Efficient Ranking, Order Statistics, and Sorting under CKKS

Federico Mazzone[1], Maarten Everts[1,2], Florian Hahn[1], Andreas Peter[3]

[1] University of Twente (Netherlands) - [2] Linksight (Netherlands)

[3] Carl von Ossietzky Universität Oldenburg (Germany)

34th USENIX Security Symposium

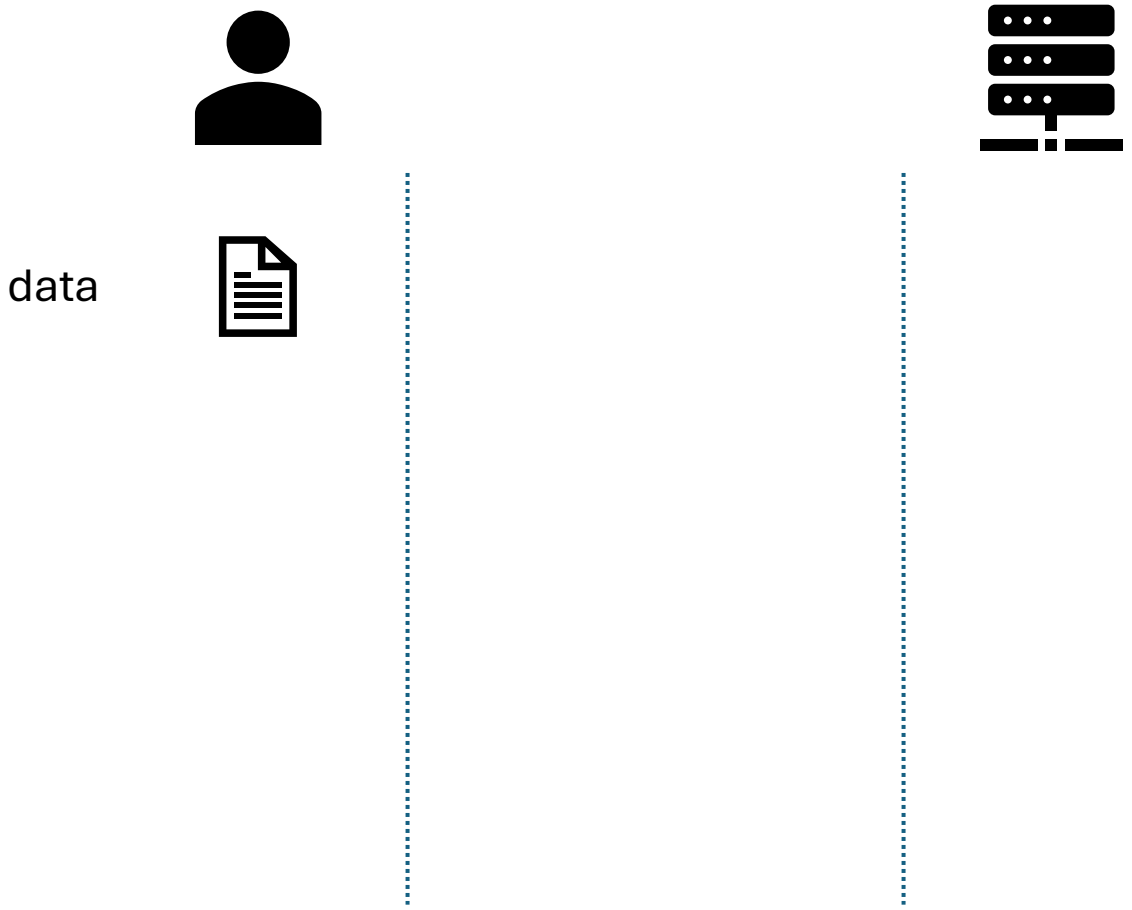
August 13-15, 2025

Seattle, WA, USA

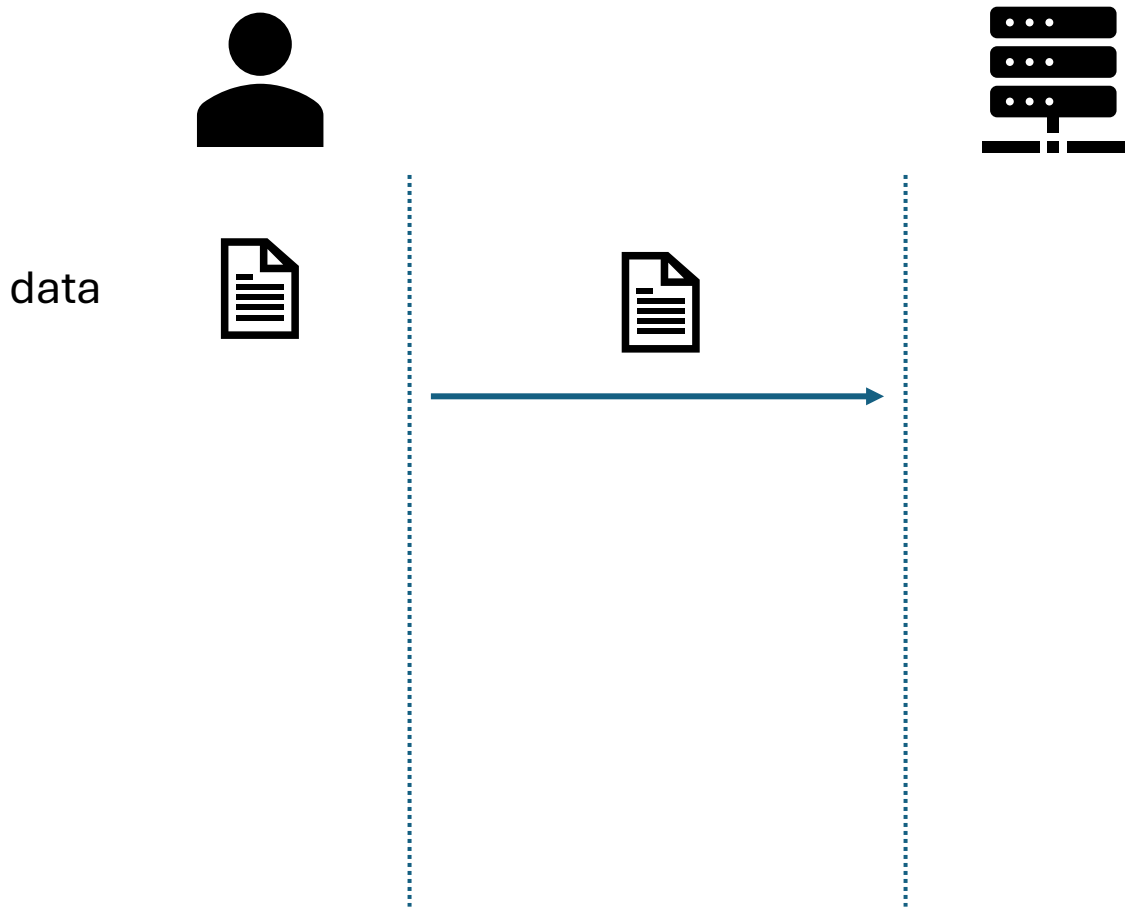
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 965315.

This work was supported by the Netherlands Organization for Scientific Research under NWO:SHARE project [CS.011].

