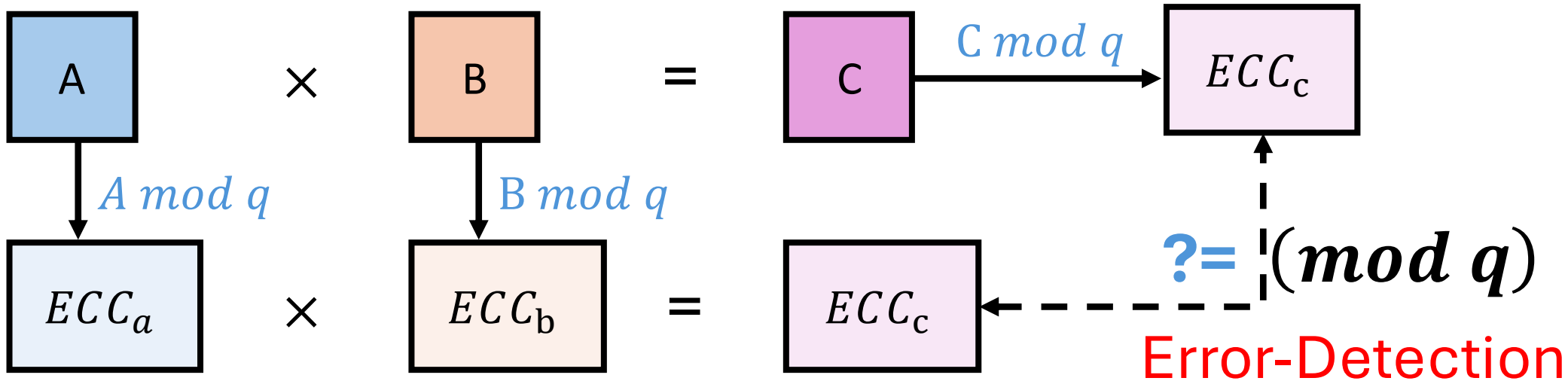
Computation Protection



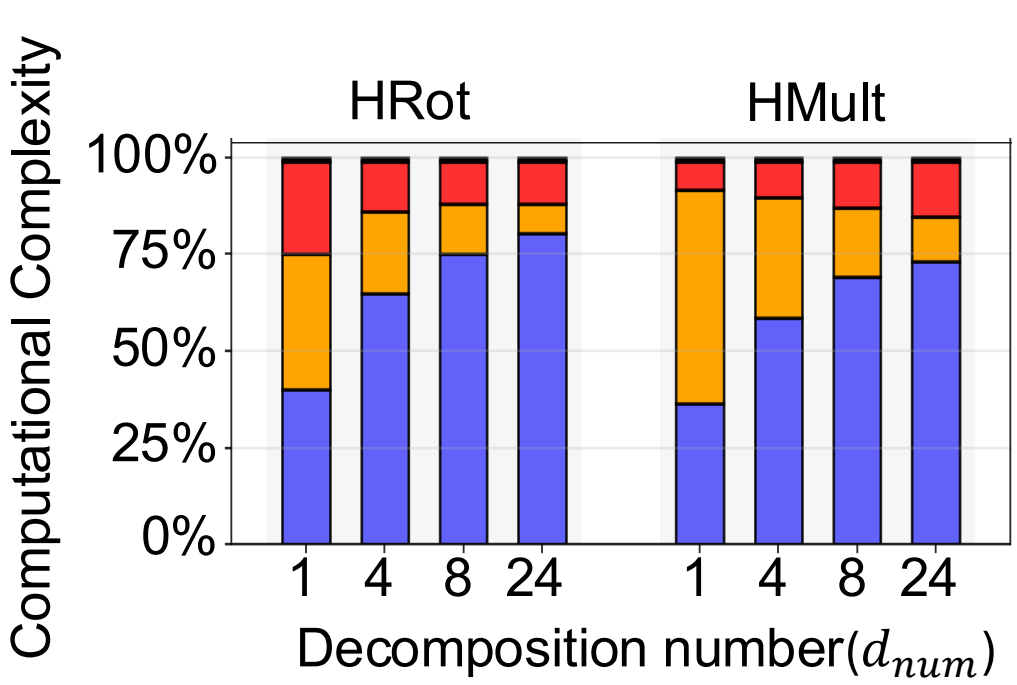
Construct computational compatible ECC

Computation Protection

e.g.



FHE Computation Kernels



Computational Protection

NTT

Elementwise Operation

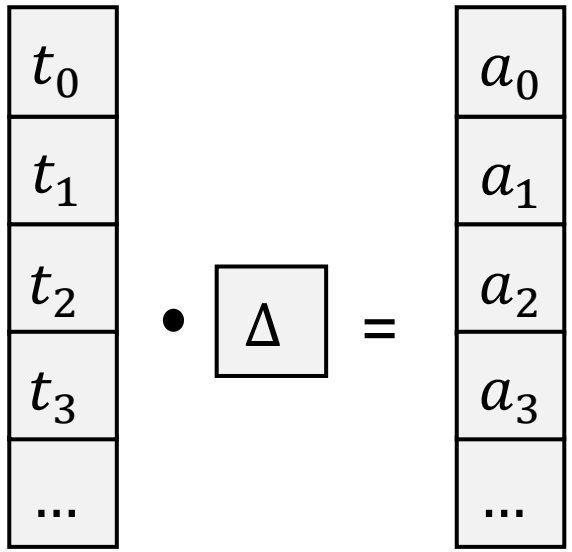
Scale Operation

> 98%

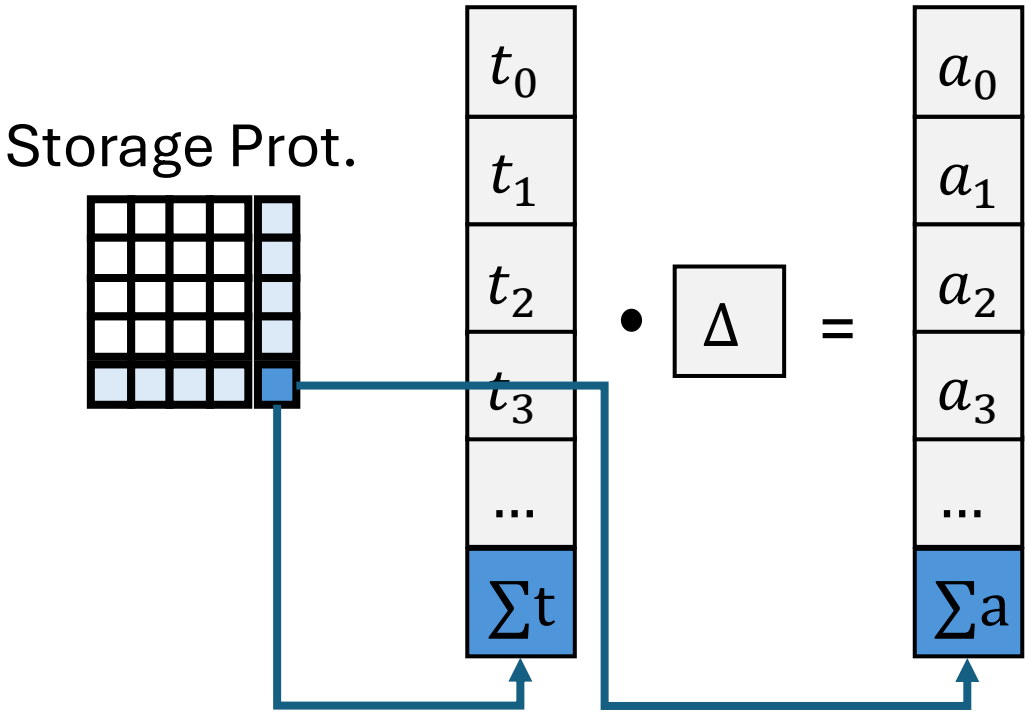
Others

ASPLOS

Scaling Operation (e.g., BConv)

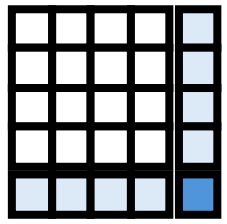


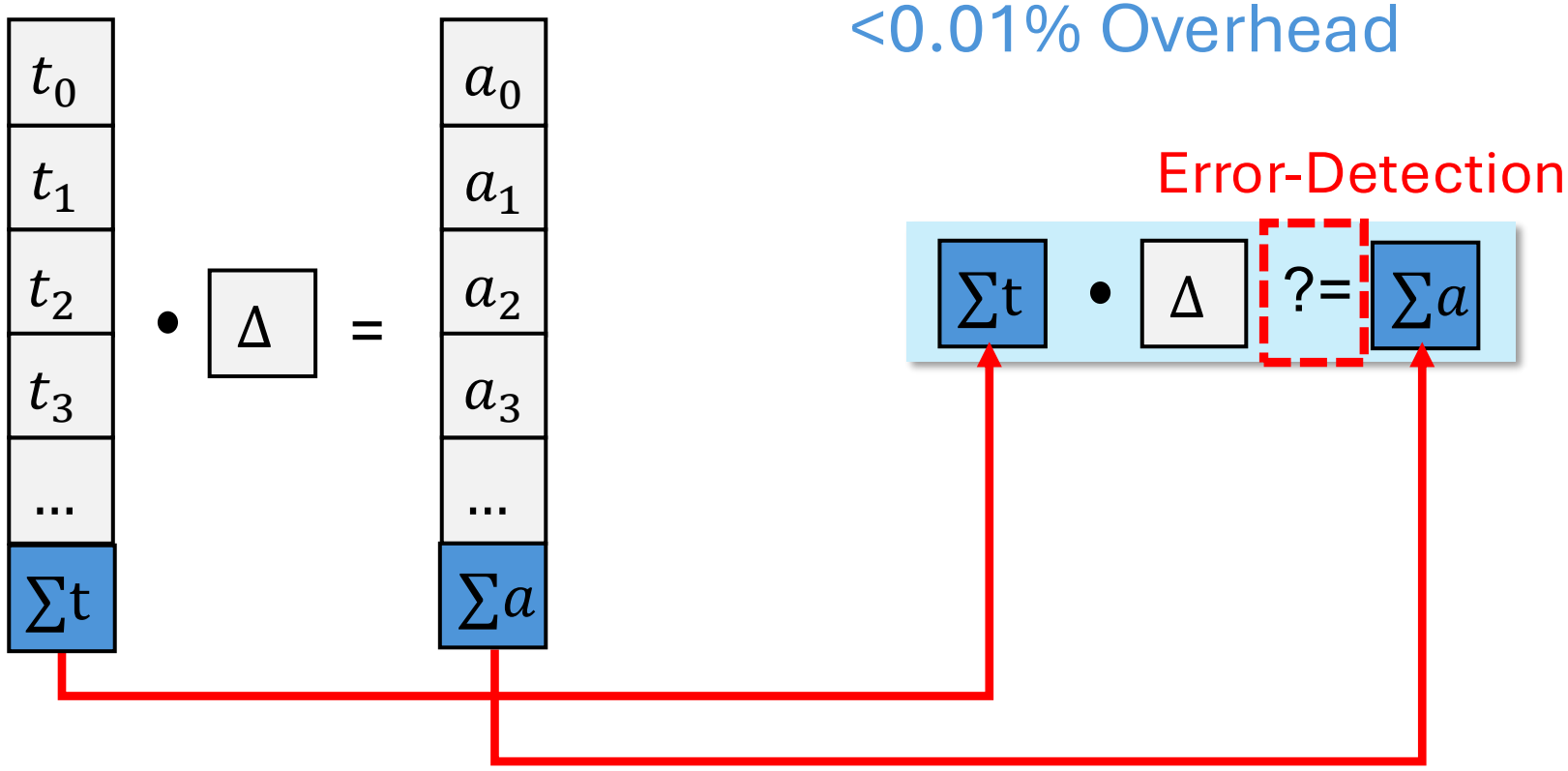
Scaling Operation (e.g., BConv)



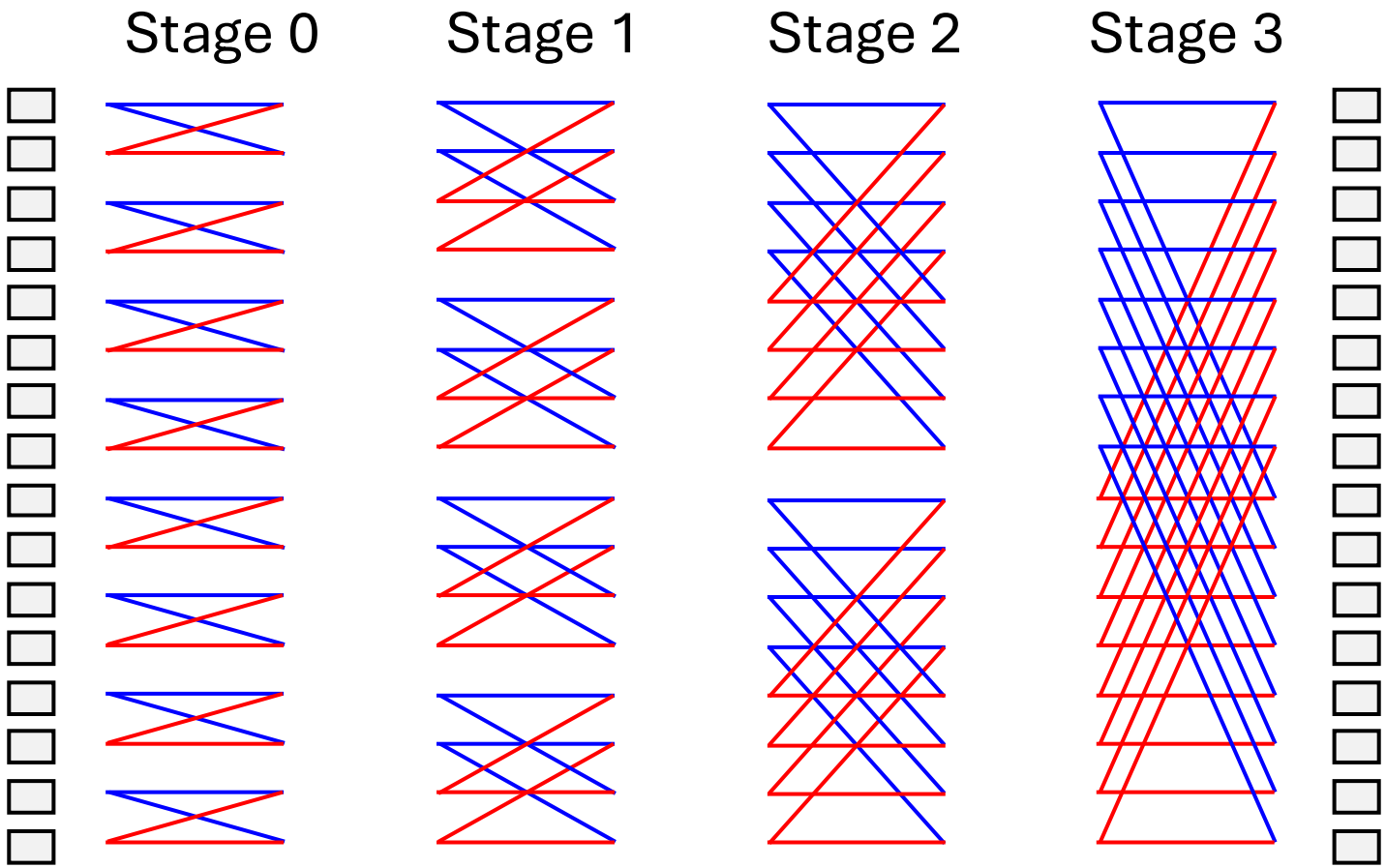
Scaling Operation (e.g., BConv)

Storage Prot.