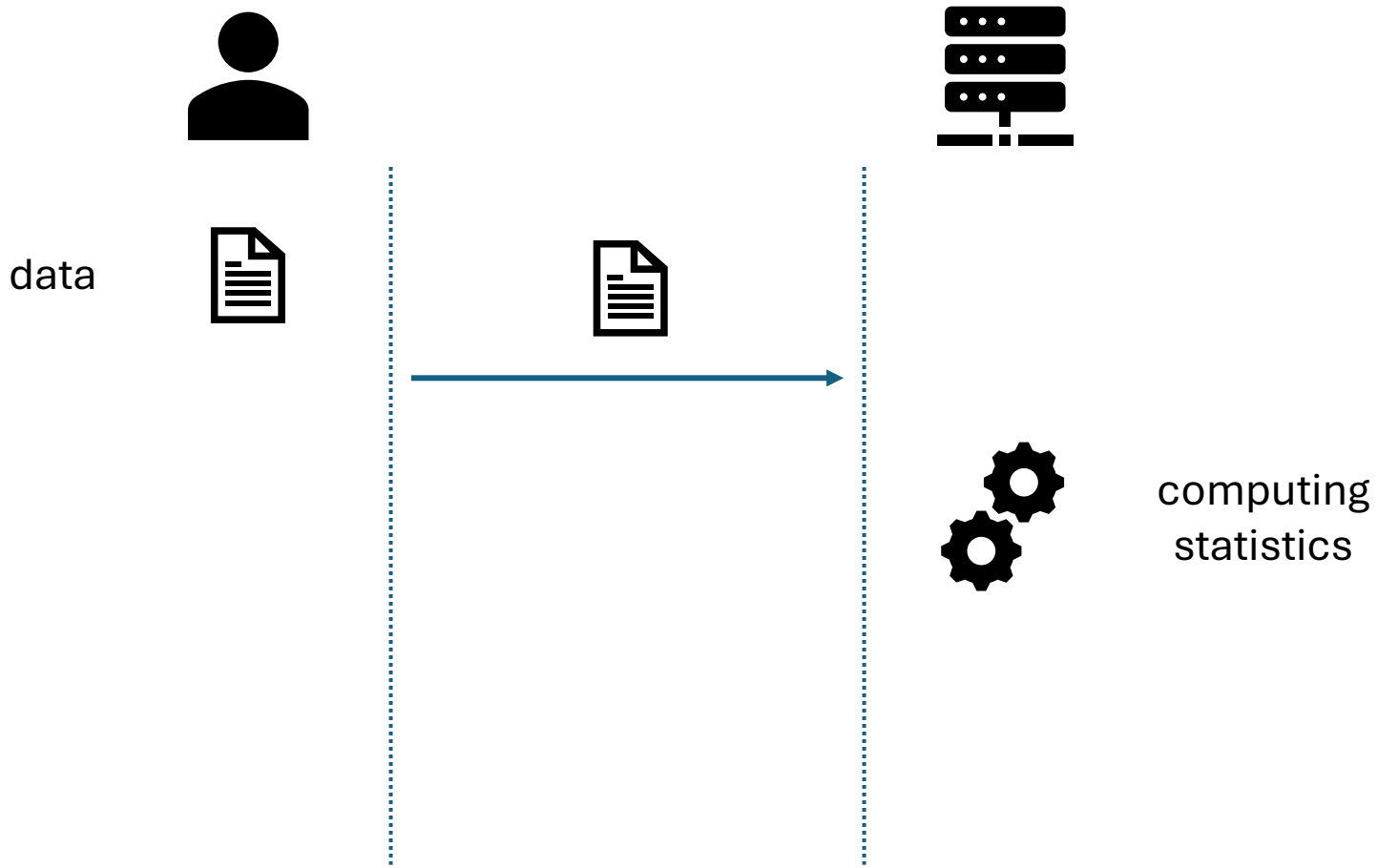
Scenario: Secure Cloud Computing



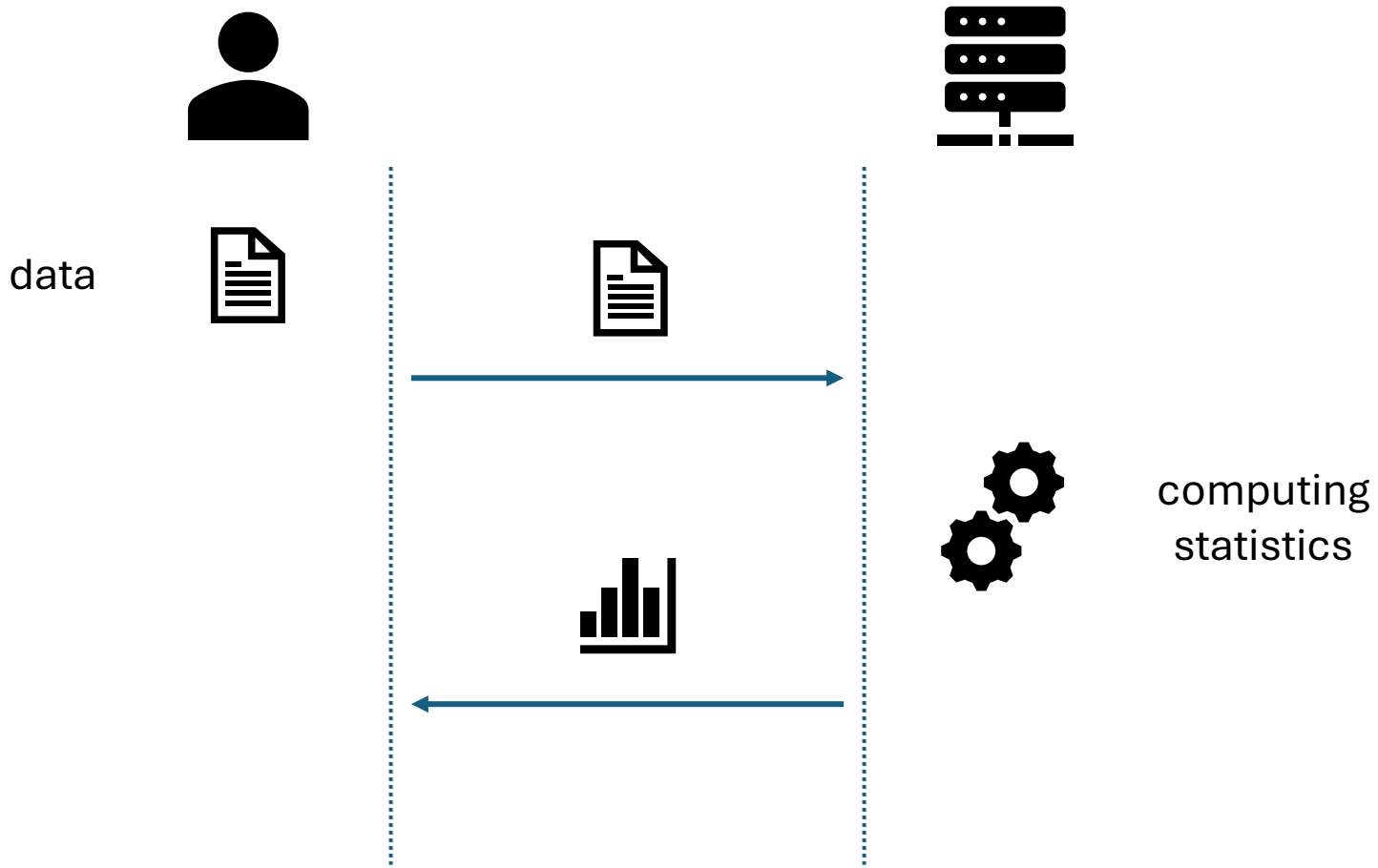
Scenario: Secure Cloud Computing



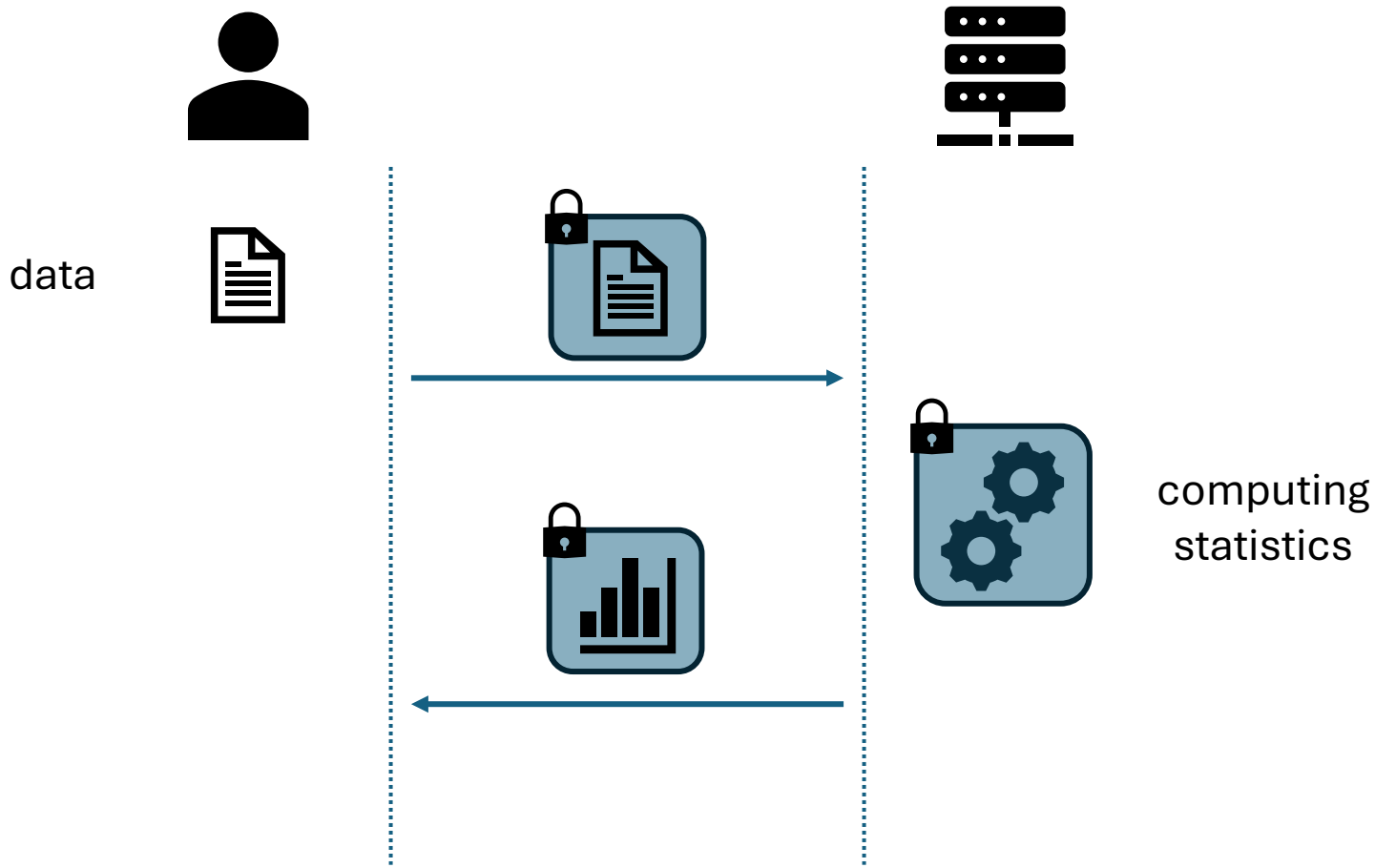
Scenario: Secure Cloud Computing



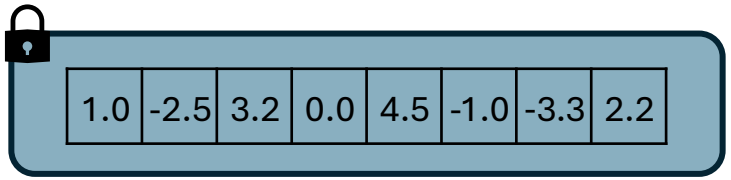
Scenario: Secure Cloud Computing



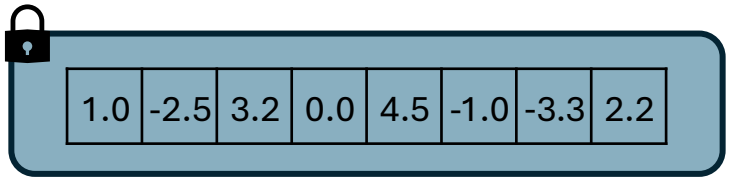
Scenario: Secure Cloud Computing



CKKS: Native SIMD Operations



CKKS: Native SIMD Operations

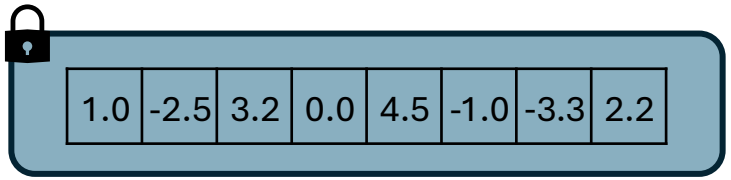


1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

-1.0	2.0	1.8	-3.0	-0.5	1.0	0.3	0.8

0.0	-0.5	5.0	-3.0	4.0	0.0	-3.0	3.0

CKKS: Native SIMD Operations



1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

-1.0	2.0	1.8	-3.0	-0.5	1.0	0.3	0.8

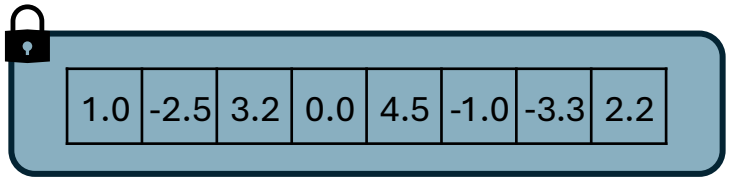
0.0	-0.5	5.0	-3.0	4.0	0.0	-3.0	3.0

1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

-1.0	2.0	1.8	-3.0	-0.5	1.0	0.3	0.8

-1.0	-5.0	5.76	0.0	-2.3	-1.0	-1.0	1.8

CKKS: Native SIMD Operations



1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

+

-1.0	2.0	1.8	-3.0	-0.5	1.0	0.3	0.8

=

0.0	-0.5	5.0	-3.0	4.0	0.0	-3.0	3.0