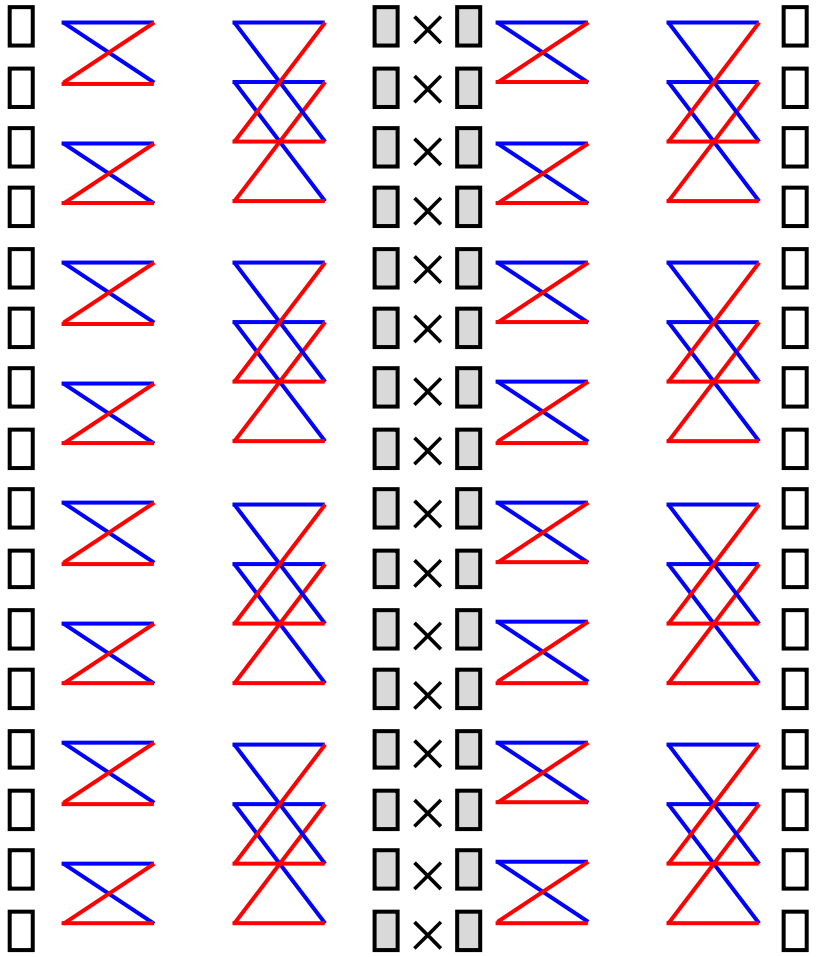




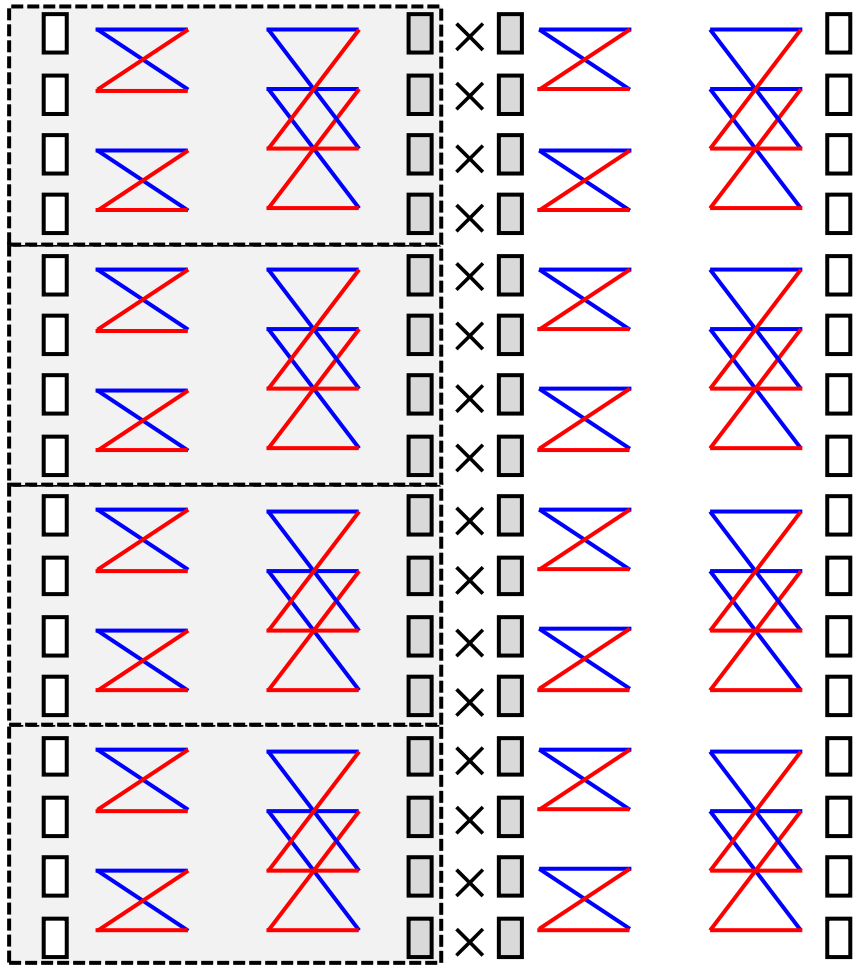
NTT Protection

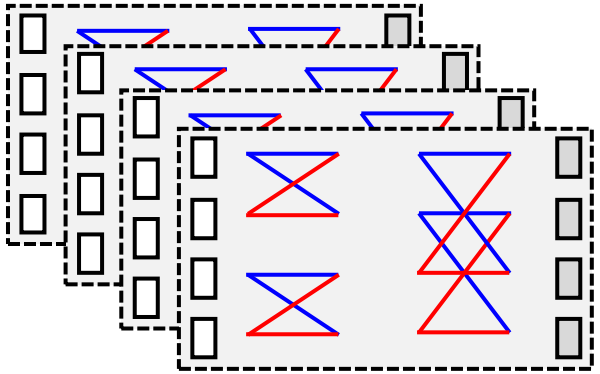


NTT Protection

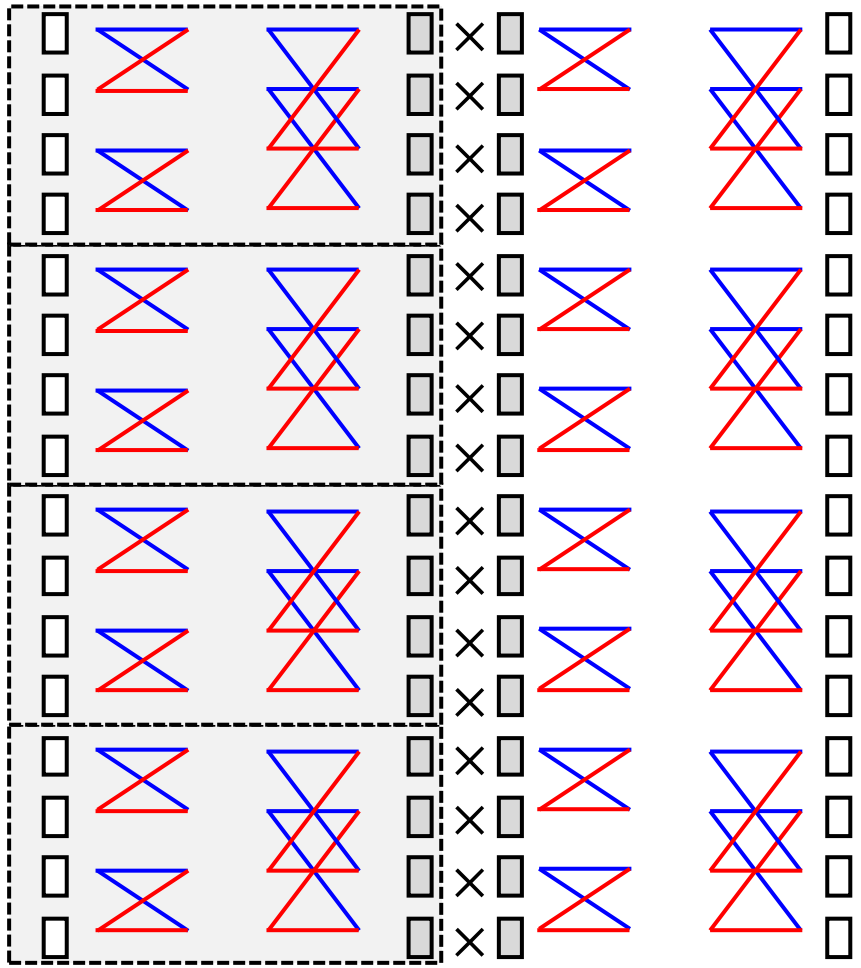


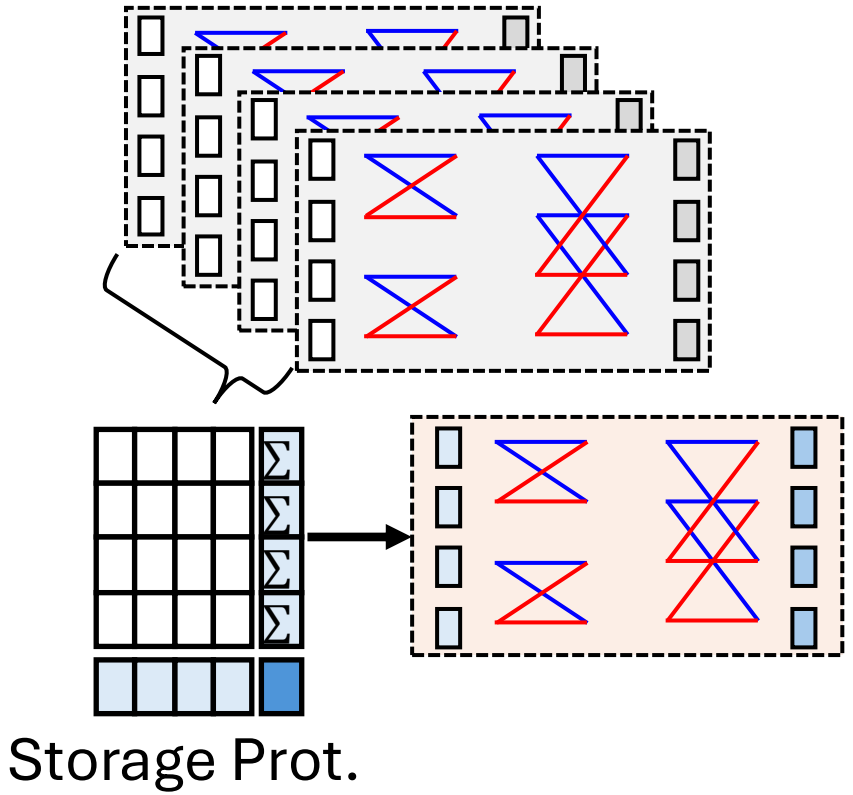
NTT Protection



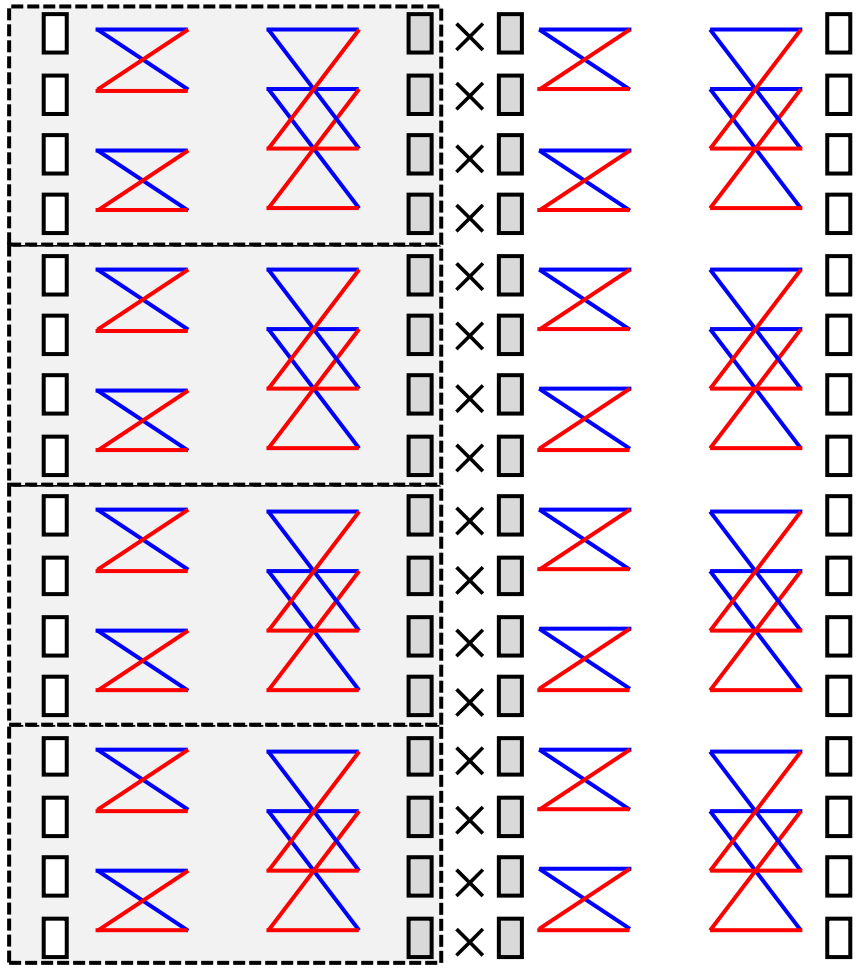


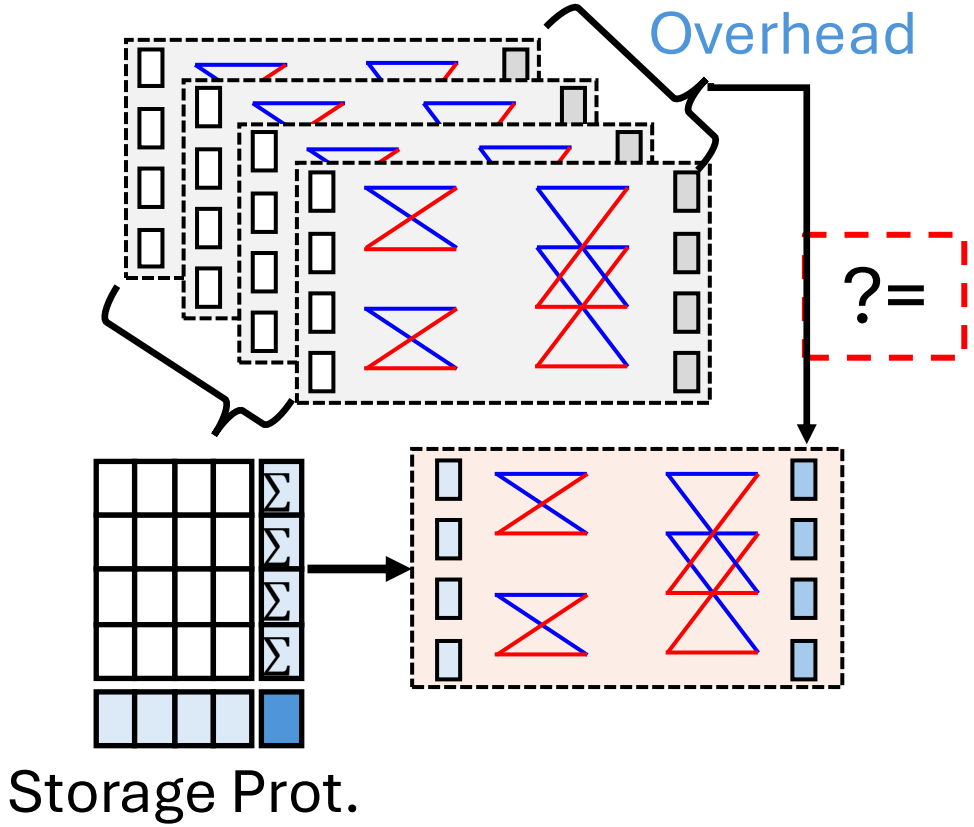
NTT Protection



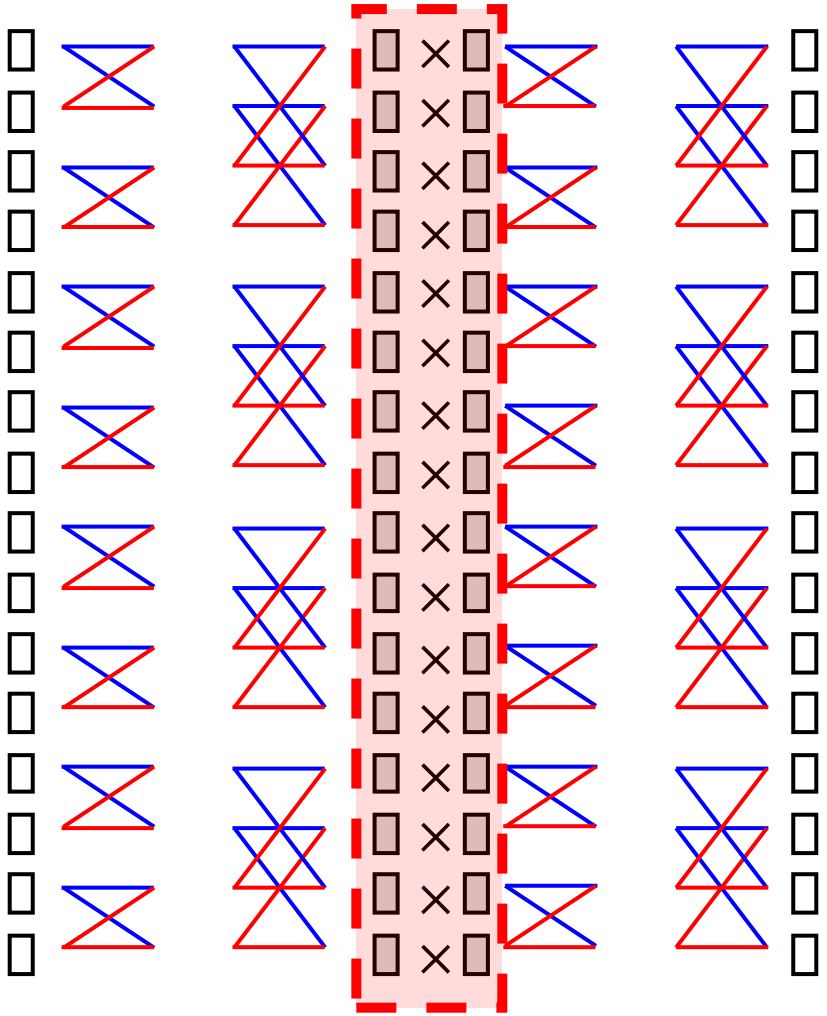


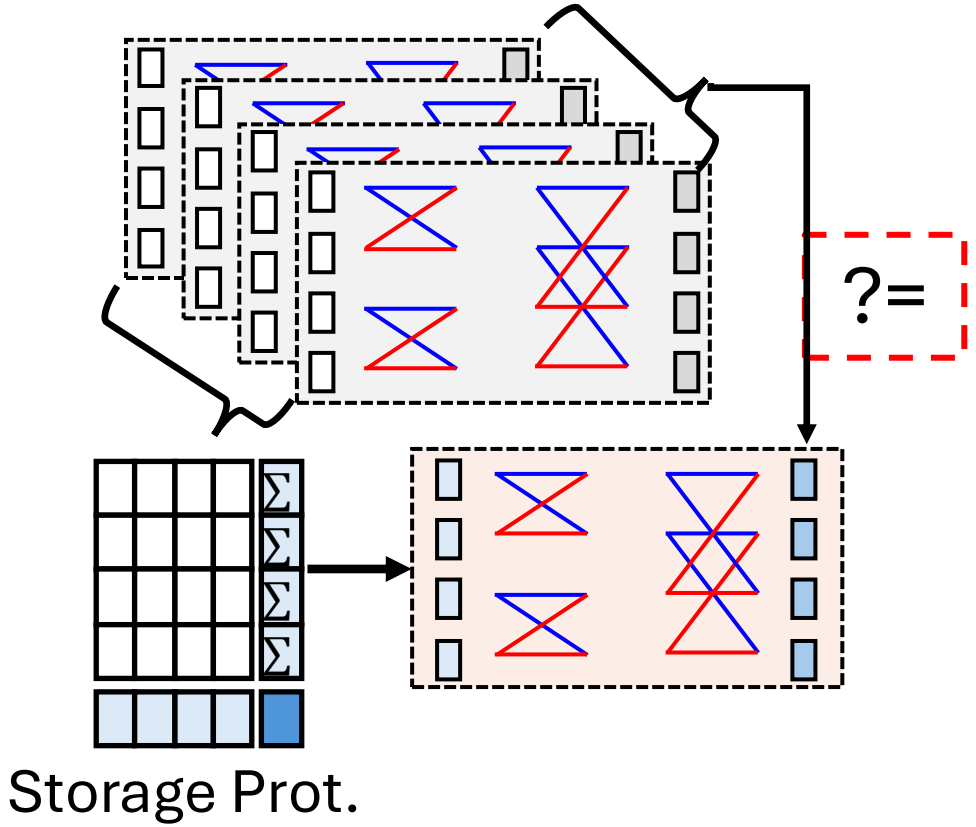
NTT Protection



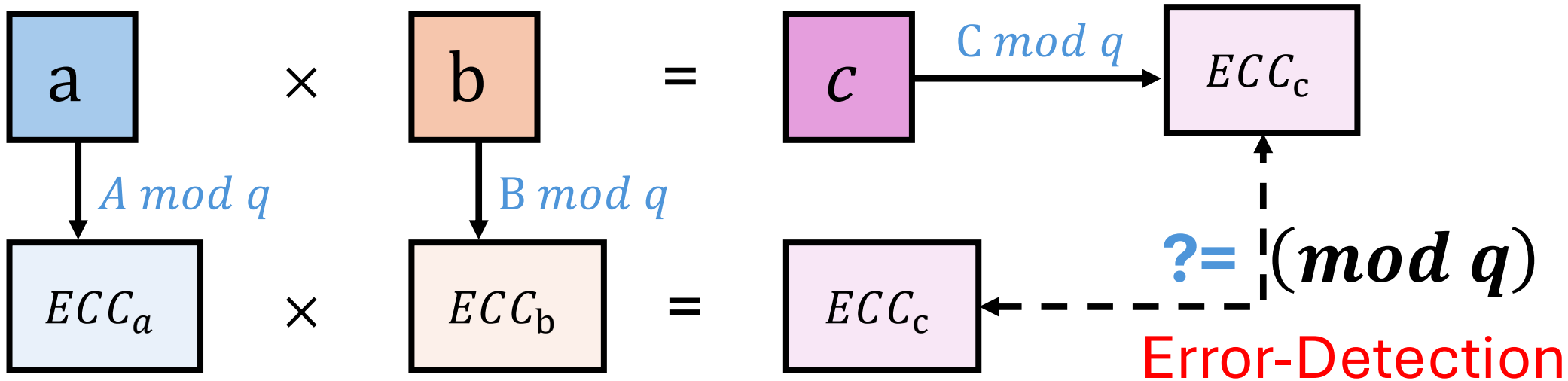


NTT Protection

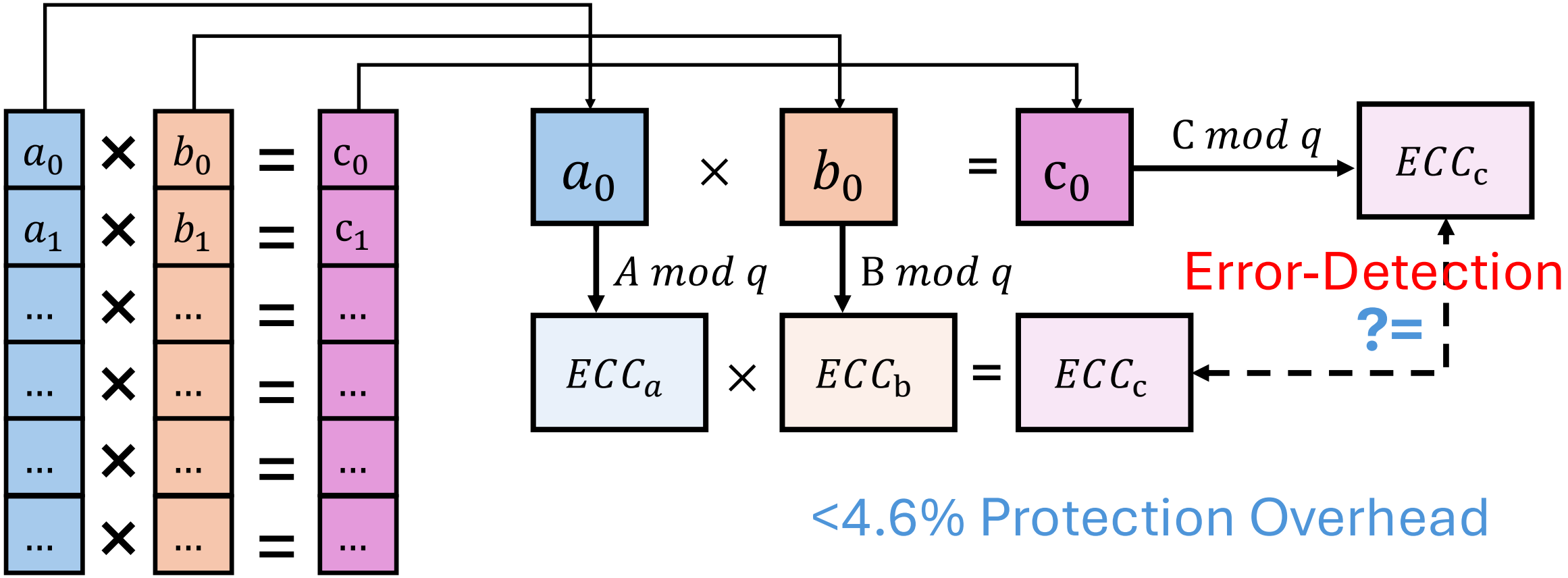




Elementwise Protection (e.g. EvkMult)

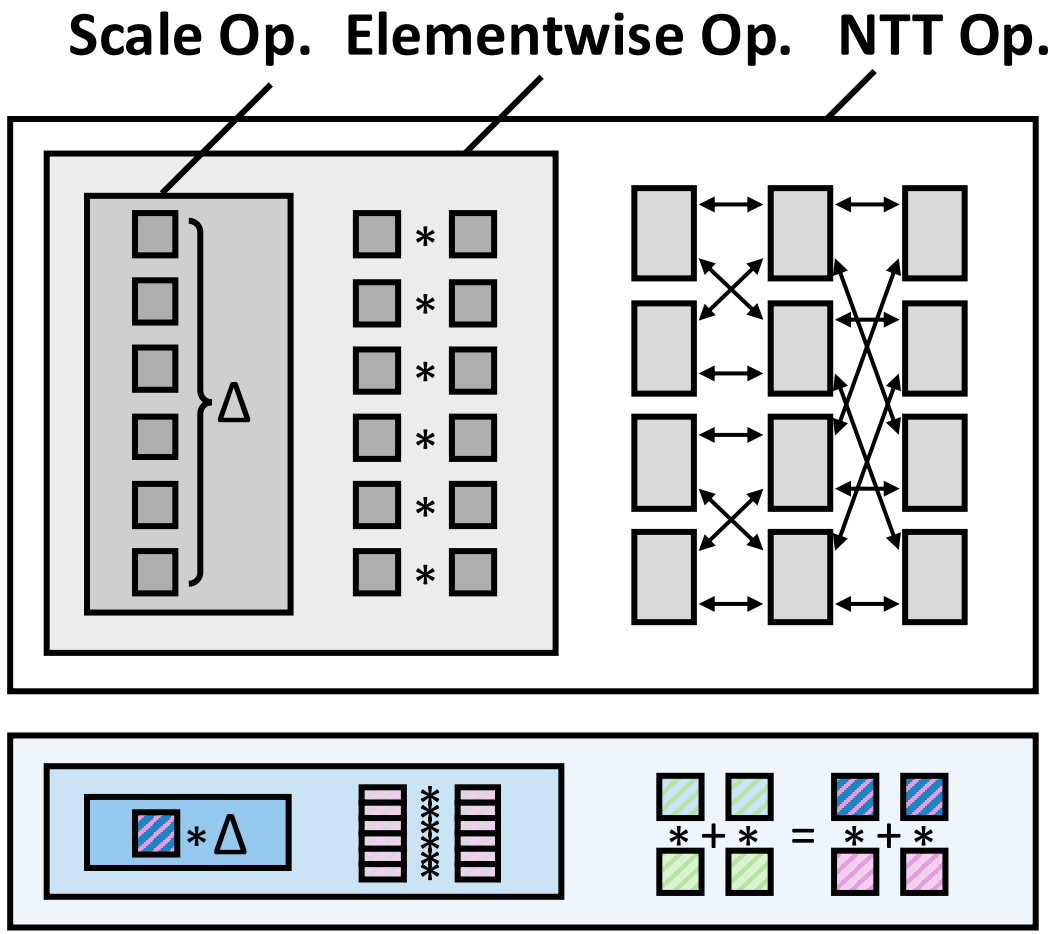


Elementwise Protection (e.g. EvkMult)



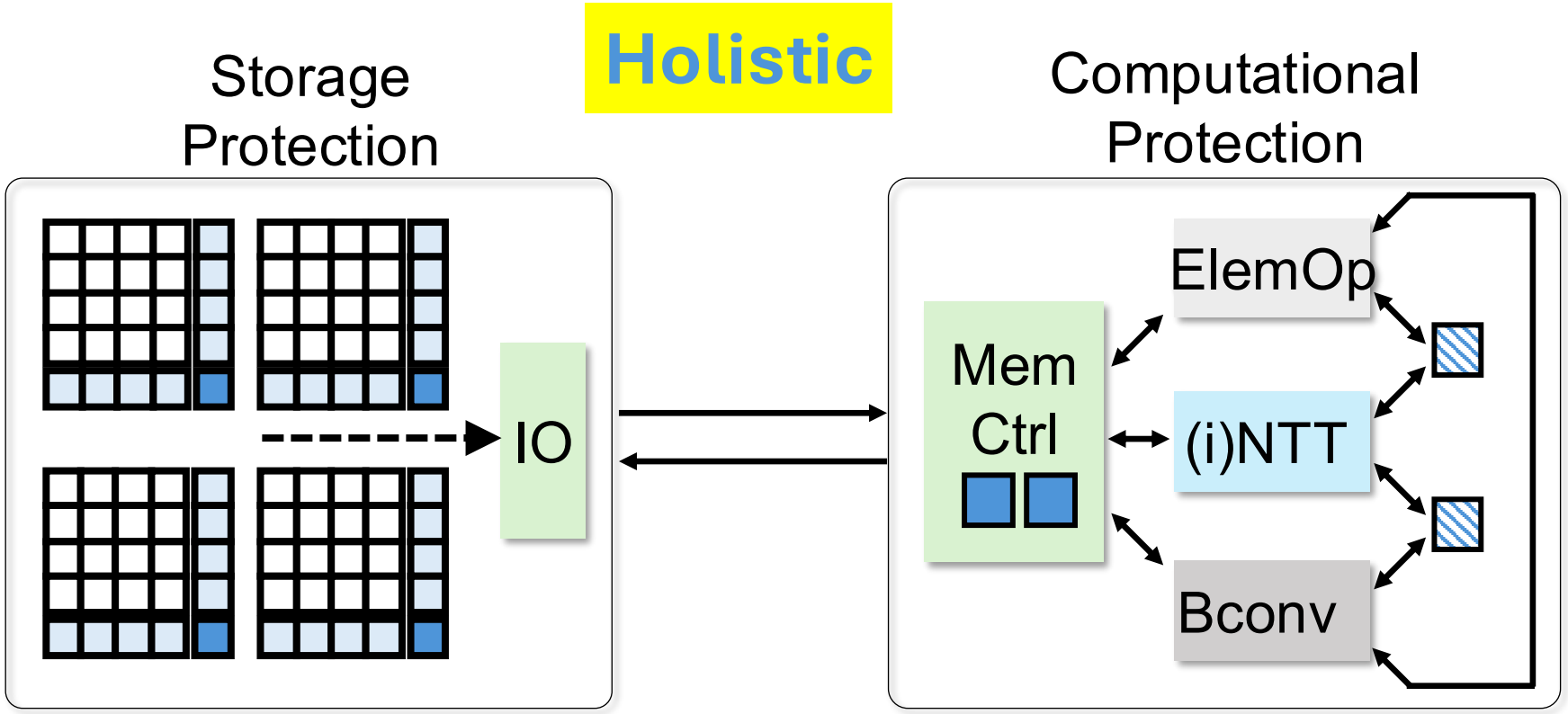
<4.6% Protection Overhead

Conclusion



Hierarchical

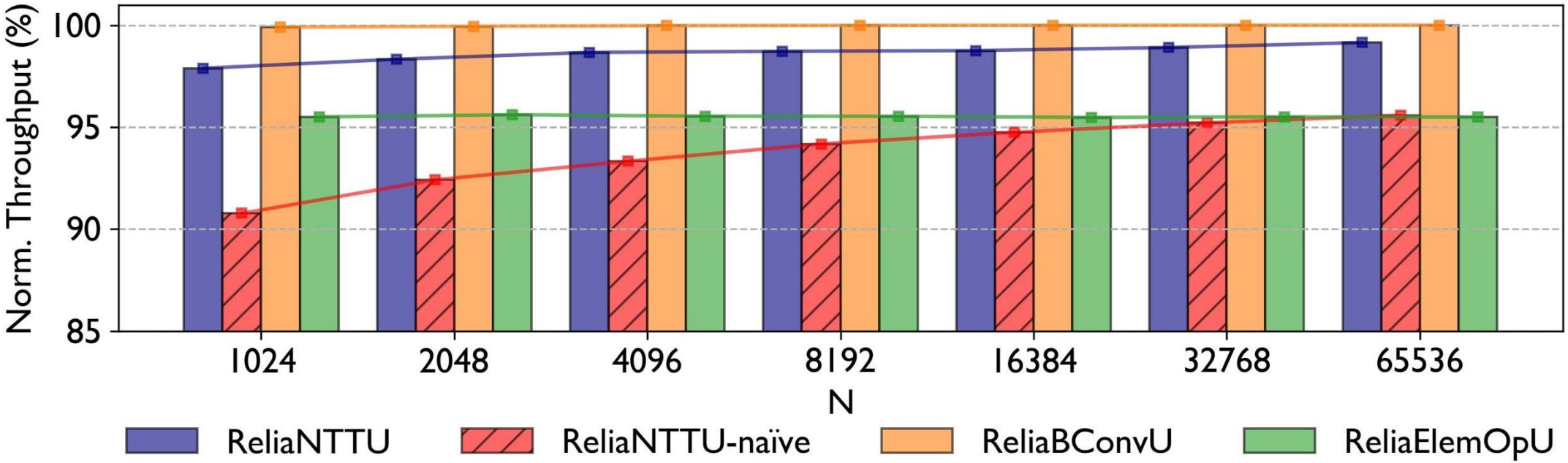
Conclusion


Holistic
Storage Protection
Computational Protection
IO
Mem Ctrl
ElemOp
(i)NTT
Bconv