1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

×

-1.0	2.0	1.8	-3.0	-0.5	1.0	0.3	0.8

=

-1.0	-5.0	5.76	0.0	-2.3	-1.0	-1.0	1.8

1.0	-2.5	3.2	0.0	4.5	-1.0	-3.3	2.2

≪

3

=

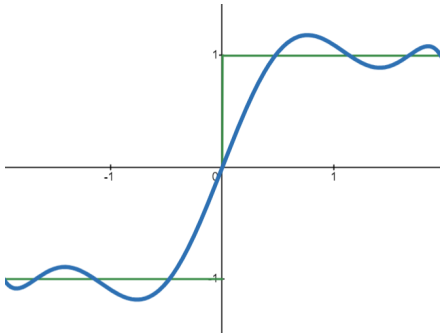
0.0	4.5	-1.0	-3.3	2.2	1.0	-2.5	3.2

CKKS: Value Comparison ($x > y$)

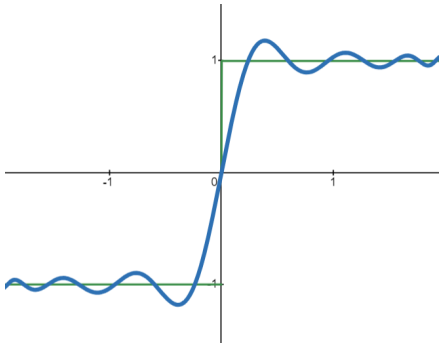
$$x > y = \frac{\text{sign}(x - y) + 1}{2} = \begin{cases} 1, & x > y \\ 0.5, & x = y \\ 0, & x < y \end{cases}$$

CKKS: Value Comparison ($x > y$)

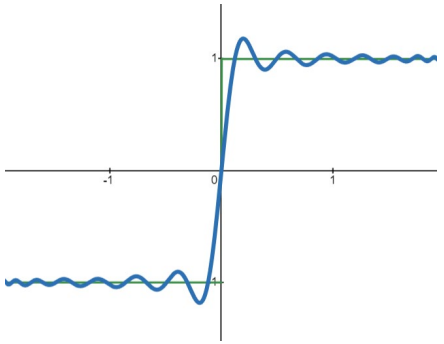
$$x > y = \frac{\text{sign}(x - y) + 1}{2} = \begin{cases} 1, & x > y \\ 0.5, & x = y \\ 0, & x < y \end{cases}$$



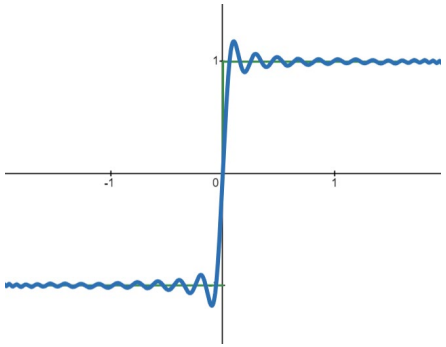
degree 8



degree 16



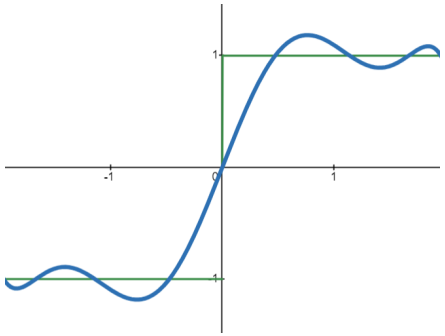
degree 32



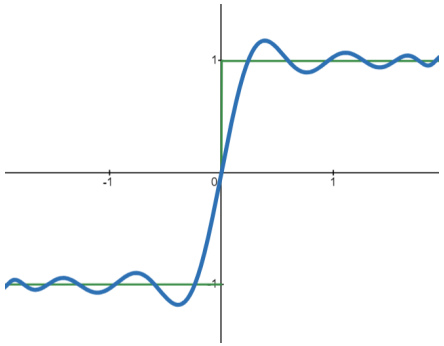
degree 64

CKKS: Value Comparison ($x > y$)

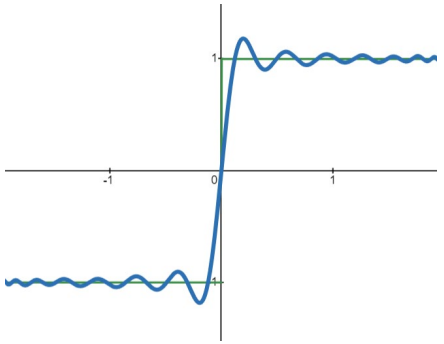
$$x > y = \frac{\text{sign}(x - y) + 1}{2} = \begin{cases} 1, & x > y \\ 0.5, & x = y \\ 0, & x < y \end{cases}$$



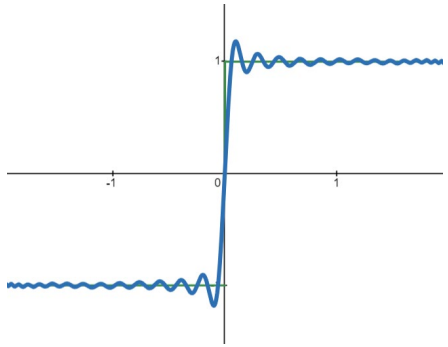
degree 8



degree 16



degree 32



degree 64

EXPENSIVE!