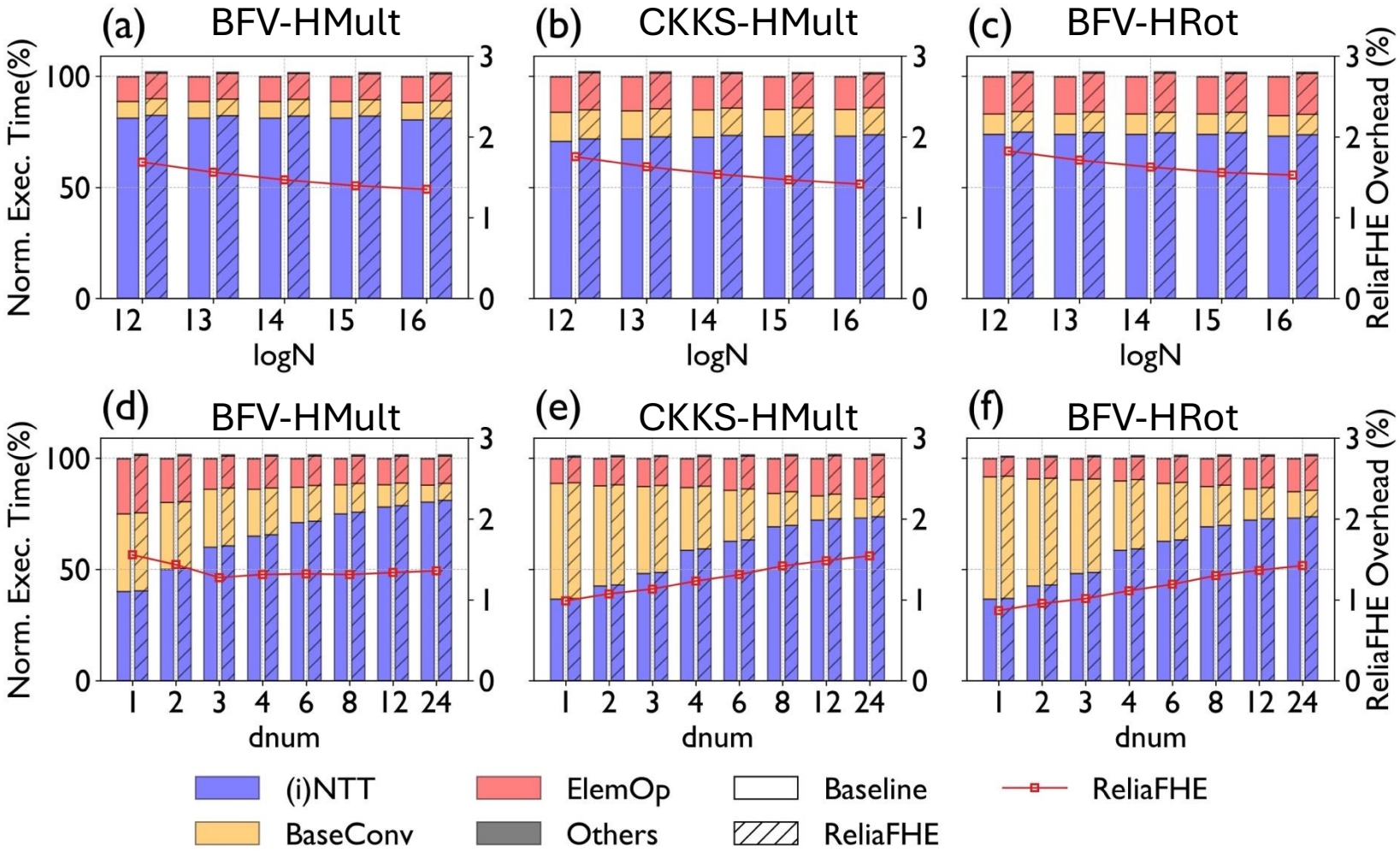
Conclusion

Holistic	Hierarchical	Lightweight
Complete Protection	**Multi-level Defense**	**Minimal Impact**
Integrated security for both **Memory** and **Computation**.	Layered granularity: **Scale operation** →**Elementwise Operation** → **NTT operation**.	Exceptional efficiency with only **1%~2%** overhead.

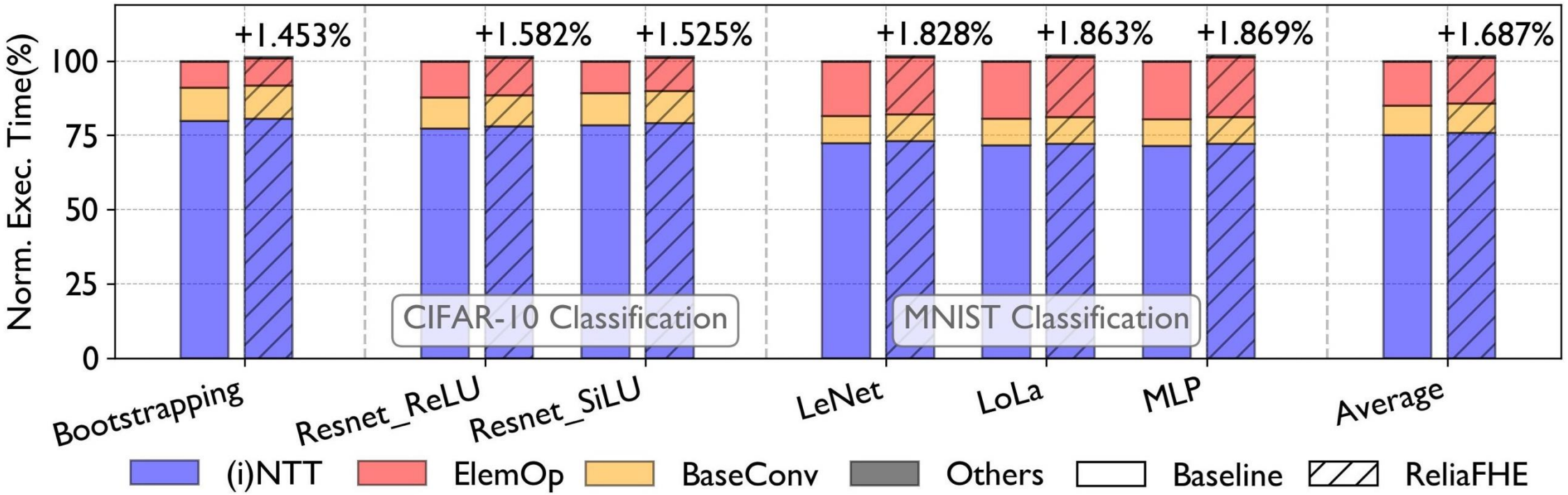
Performance-Operation Kernels



Performance-Primitive Operations

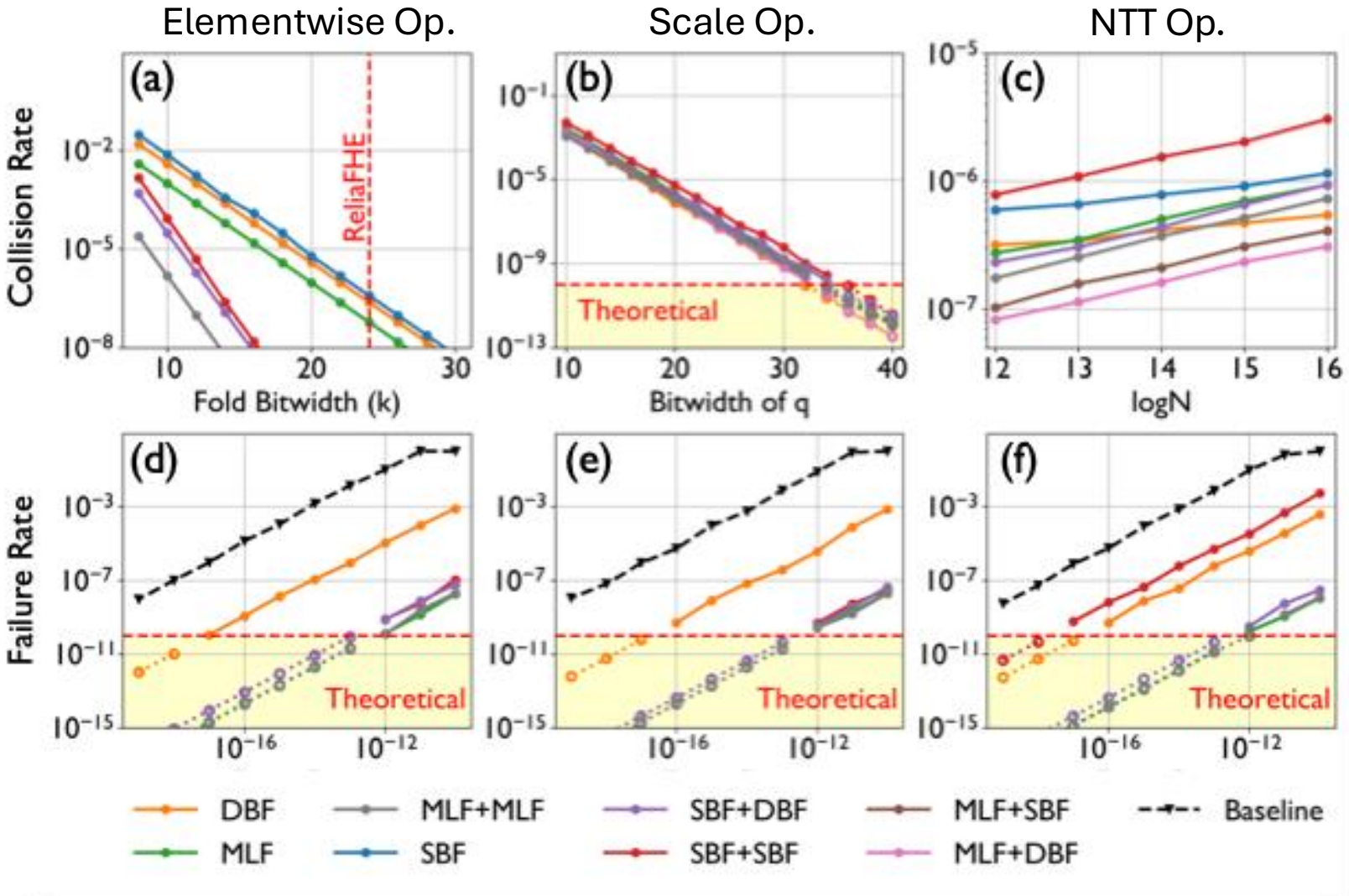


Performance-Application



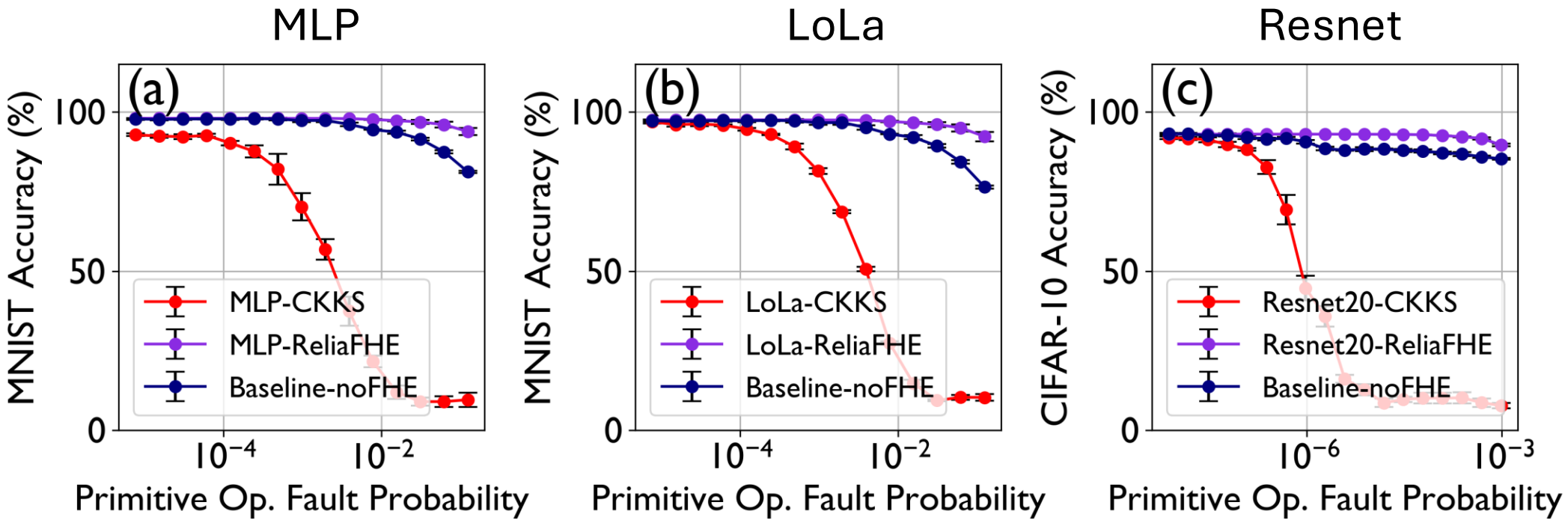
ASPLOS

Reliability



ASPLOS

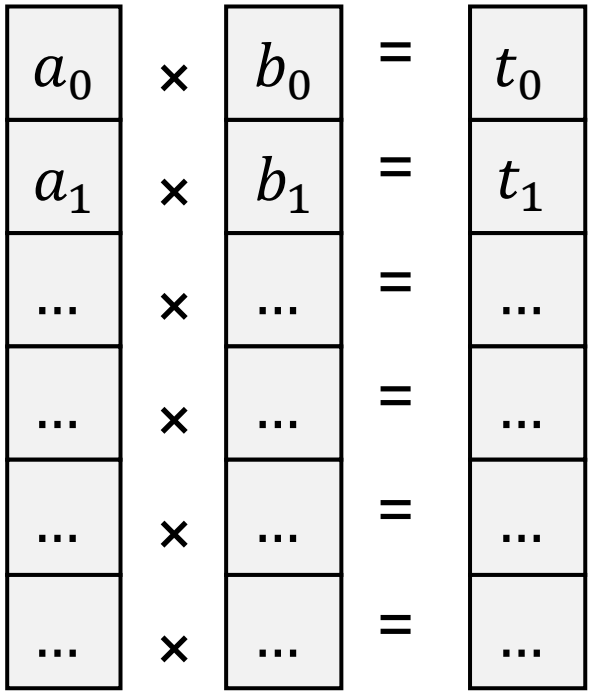
Error-tolerance Ability



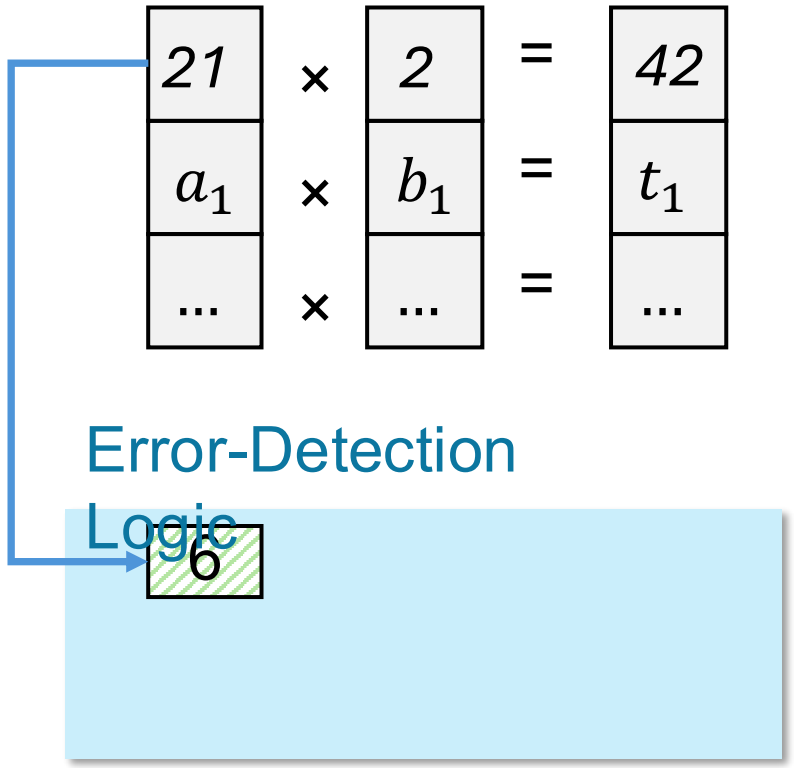
Thank you for listening
Q&A

Fan Li

Elementwise Operation (e.g. EvkMult)