Computational Bottleneck

This operation is a bottleneck in comparison-heavy algorithms:

- Ranking
- Computing minimum/maximum
- Sorting

Computational Bottleneck

This operation is a bottleneck in comparison-heavy algorithms:

- Ranking
- Computing minimum/maximum
- Sorting

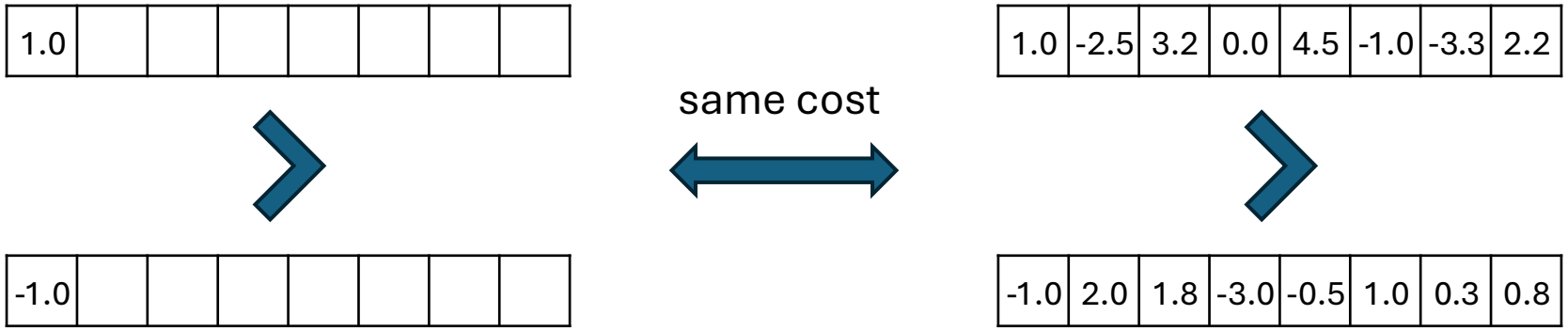
CHALLENGE: You can’t work with individual values!

Computational Bottleneck

This operation is a bottleneck in comparison-heavy algorithms:

- Ranking
- Computing minimum/maximum
- Sorting

CHALLENGE: You can't work with individual values!



Existing Solutions – Swap-Based Methods

Paper	Method	Comparison Depth	Complexity
Chatterjee et al. (Indocrypt 2013)	Bubble Sort, Insertion Sort	N^2	$O(N^2)$
Chatterjee et al. (IEEE TSC 2017)	Quick Sort	N^2	$O(N^2)$
Emmadi et al. (ICCCRI 2015)	Bitonic Sort, Odd-Even Merge Sort	$\log^2 N$	$O(N \log^2 N)$
Lu et al. (IEEE S&P 2021)	Bitonic Sort	$\log^2 N$	$O(N \log^2 N)$
Hong et al. (IEEE TIFS 2021)	k-Way Sorting Networks	$k \log_k^2 N$	$O\left(Nk \log_k^2 N\right)$

GOAL

To reduce the comparison depth of comparison-based algorithms under FHE.

GOAL

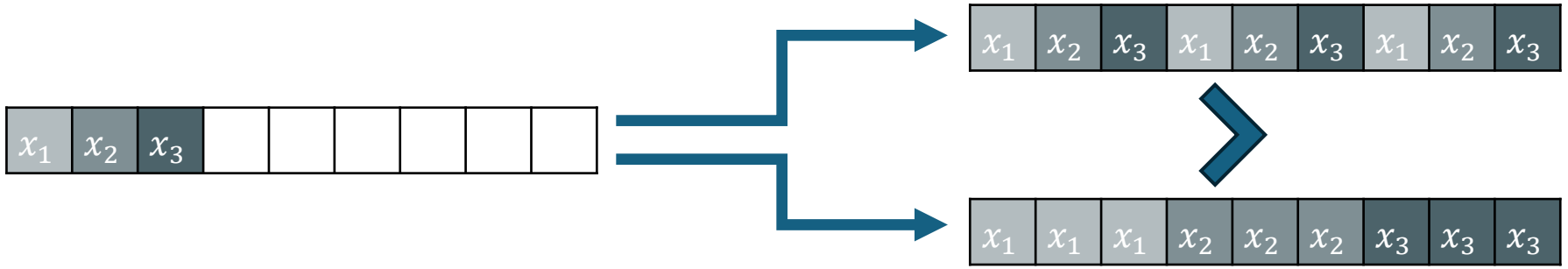
To reduce the comparison depth of comparison-based algorithms under FHE.

CONTRIBUTION

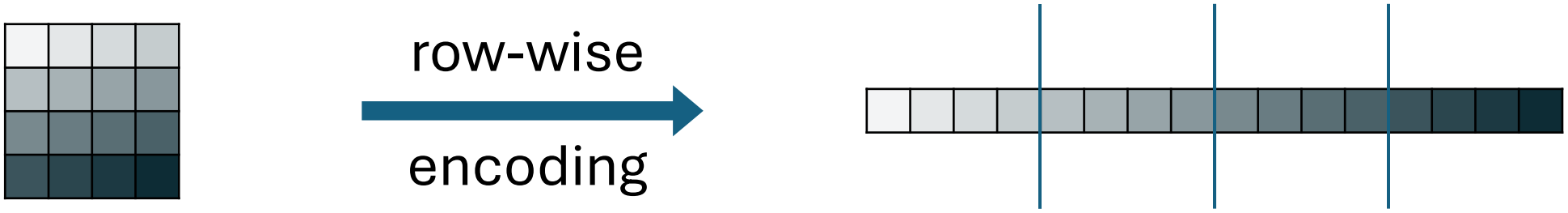
We design algorithms for ranking, order statistics, and sorting under CKKS that requires a comparison depth of at most 2.

Our Idea to Minimize Comparisons

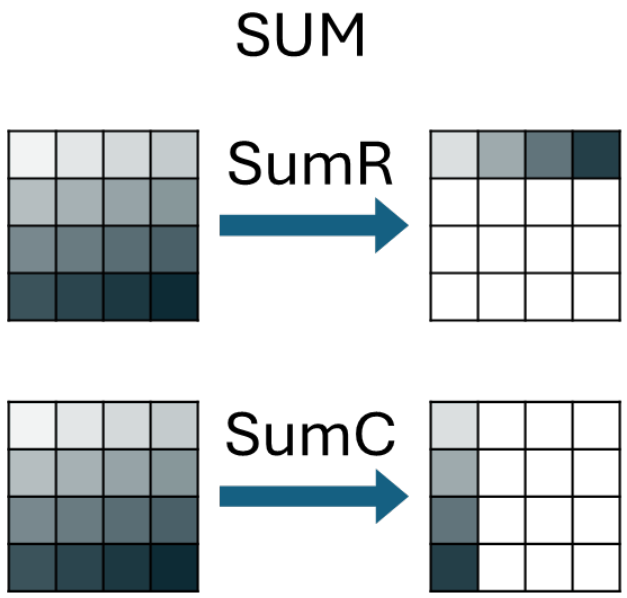
- Use SIMD to compare all elements at once.