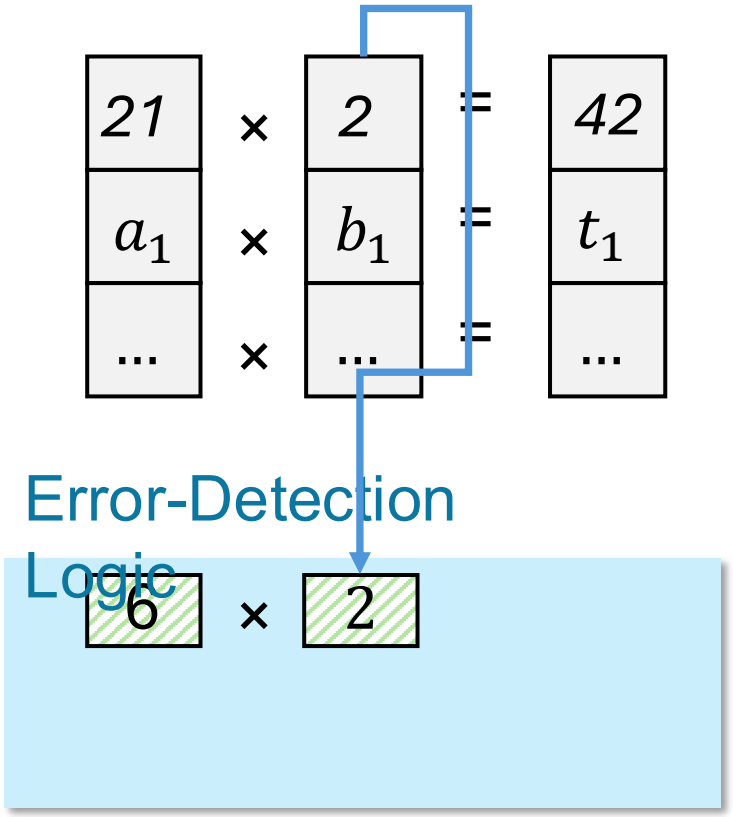
Elementwise Operation (e.g. EvkMult)



21=b0000 0000 0001 0101

21_{Σ}=b(0000 + 0000 + 0001 + 0101)
=b0110
=6

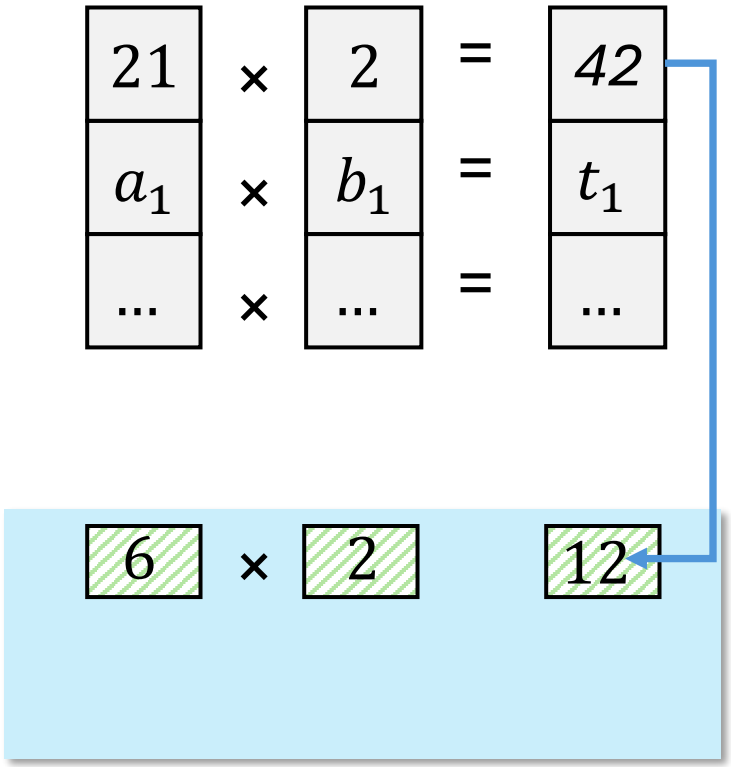
Elementwise Operation (e.g. EvkMult)



2 =b0000 0000 0000 0010

2_Σ=b(0000 + 0000 + 0000 + 0010)
=b0010
=2

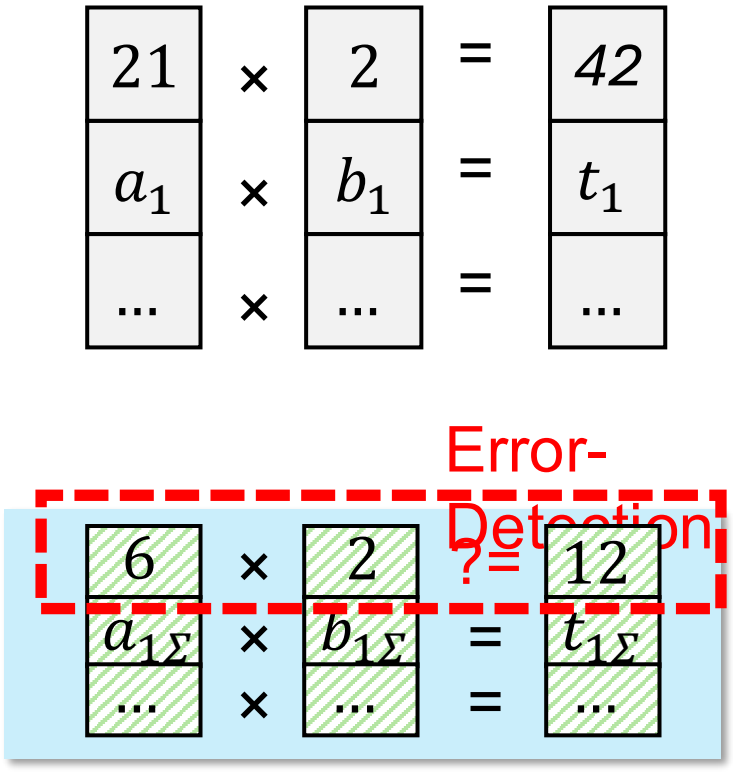
Elementwise Operation (e.g. EvkMult)



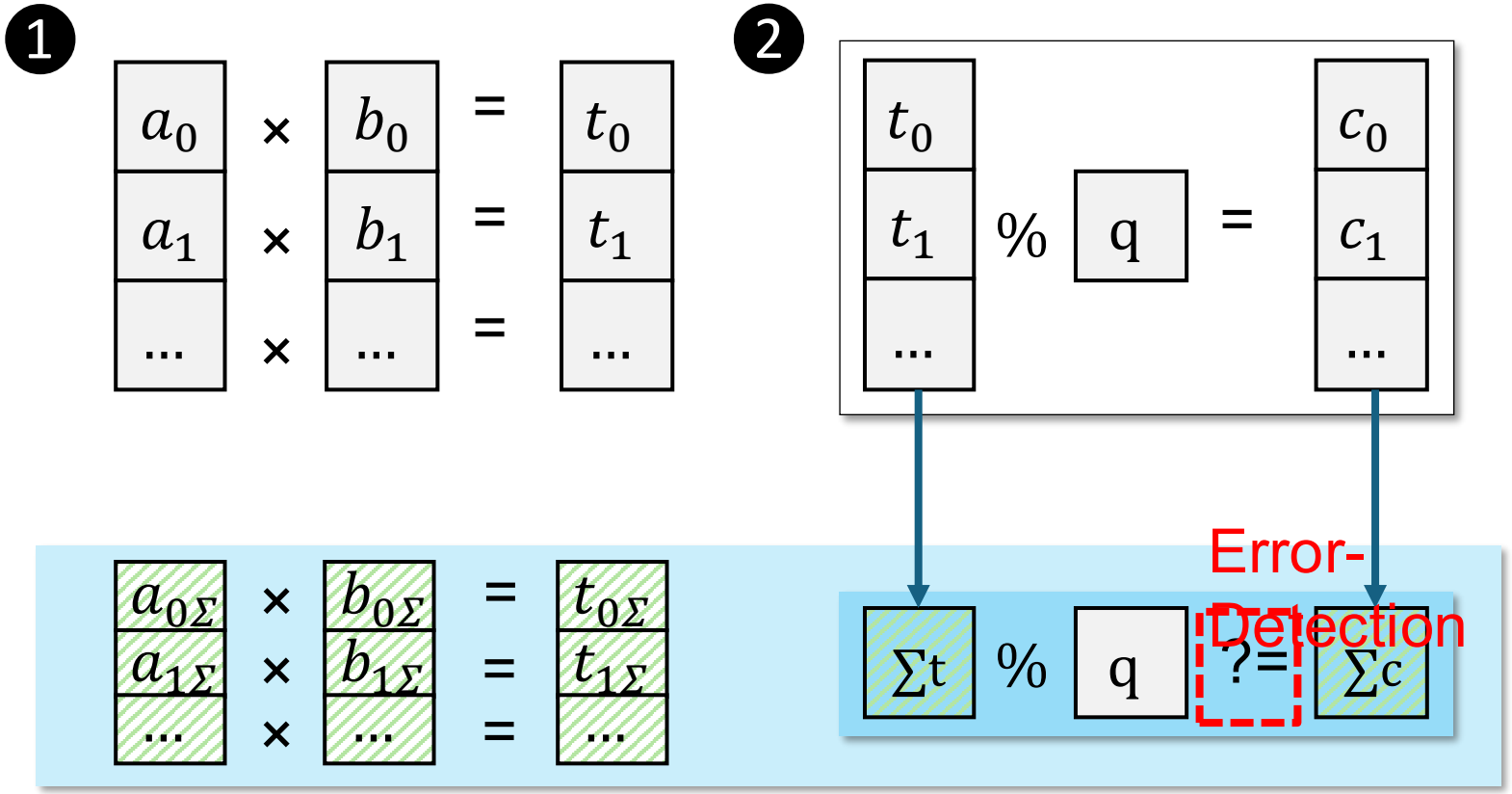
42=b0000 0000 0010 1010

42_{Σ}=b(0000 + 0000 + 0010 + 1010)
=b1100
=12

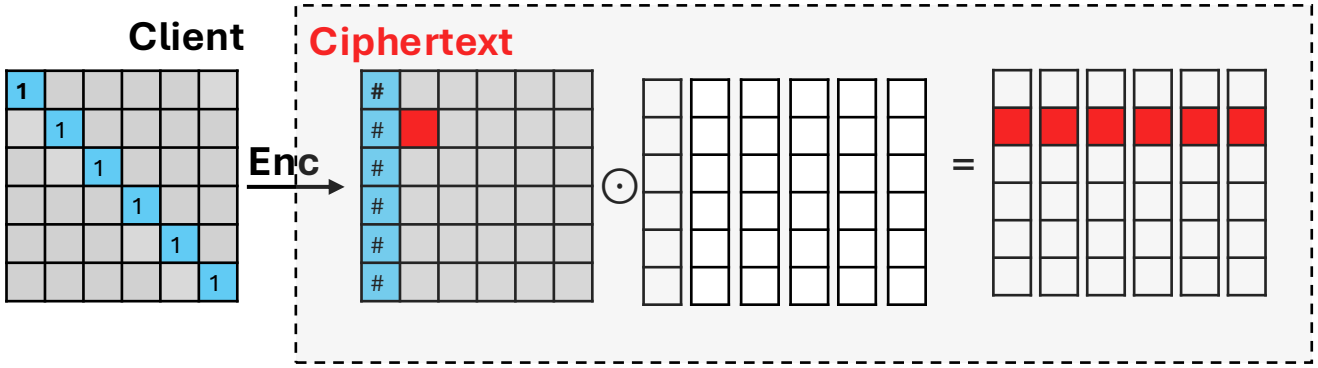
Elementwise Operation (e.g. EvkMult)



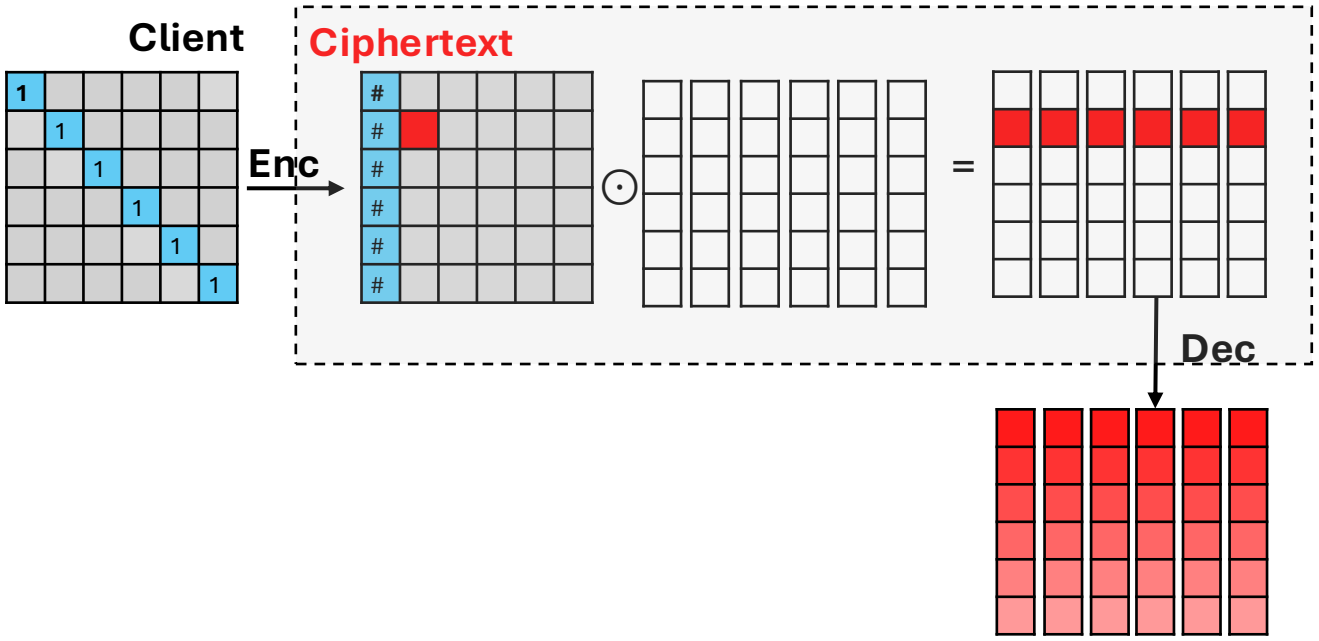
Elementwise Operation (e.g. EvkMult)



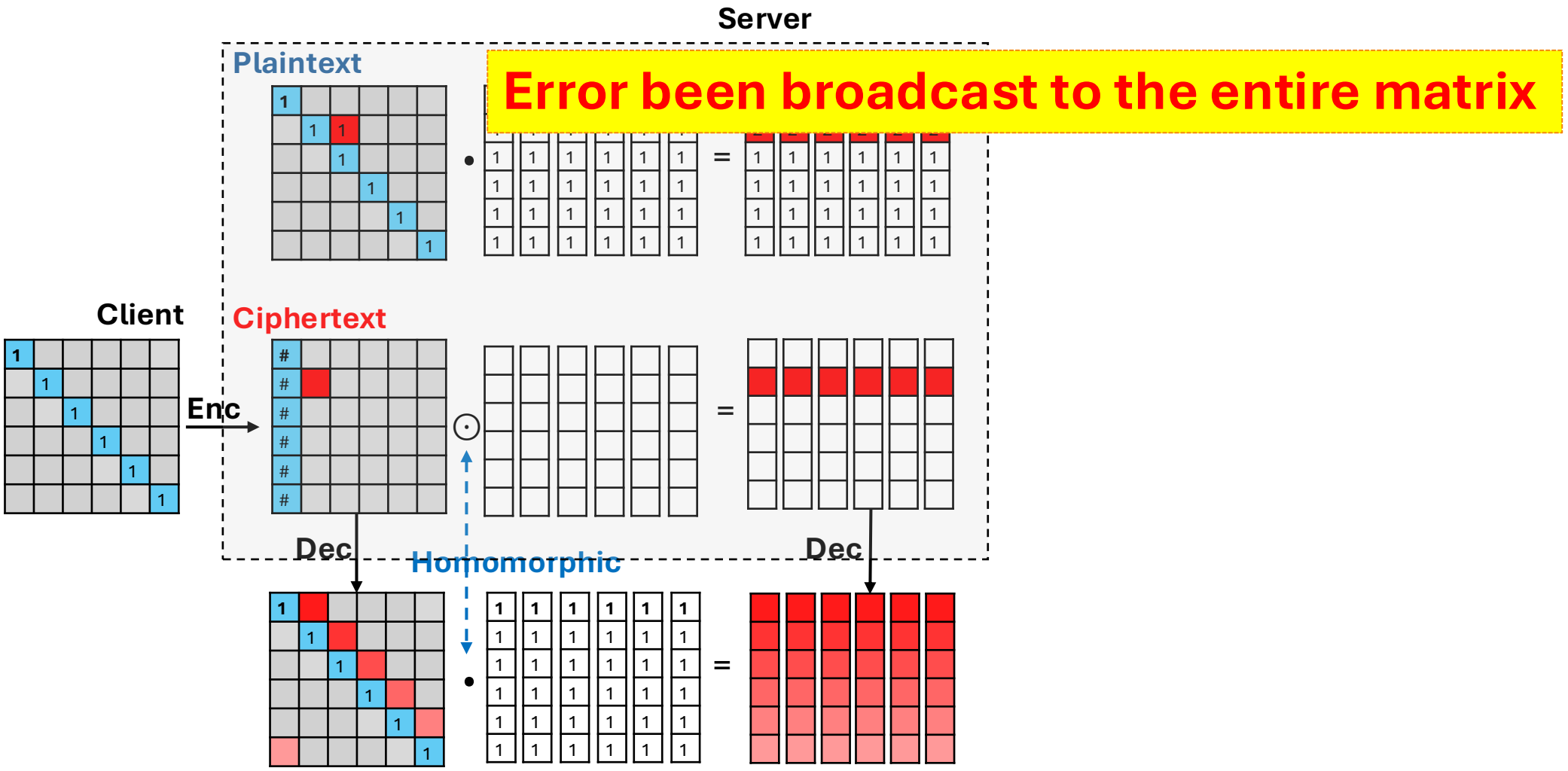
Algorithm-SIMD Packing



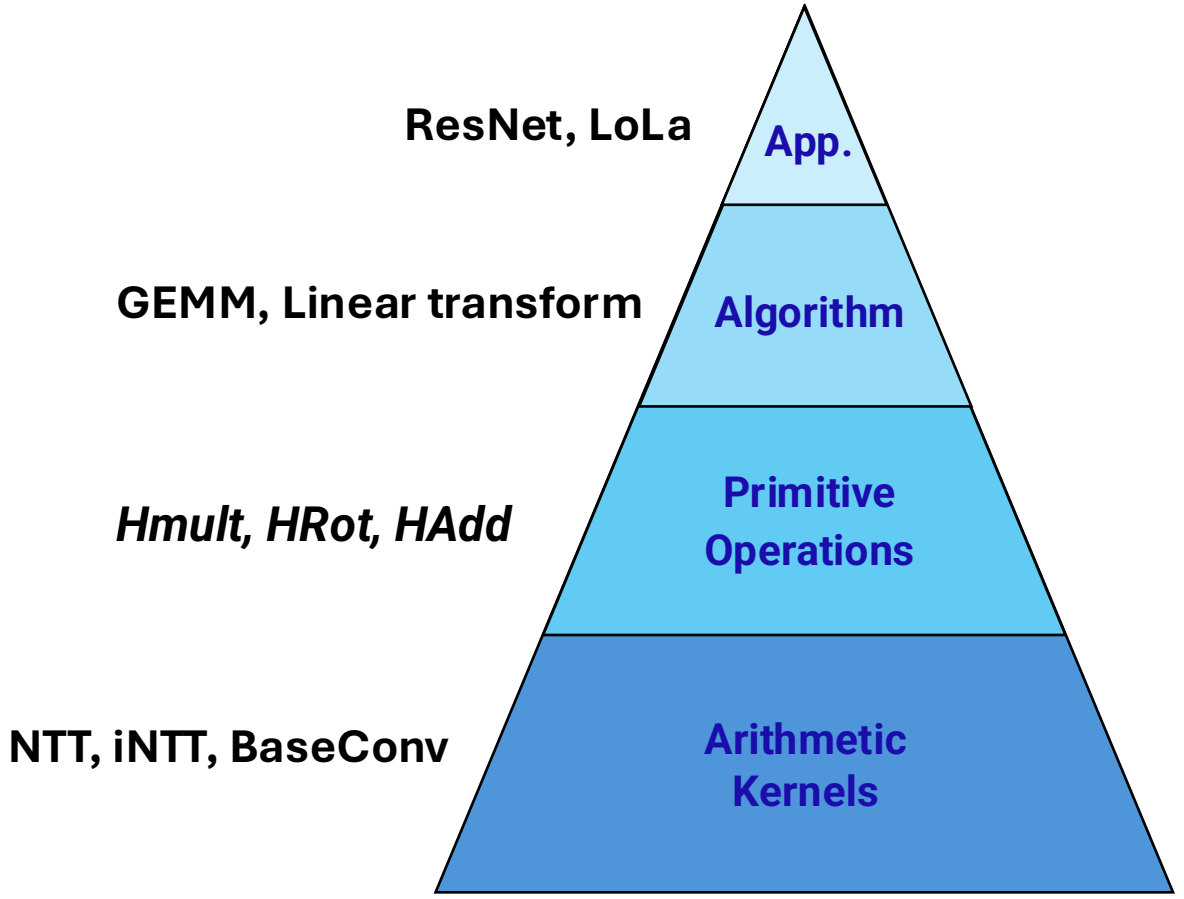
Algorithm-SIMD Packing



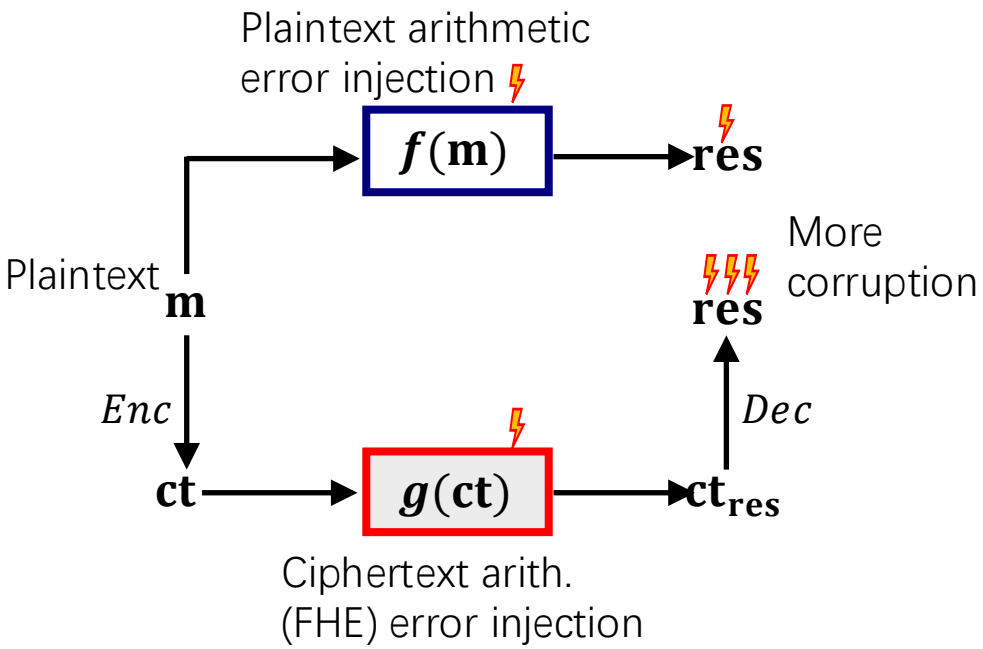
Algorithm-SIMD Packing

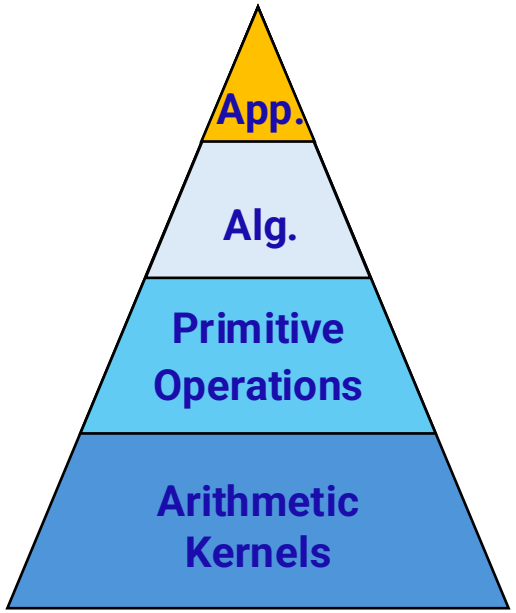


Operations

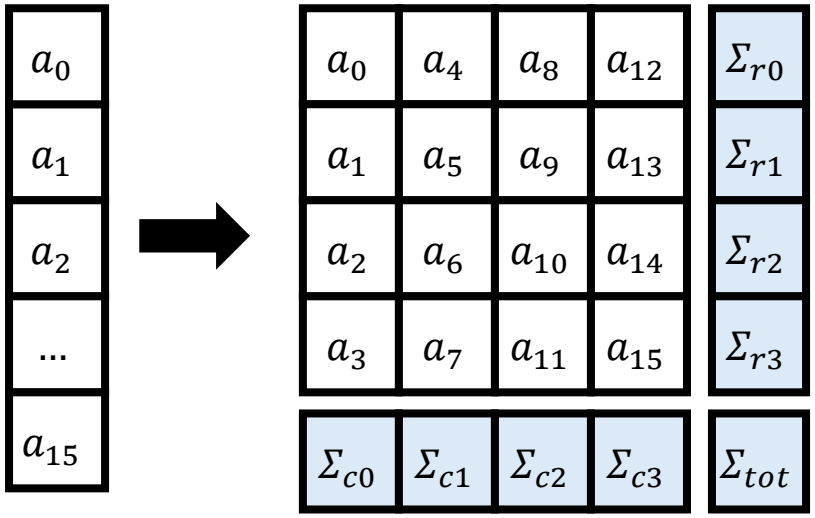

ResNet, LoLa
App.
GEMM, Linear transform
Algorithm
Hmult, HRot, HAdd
Primitive Operations
NTT, iNTT, BaseConv
Arithmetic Kernels

Application



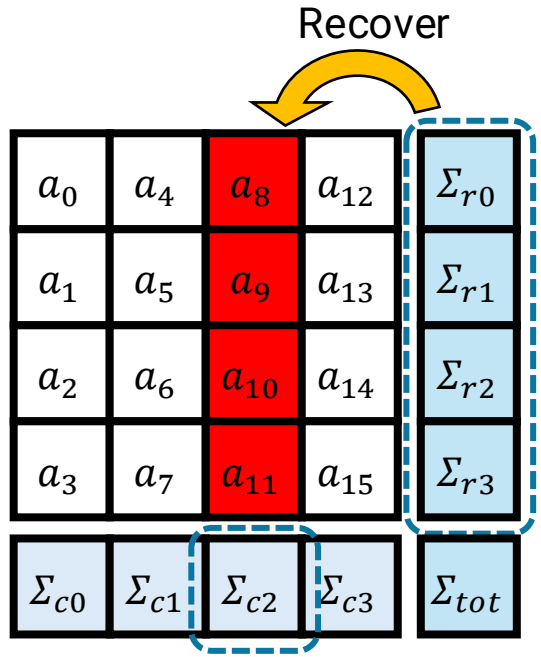


Storage Protection(e.g. coefficient)



Σ_{tot}: total Checksum

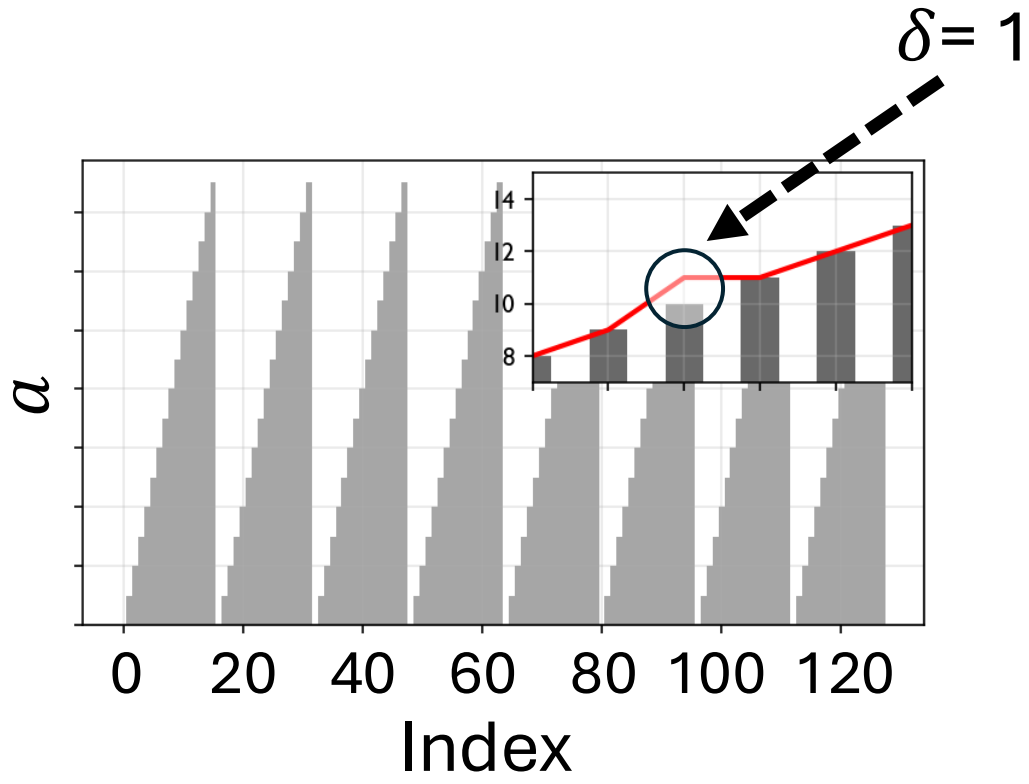
$\Sigma_{r,}\ \Sigma_c$: row and col-wise Checksum