Row-Wise Encoding of a Matrix

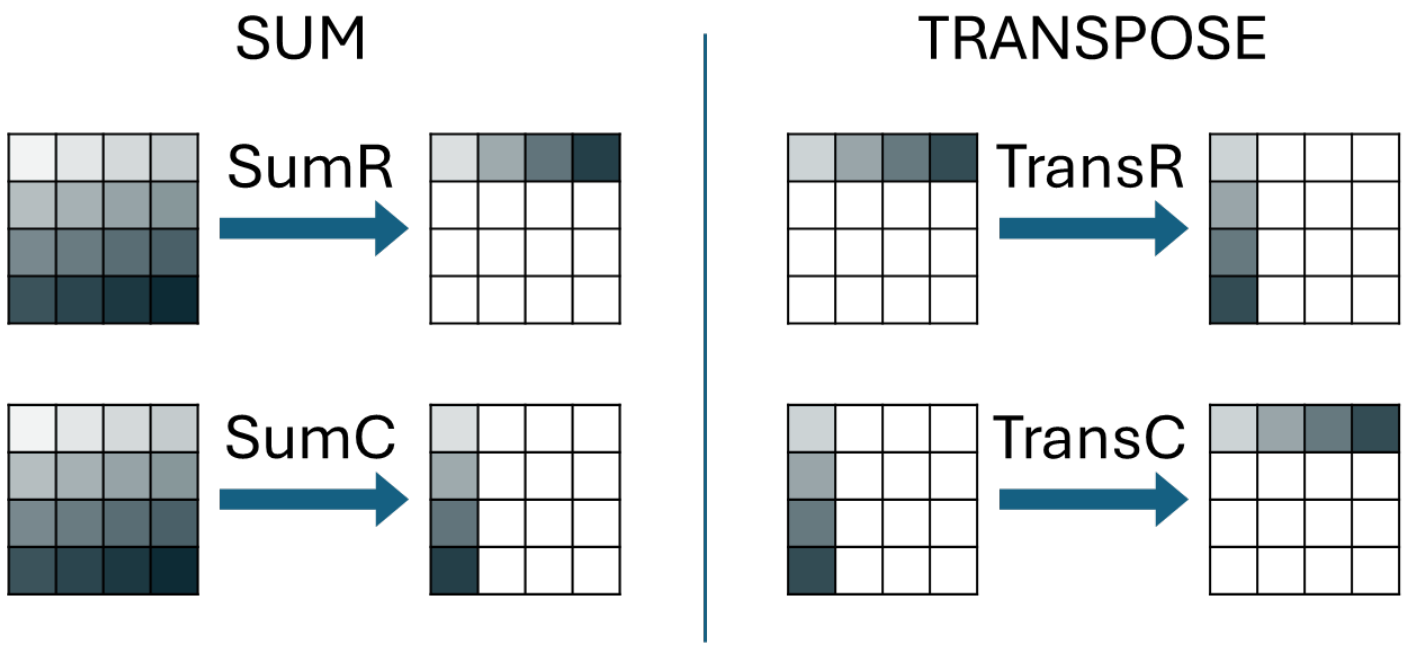


Matrix Operations



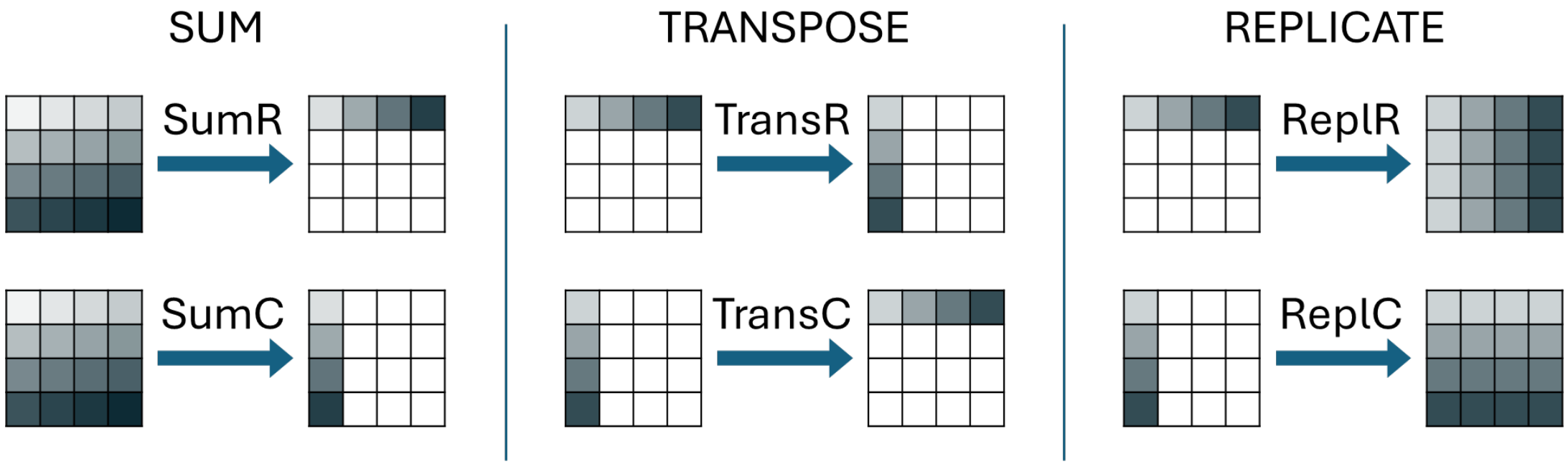
$\log N$ additions and rotations

Matrix Operations



$\log N$ additions and rotations

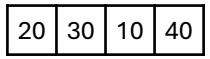
Matrix Operations



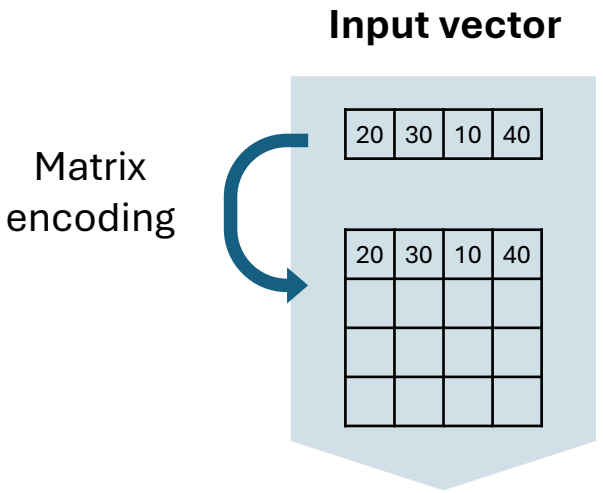
$\log N$ additions and rotations

Ranking

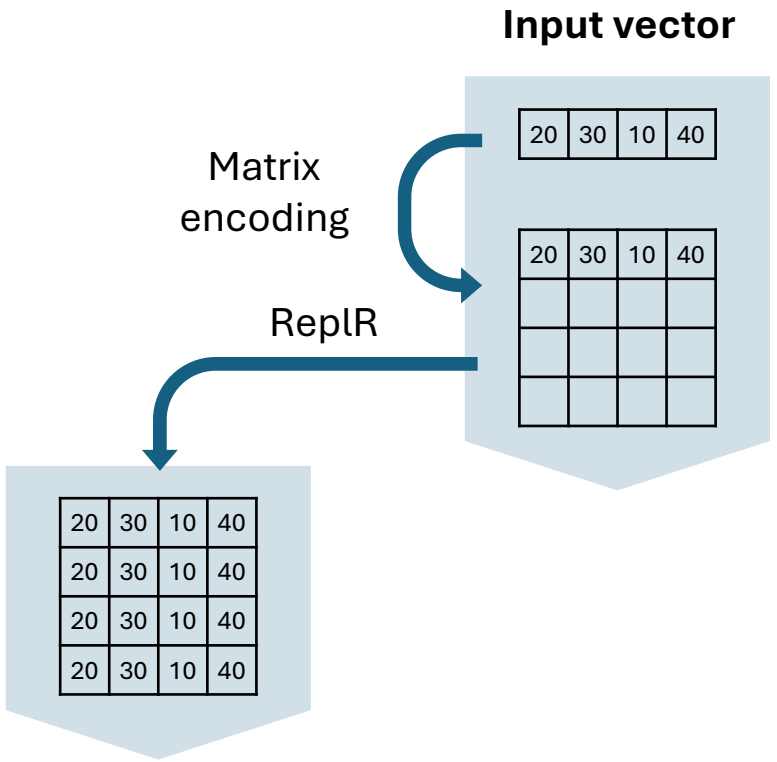
Input vector



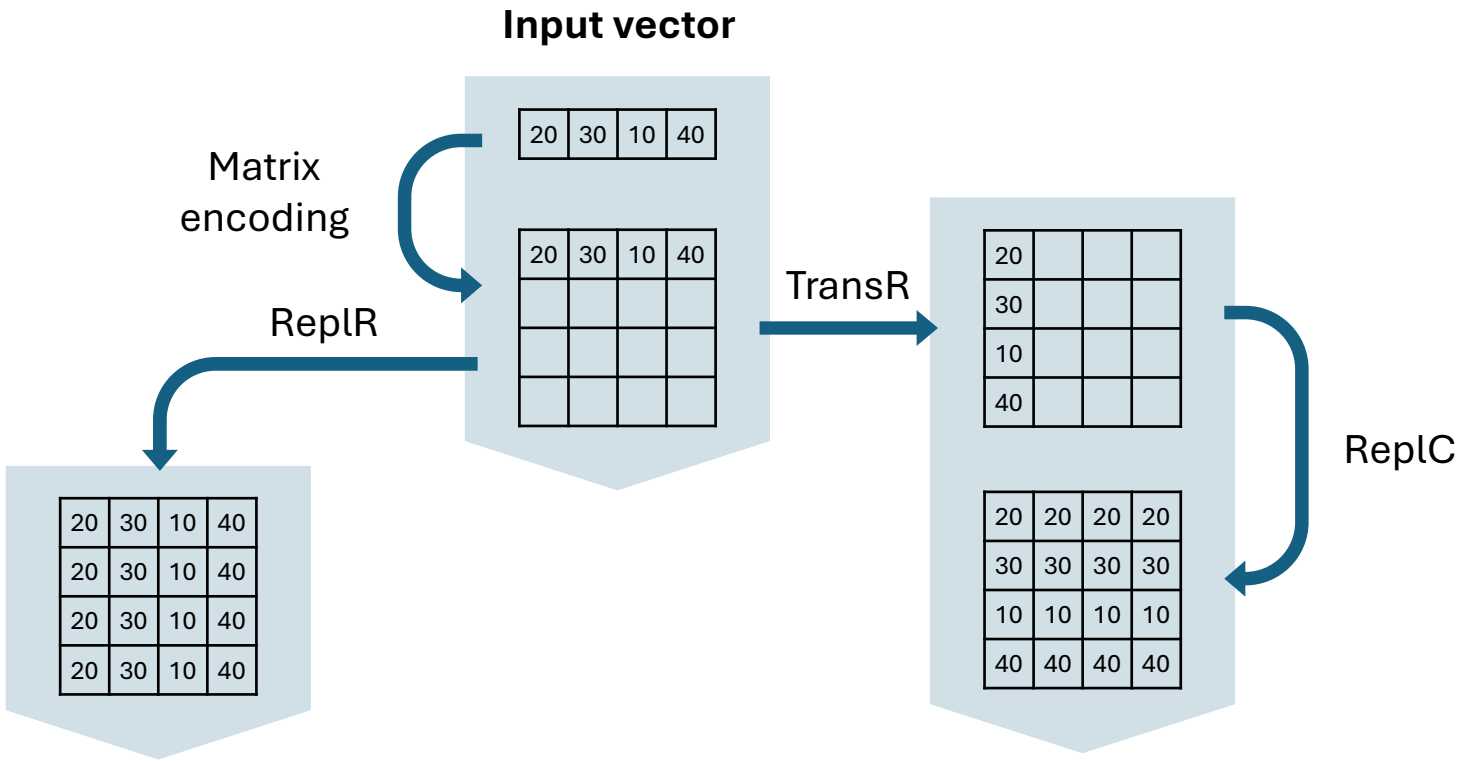
Ranking



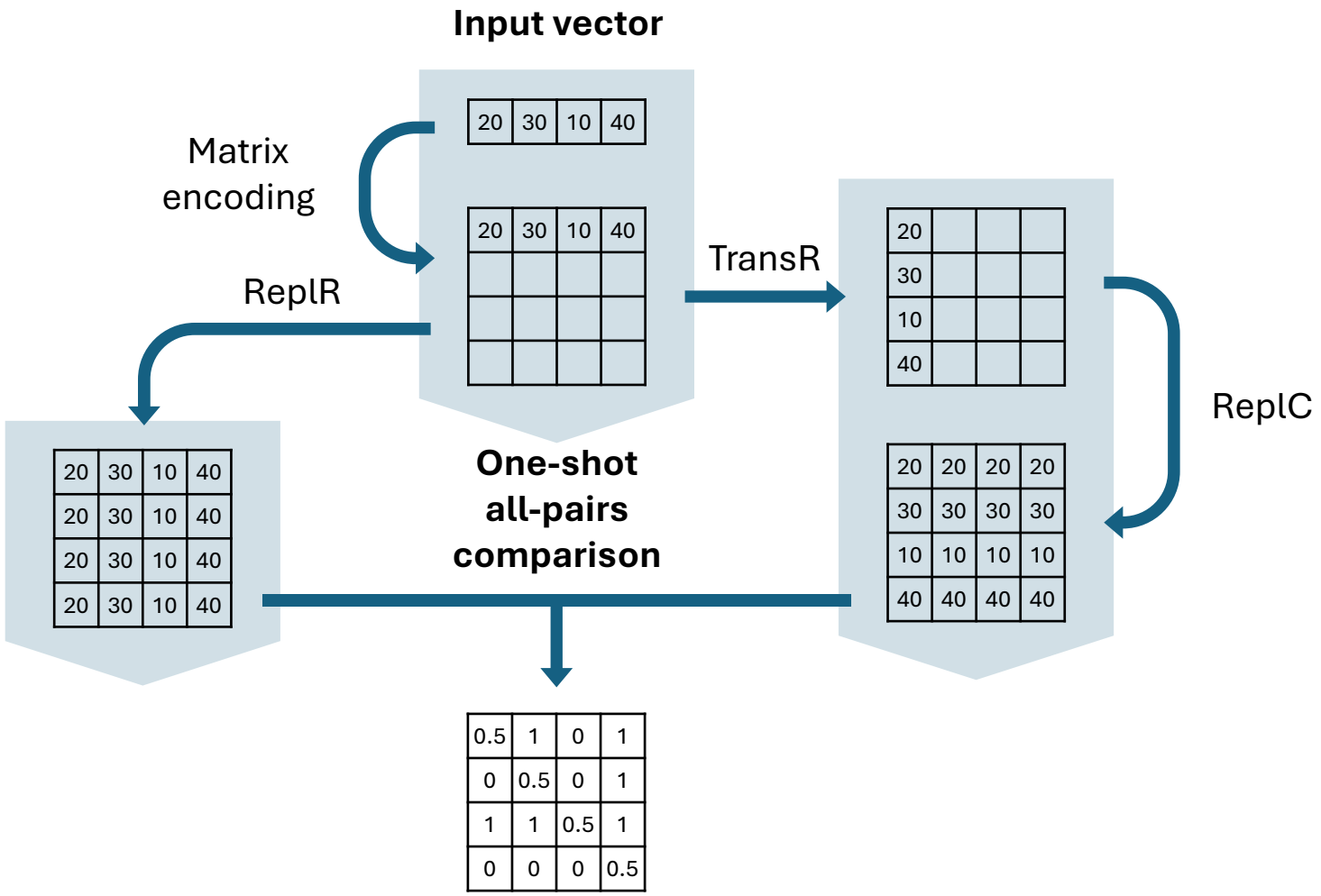
Ranking


Input vector
20 30 10 40
Matrix encoding