Detect

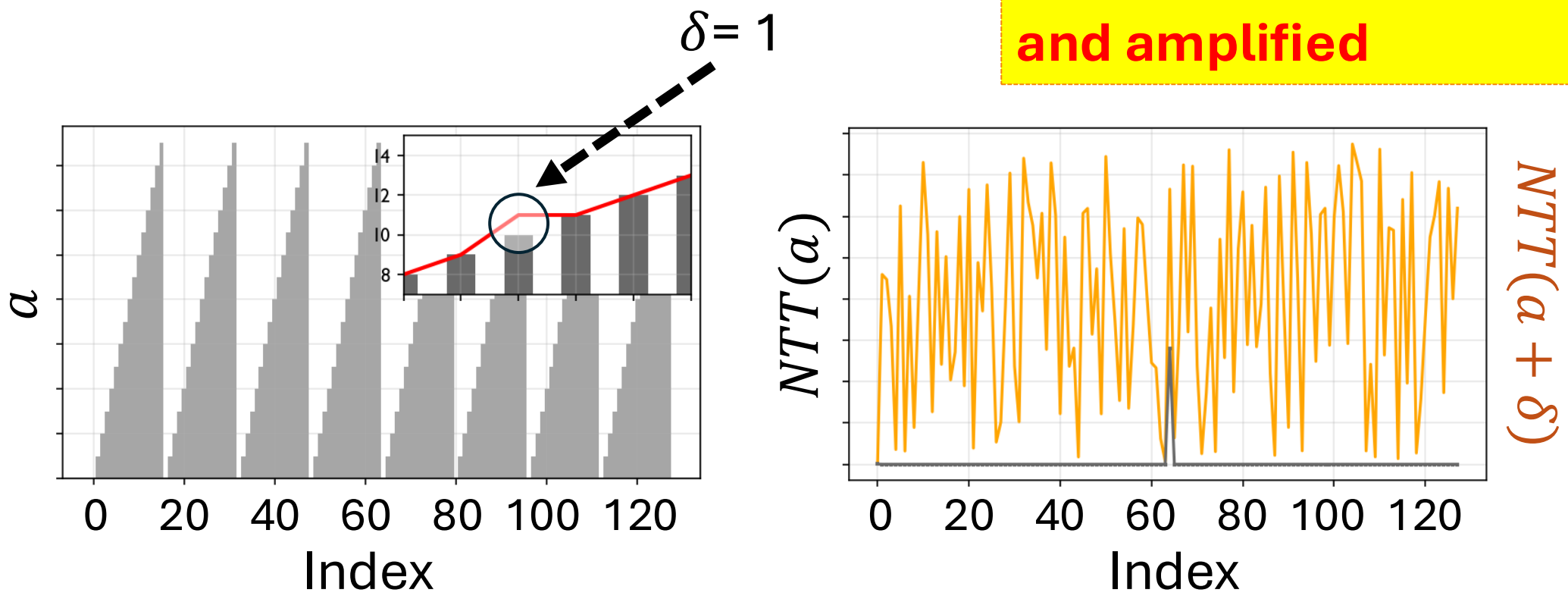
$\Sigma_{c2} \neq a_8 + a_9 + a_{10} + a_{11}$

Arithmetic Kernel-(i)NTT



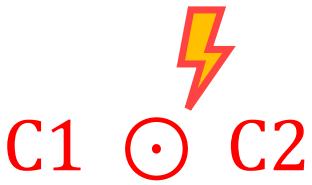
Arithmetic Kernel-(i)NTT

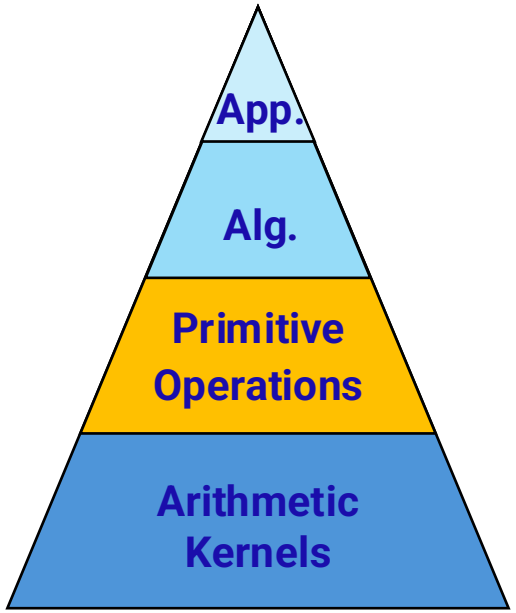
Error been broadcasted and amplified



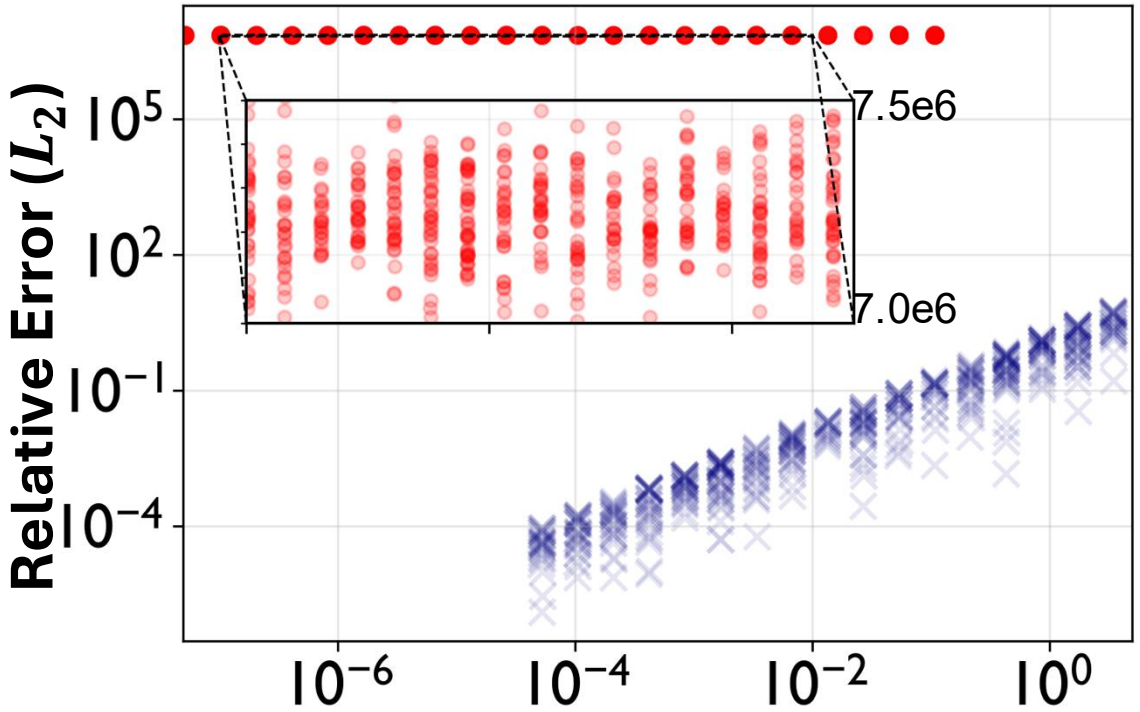
Primitive Operations-Hadamard Product

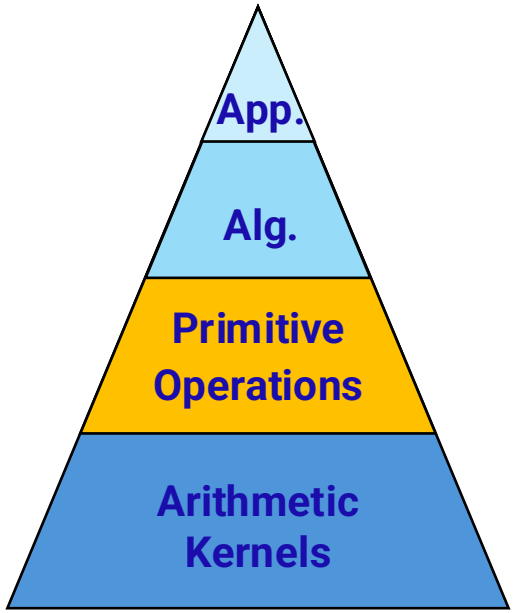
Versus



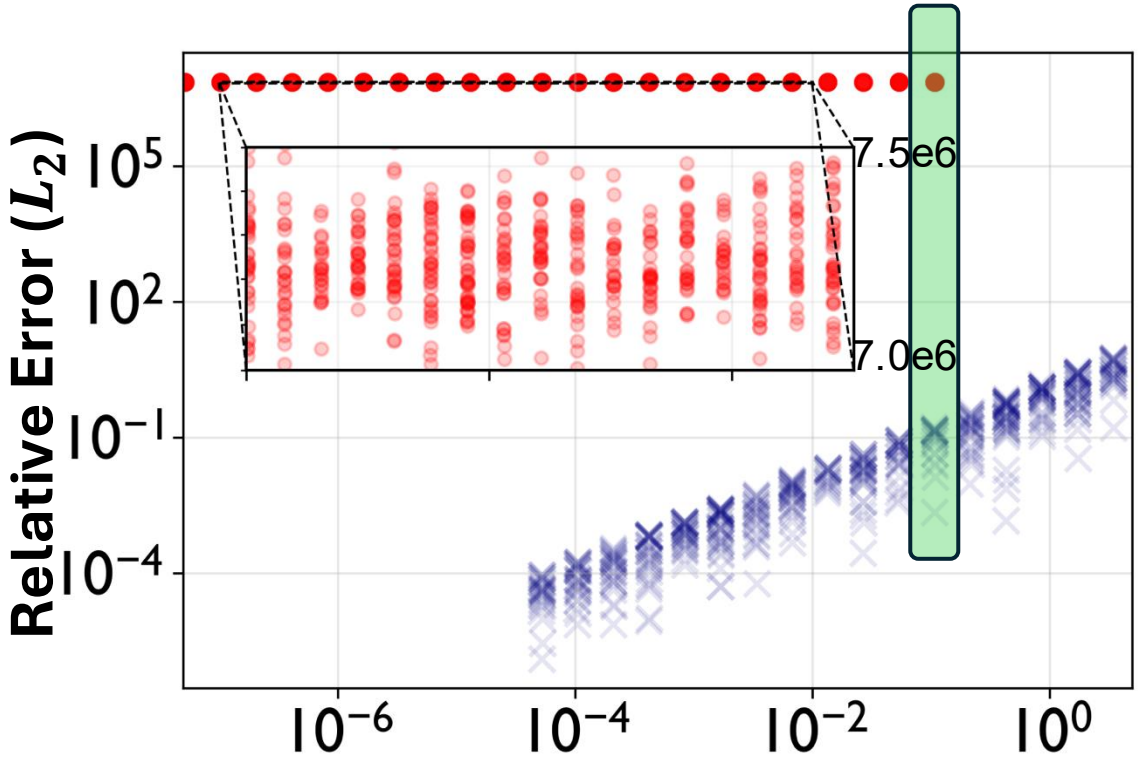


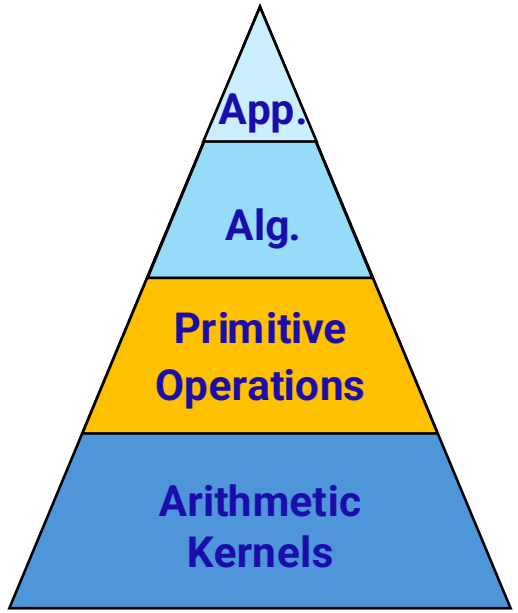
Primitive Operations-Hadamard Product



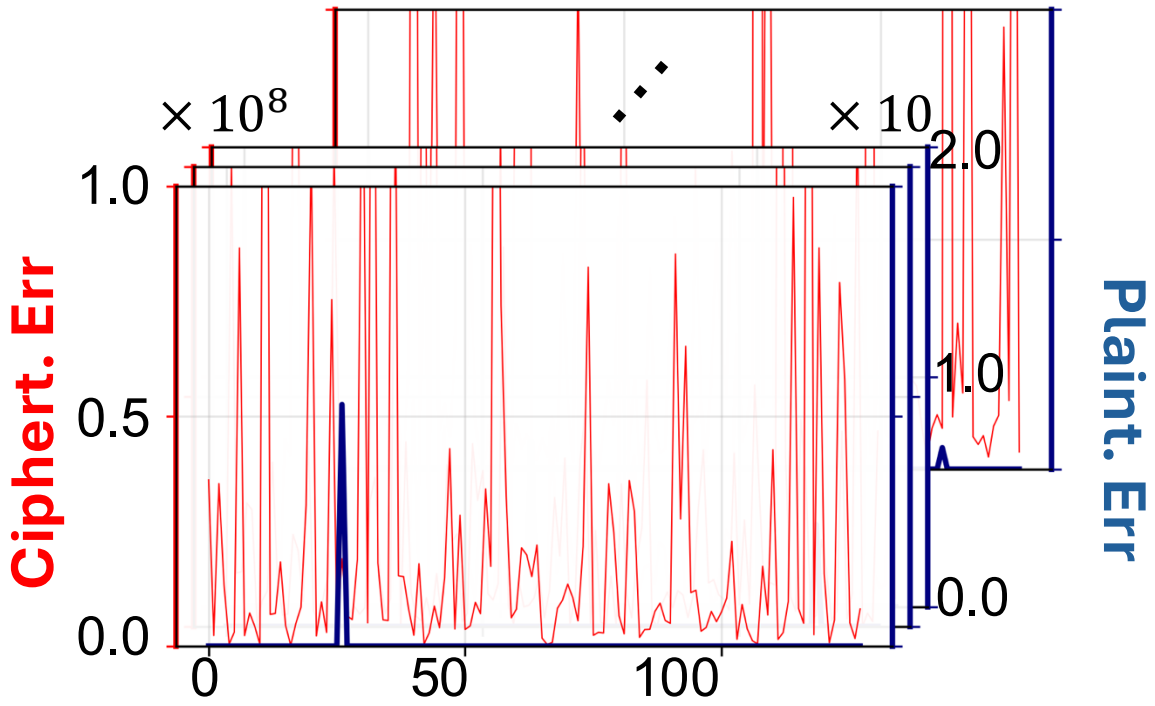


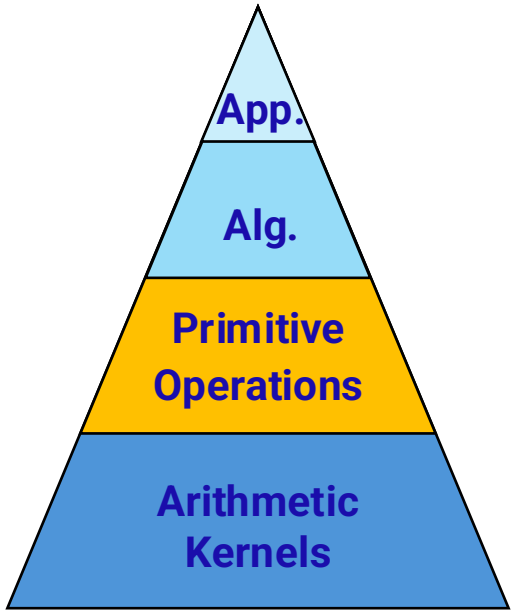
Primitive Operations-Hadamard Product



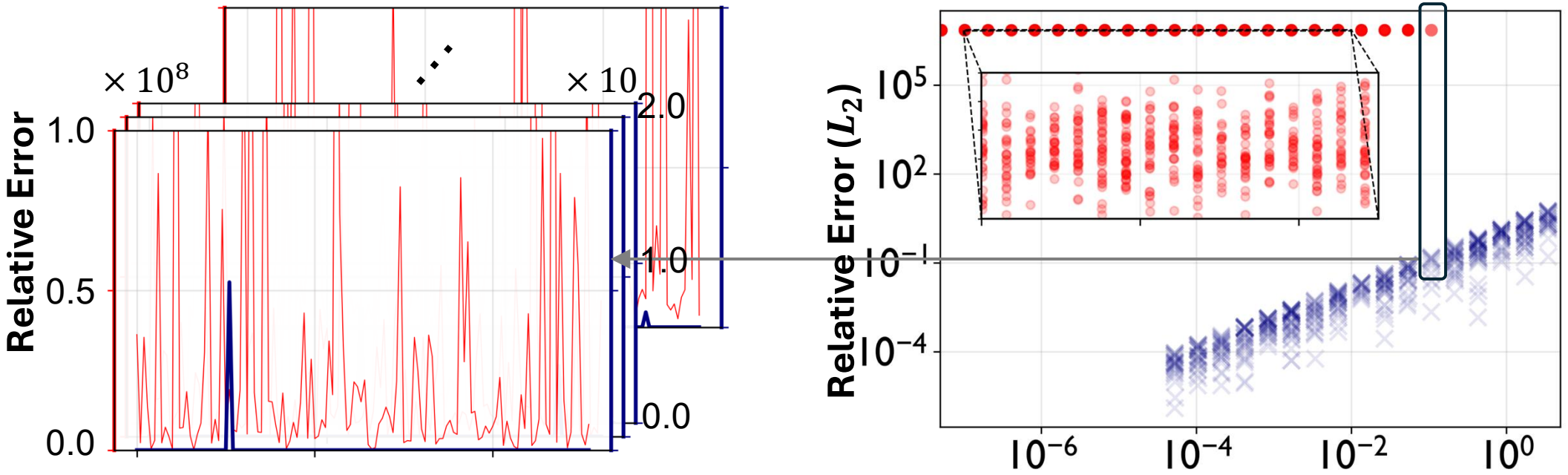


Primitive Operations-Hadamard Product



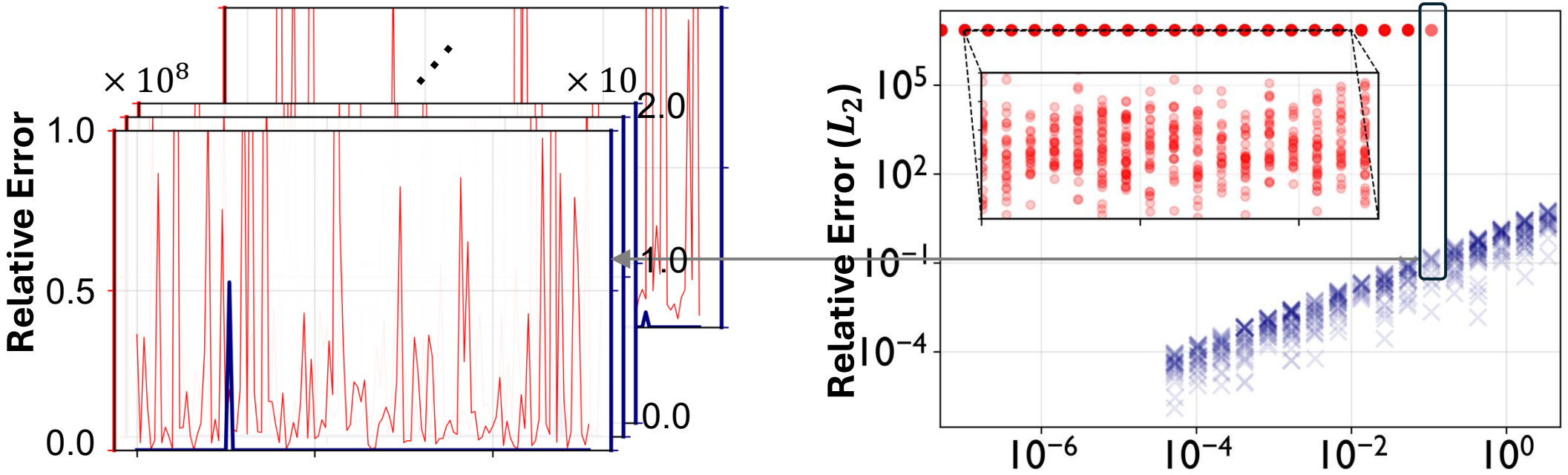


Primitive Operations-Hadamard Product



ASPLOS

Primitive Operations-Hadamard Product

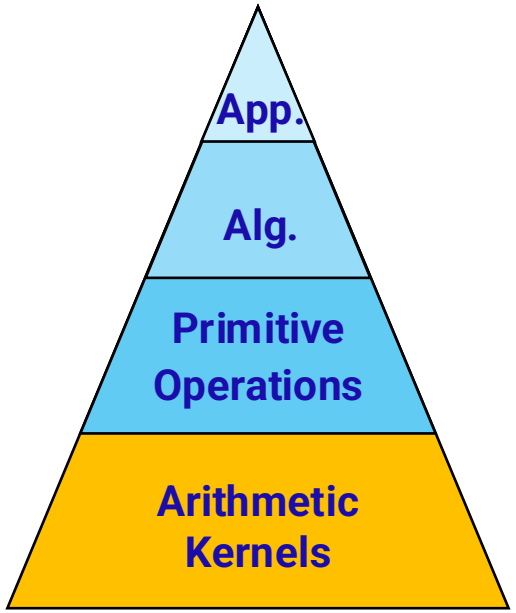


Error been broadcast to all elements

ASPLOS

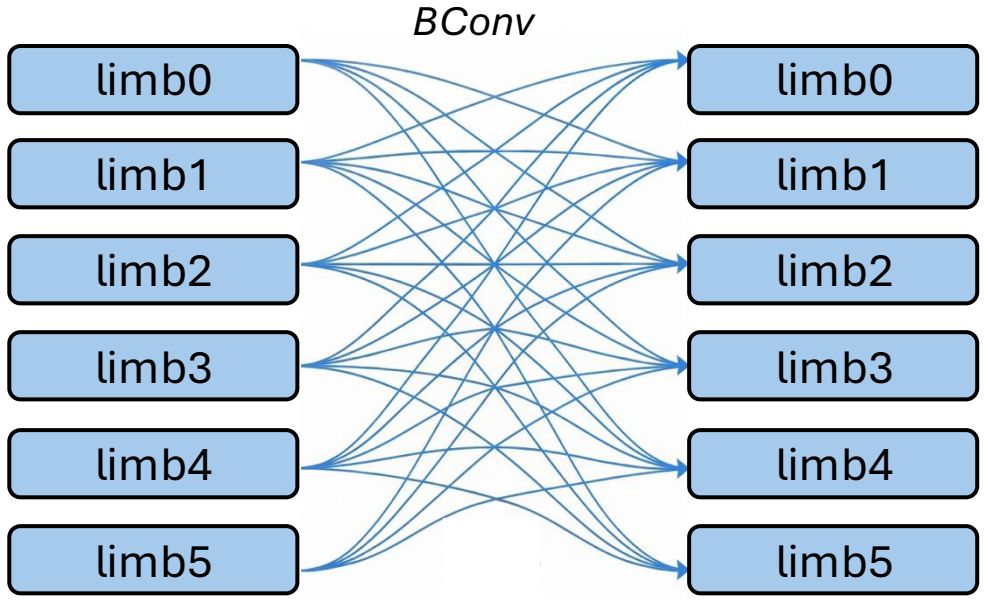
Arithmetic Kernel-BaseConversion

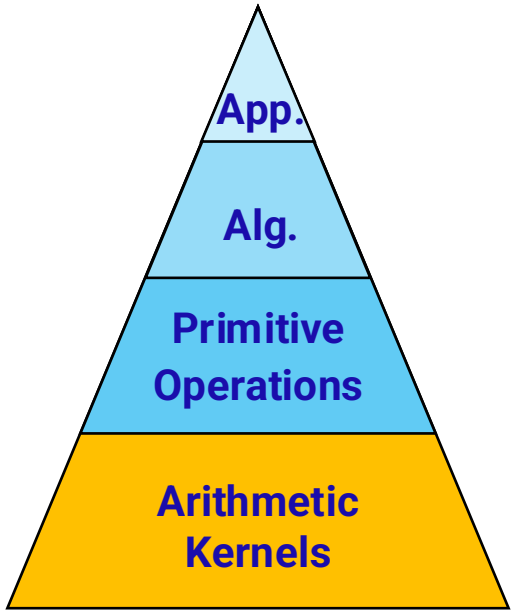
$$[\mathbf{a}]_{\mathcal{B}} = FastBaseConv_{\{\mathcal{C}\to\mathcal{B}\}}([\mathbf{a}]_{\mathcal{C}}) = \sum_{i=0}^{L}\left([\mathbf{a}]_{q_i}\cdot\frac{q_i}{Q}mod q_i\right)\cdot\frac{Q}{q_i}mod p_j$$



Arithmetic Kernel-BaseConversion

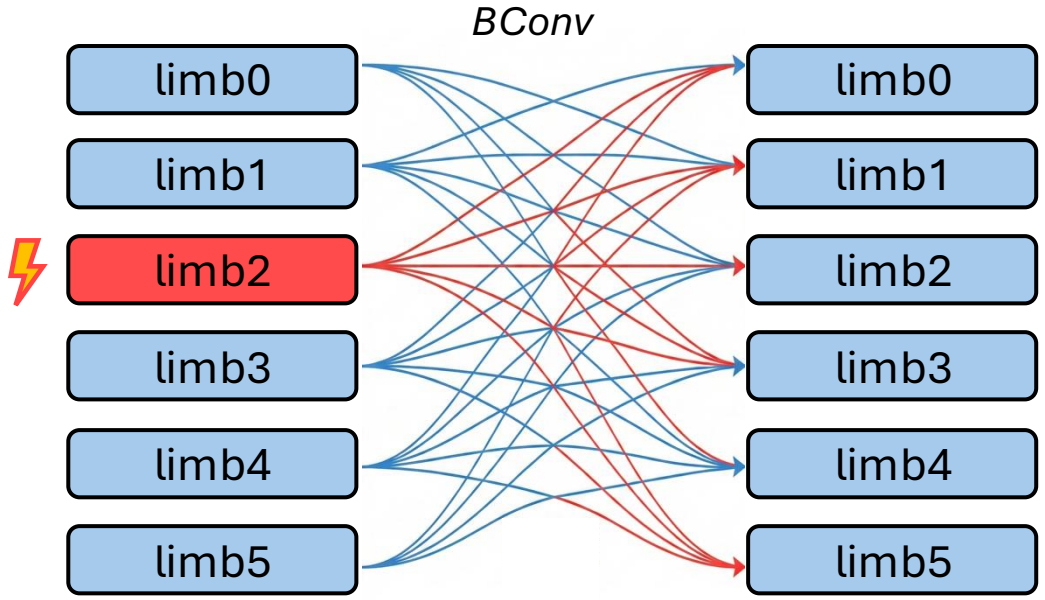
$$[\mathbf{a}]_{\mathcal{B}} = FastBaseConv_{\{\mathcal{C}\to\mathcal{B}\}}([\mathbf{a}]_{\mathcal{C}}) = \sum_{i=0}^{L}\left([\mathbf{a}]_{q_i}\cdot\frac{q_i}{Q}mod q_i\right)\cdot\frac{Q}{q_i}mod p_j$$

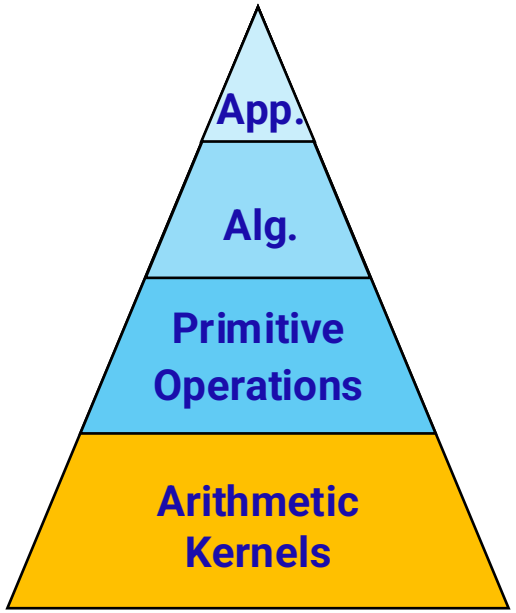




Arithmetic Kernel-BaseConversion

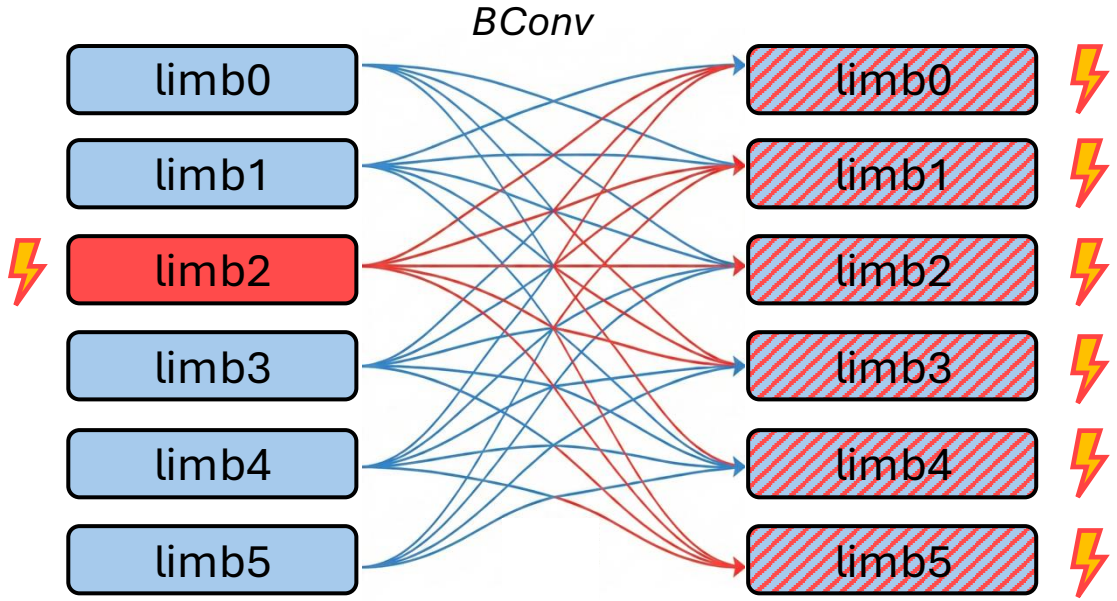
$$[\mathbf{a}]_{\mathcal{B}} = FastBaseConv_{\{\mathcal{C}\to\mathcal{B}\}}([\mathbf{a}]_{\mathcal{C}}) = \sum_{i=0}^{L}\left([\mathbf{a}]_{q_i}\cdot\frac{q_i}{Q}mod q_i\right)\cdot\frac{Q}{q_i}mod p_j$$

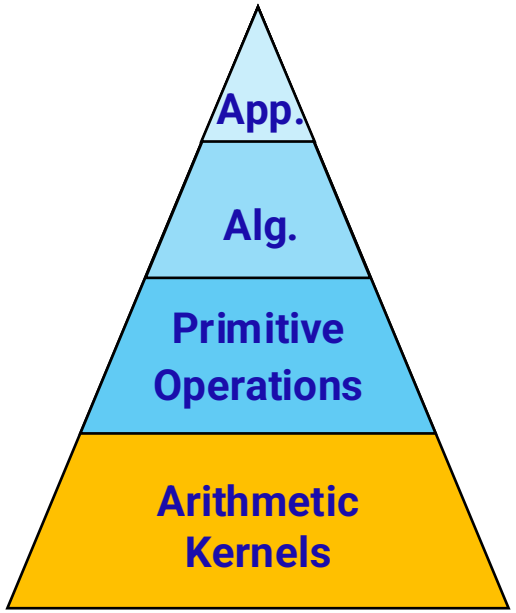




Arithmetic Kernel-BaseConversion

$$[\mathbf{a}]_{\mathcal{B}} = FastBaseConv_{\{\mathcal{C}\to\mathcal{B}\}}([\mathbf{a}]_{\mathcal{C}}) = \sum_{i=0}^{L}\left([\mathbf{a}]_{q_i}\cdot\frac{q_i}{Q}mod q_i\right)\cdot\frac{Q}{q_i}mod p_j$$





Error been broadcast to all limbs

Elementwise Operation -- Modulus

Algorithm 1 Resilient Barrett Reduction ($\mathbf{c} \equiv \mathbf{t} \bmod q$)

Notation: modulus q; $\mathbf{t} \in \mathbb{Z}_{2^{2K}}^{N}$

Precomputed: $K = \lceil \log_2 q \rceil$, $\mu = \lfloor 2^{2K}/q \rfloor$

Input: $\mathbf{t}$, t_{tot}; **Output:** $\mathbf{c}$, c_{tot},

1: **procedure** RELIA-BARRETT-REDUCE($\mathbf{t}, t_{tot}$)
2: **for** each t_i in $\mathbf{t}$ **do**
3: $s_i \leftarrow (t_i \times \mu) \gg 2K$
4: $c_i \leftarrow t_i - s_i \times q$
5: **if** $c_i \geq q$ **then** $c_i \leftarrow c_i - q$ **end if**
6: **assert** $0 \leq c_i < q$ //range check
7: **end for**
8: $c_{tot} \leftarrow t_{tot} \pmod q$
9: **assert** $c_{tot} == \sum \mathbf{c}$ //verify inter-elem sum
10: **return** $\mathbf{c}, c_{tot}$
11: **end procedure**