20 30 10 40
ReplR
20 30 10 40
20 30 10 40
20 30 10 40
20 30 10 40

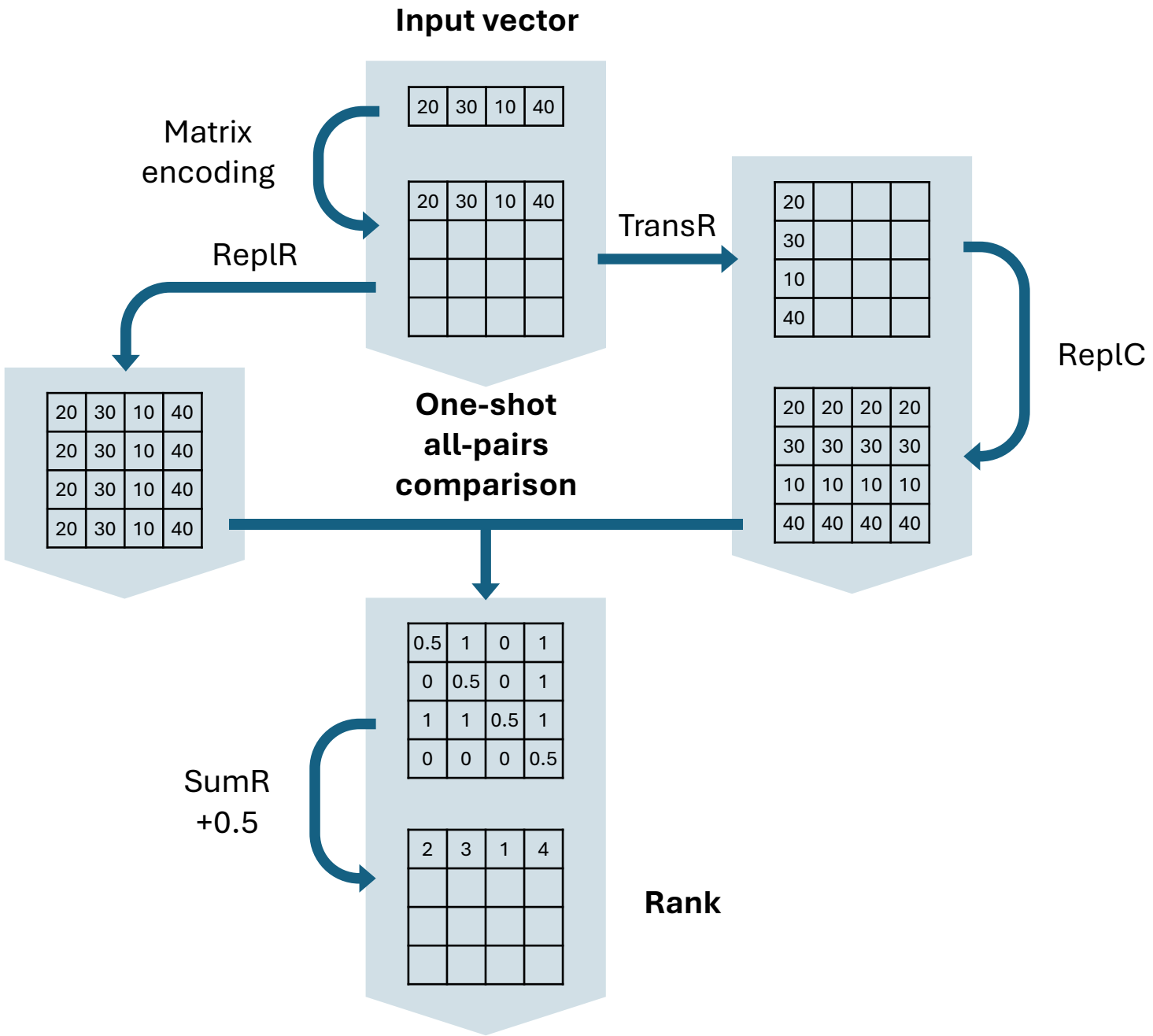
Ranking


Input vector
20 30 10 40
Matrix encoding
20 30 10 40
ReplR
TransR
20
30
10
40
ReplC
20 30 10 40
20 30 10 40
20 30 10 40
20 30 10 40
20 20 20 20
30 30 30 30
10 10 10 10
40 40 40 40

Ranking


Input vector
20 30 10 40
Matrix encoding
20 30 10 40
ReplR
TransR
20
30
10
40
ReplC
20 30 10 40
20 30 10 40
20 30 10 40
20 30 10 40
One-shot all-pairs comparison
20 20 20 20
30 30 30 30
10 10 10 10
40 40 40 40
0.5 1 0 1
0 0.5 0 1
1 1 0.5 1
0 0 0 0.5

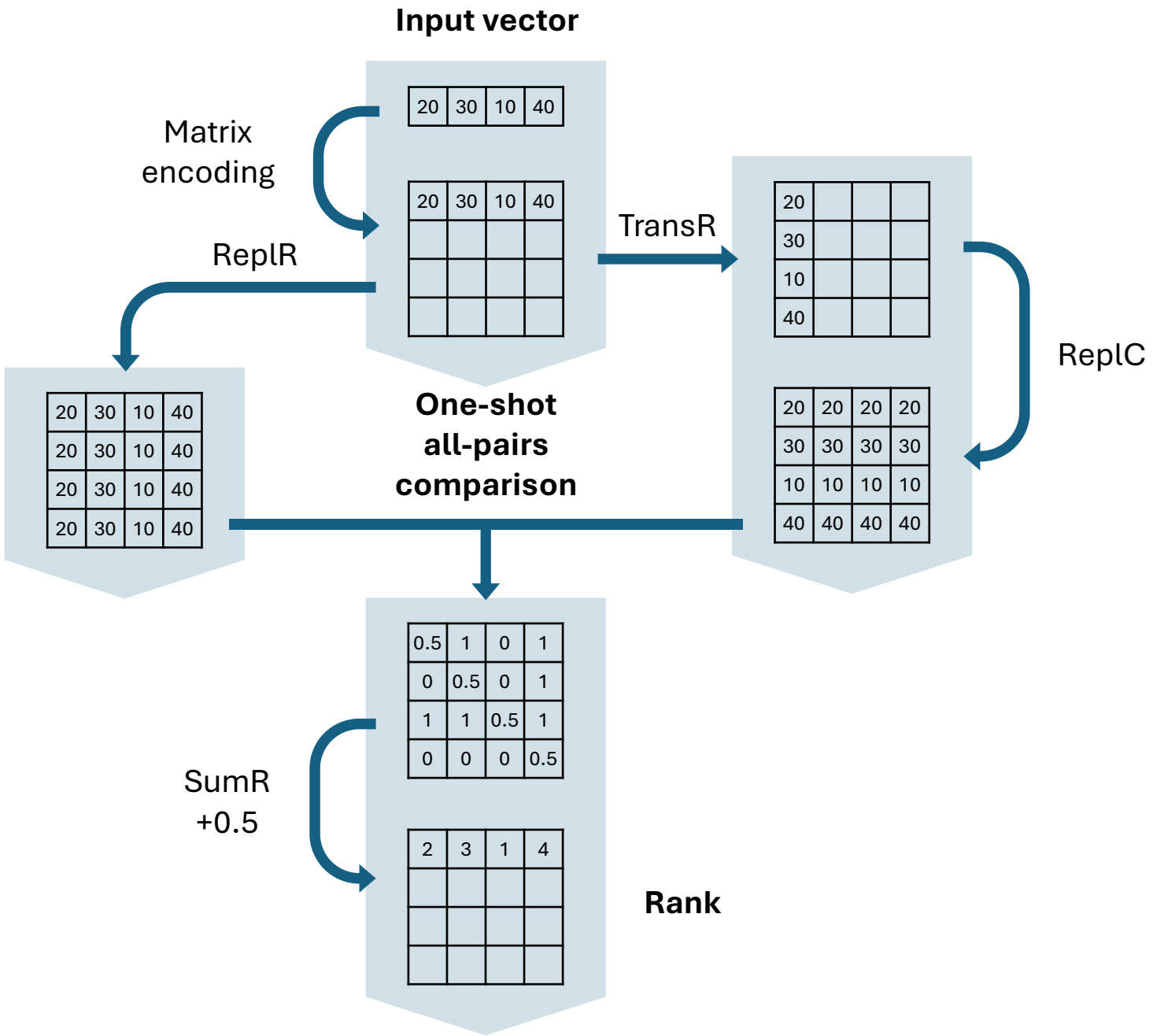
Ranking


Input vector
20 30 10 40
Matrix encoding
20 30 10 40
TransR
20
30
10
40
ReplR
ReplC
20 30 10 40
20 30 10 40
20 30 10 40
20 30 10 40
One-shot all-pairs comparison
20 20 20 20
30 30 30 30
10 10 10 10
40 40 40 40
0.5 1 0 1
0 0.5 0 1
1 1 0.5 1
0 0 0 0.5
SumR +0.5
2 3 1 4
Rank

Ranking

Extracting 1 order statistic

$$\mathrm{ind}_a(x) = \begin{cases} 1, & |x-a| < 0.5 \\ 0, & \text{otherwise} \end{cases}$$

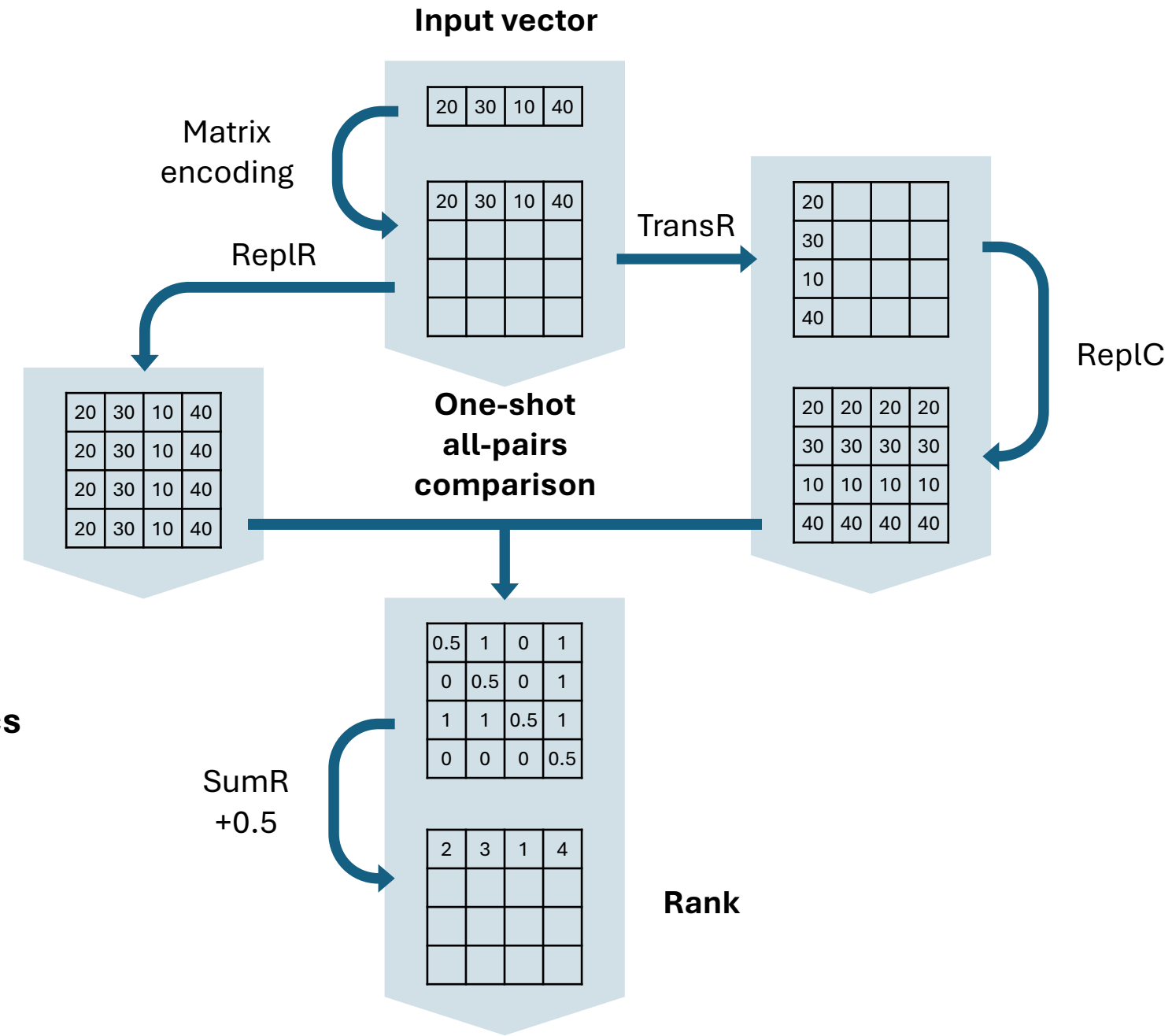


Ranking

Extracting 1 order statistic

$$\mathrm{ind}_a(x) = \begin{cases} 1, & |x-a| < 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Sorting = Extracting all N order statistics
(in parallel)

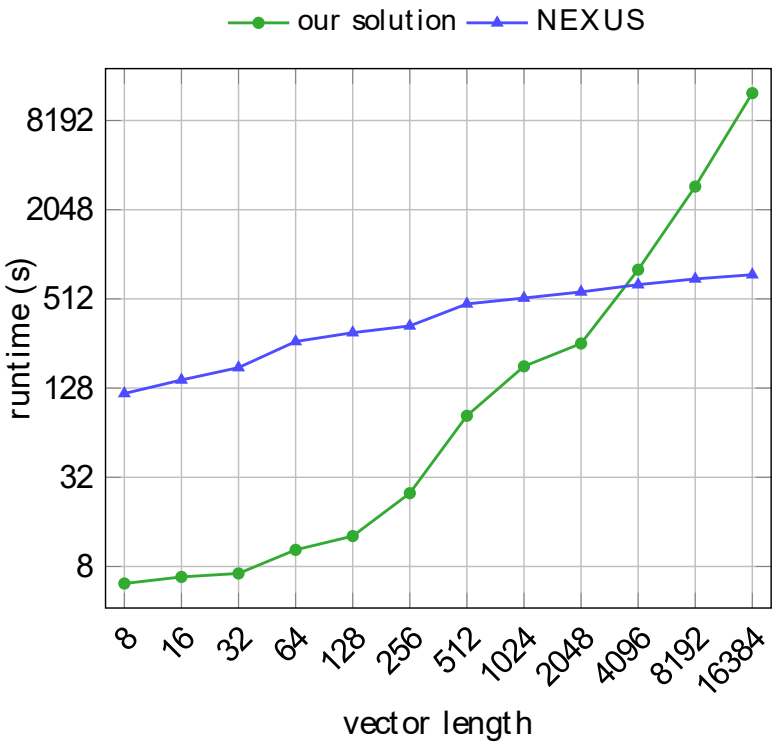


Limitation

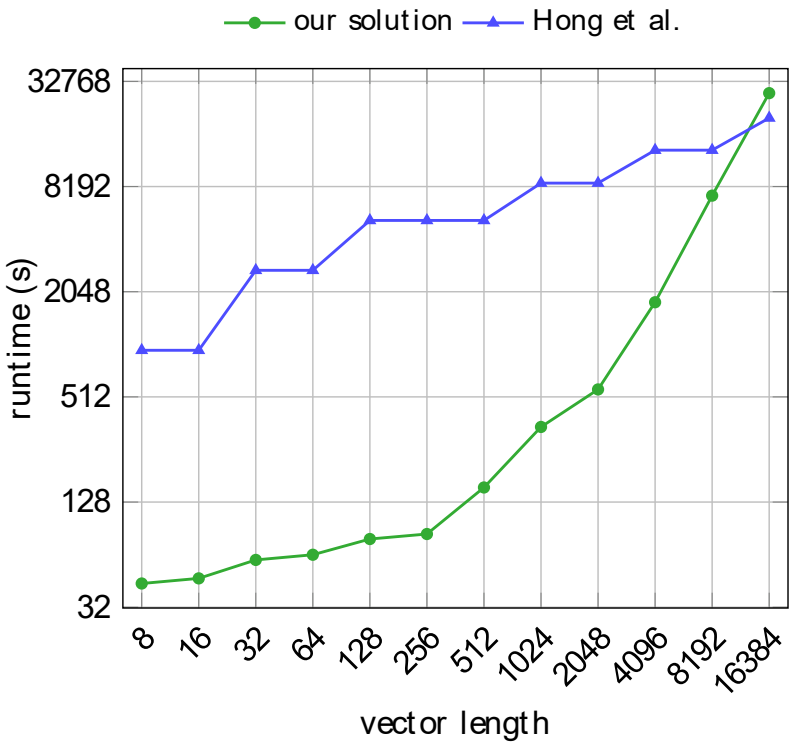
Our solution is very efficient only if $N \leq \sqrt{n}$ (number of ciphertext slots)

Limitation

Our solution is very efficient only if $N \leq \sqrt{n}$ (number of ciphertext slots)



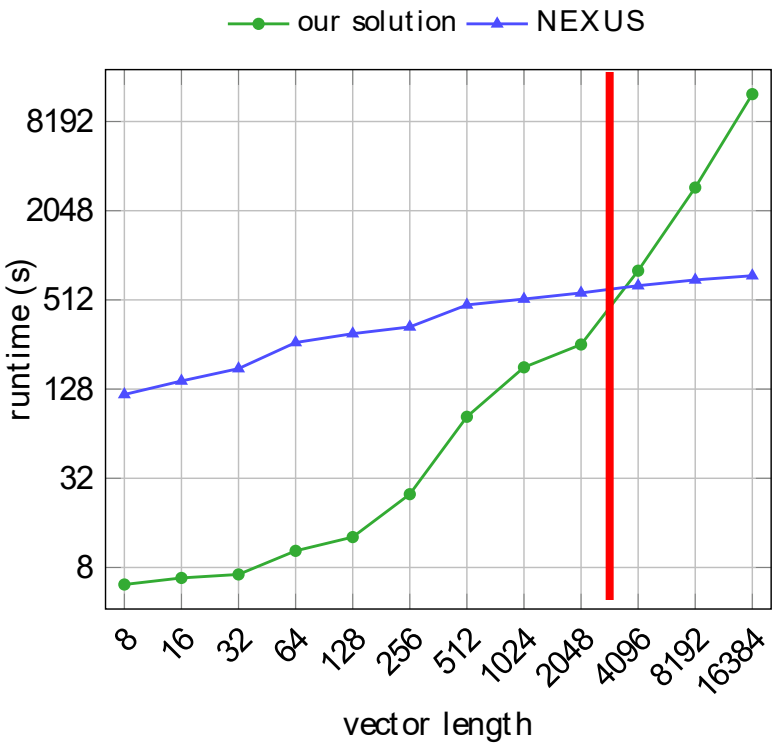
Argmax



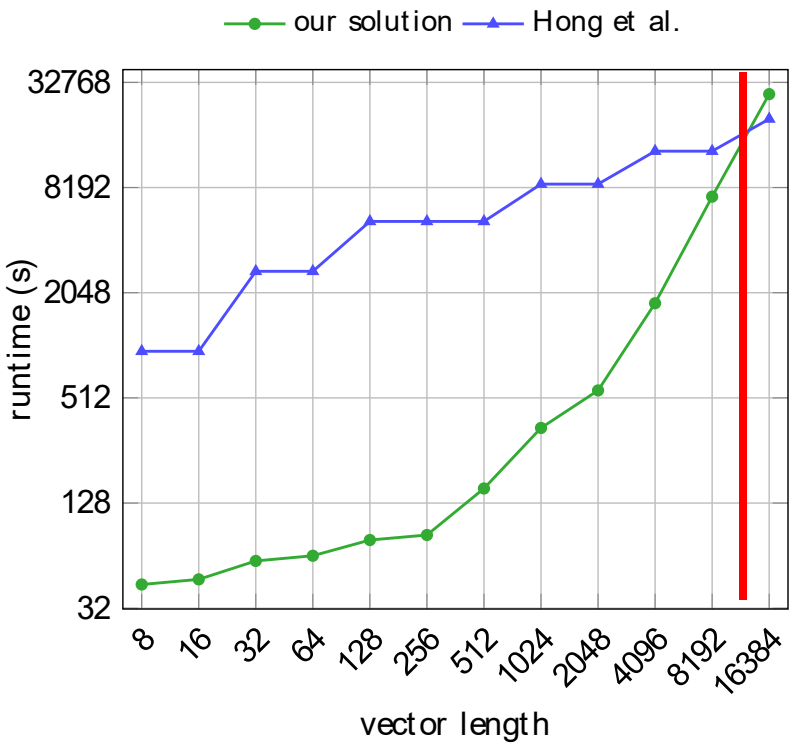
Sorting

Limitation

Our solution is very efficient only if $N \leq \sqrt{n}$ (number of ciphertext slots)



Argmax



Sorting

Our solution

- ✓ Small constant costs.
- ✗ Bad scaling (quadratic).
- ✓ Highly parallelizable.

Swap-Based solutions

- ✗ Large constant costs.
- ✓ Good scaling (logarithmic).
- ✗ Limited parallelization.

Our solution

- ☑ Small constant costs.
- ☒ Bad scaling (quadratic).
- ☑ Highly parallelizable.

▼

Ideal for **small input sizes** or highly-parallelizable environments.

Swap-Based solutions

- ☒ Large constant costs.
- ☑ Good scaling (logarithmic).
- ☒ Limited parallelization.

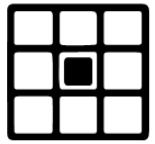
▼

Ideal for **large input sizes** (e.g., medium-large database queries).

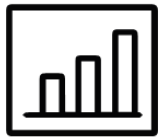
Many Applications in Machine Learning

Voting
[ranking]

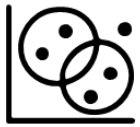
Tok-k feature selection
[ranking]



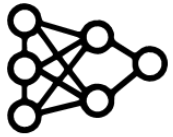
Median filtering
[median]



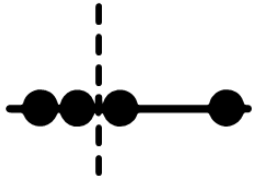
Quantization/ binning
[sorting]



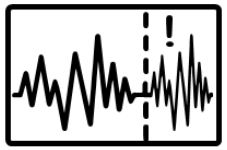
K-means
[argmin]



Neural network classification
[argmax]



Robust aggregation
[median]

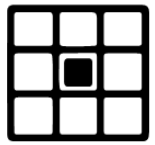


Time series change-point
[ranking]

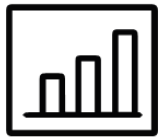
Many Applications in Machine Learning

Voting [ranking]

Tok-k feature selection [ranking]



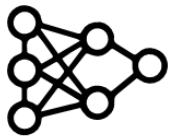
Median filtering [median]



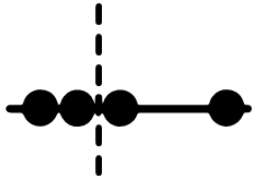
Quantization/ binning [sorting]

Work in progress!

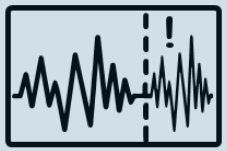
K-means [argmin]



Neural network classification [argmax]



Robust aggregation [median]



Time series change-point [ranking]

Work in progress!

Efficient Ranking, Order Statistics, and Sorting under CKKS

Corresponding author:
Federico Mazzone
f.mazzone@utwente.nl

UNIVERSITY OF TWENTE. | Semantics Cybersecurity Services

View the Paper

GitHub Repo