Graphiti: Secure Graph Computation Made More Scalable

Nishat Koti, Varsha Bhat Kukkala, Arpita Patra and **Bhavish Raj Gopal**

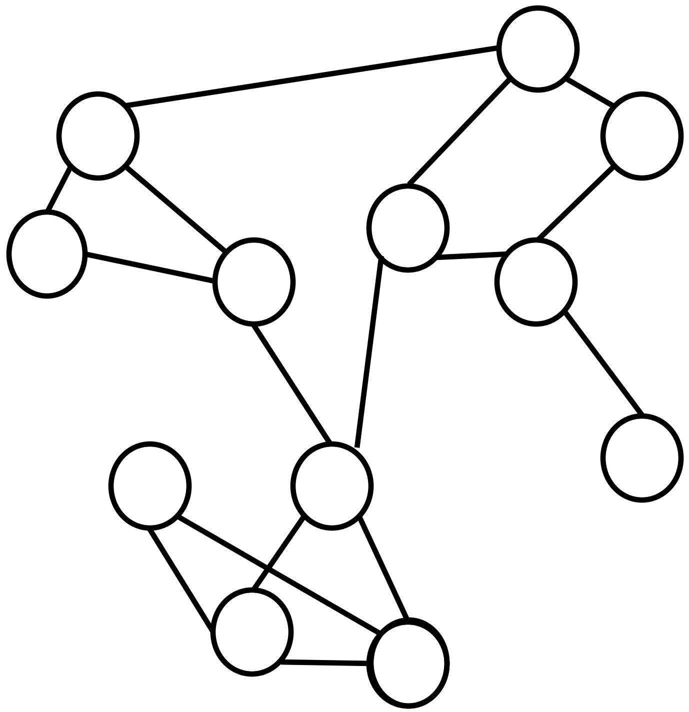
ACM CCS 2024

Presented at WPEC 2024, September 26

NIST Workshop on Privacy-Enhancing Cryptography 2024

Motivation

- Graphs - social network, contact network, communication network, etc.
- BFS, DFS, PageRank, Clustering
- Graphs may be distributed



G (V, E)

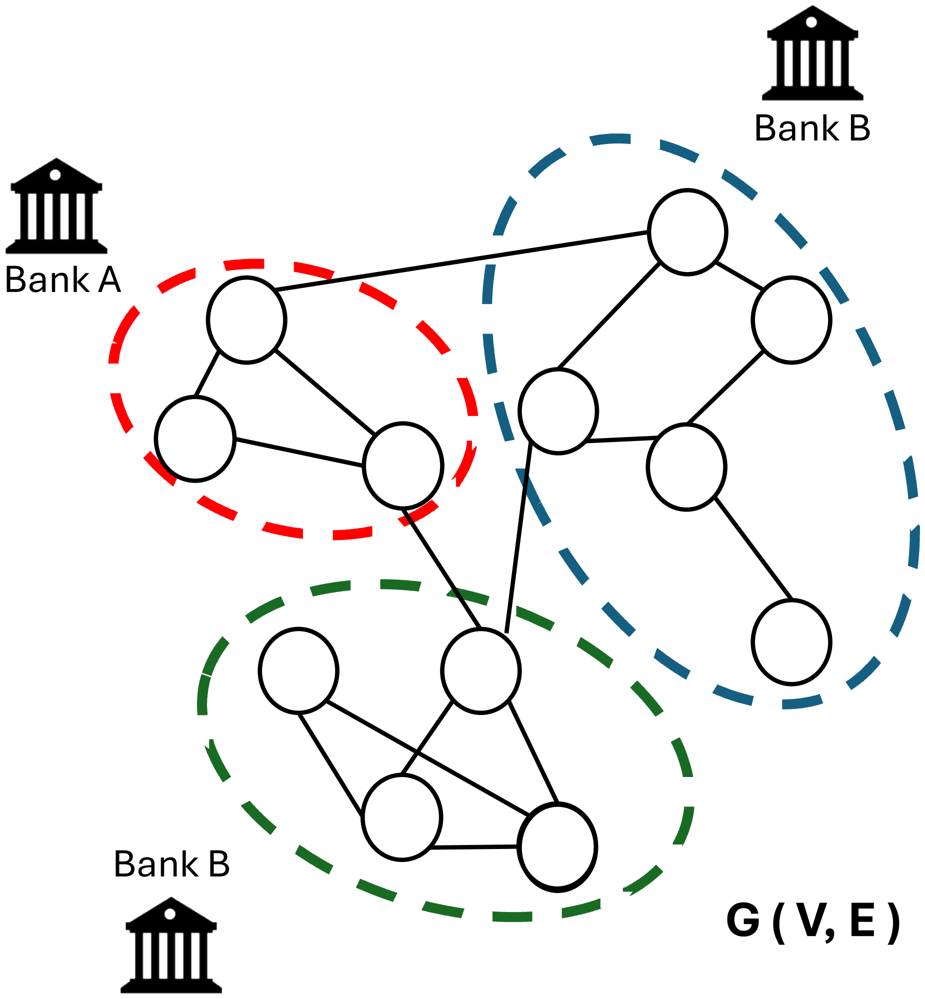
Motivation

- Transaction network

- Vertices: Bank accounts
  Edges: transactions

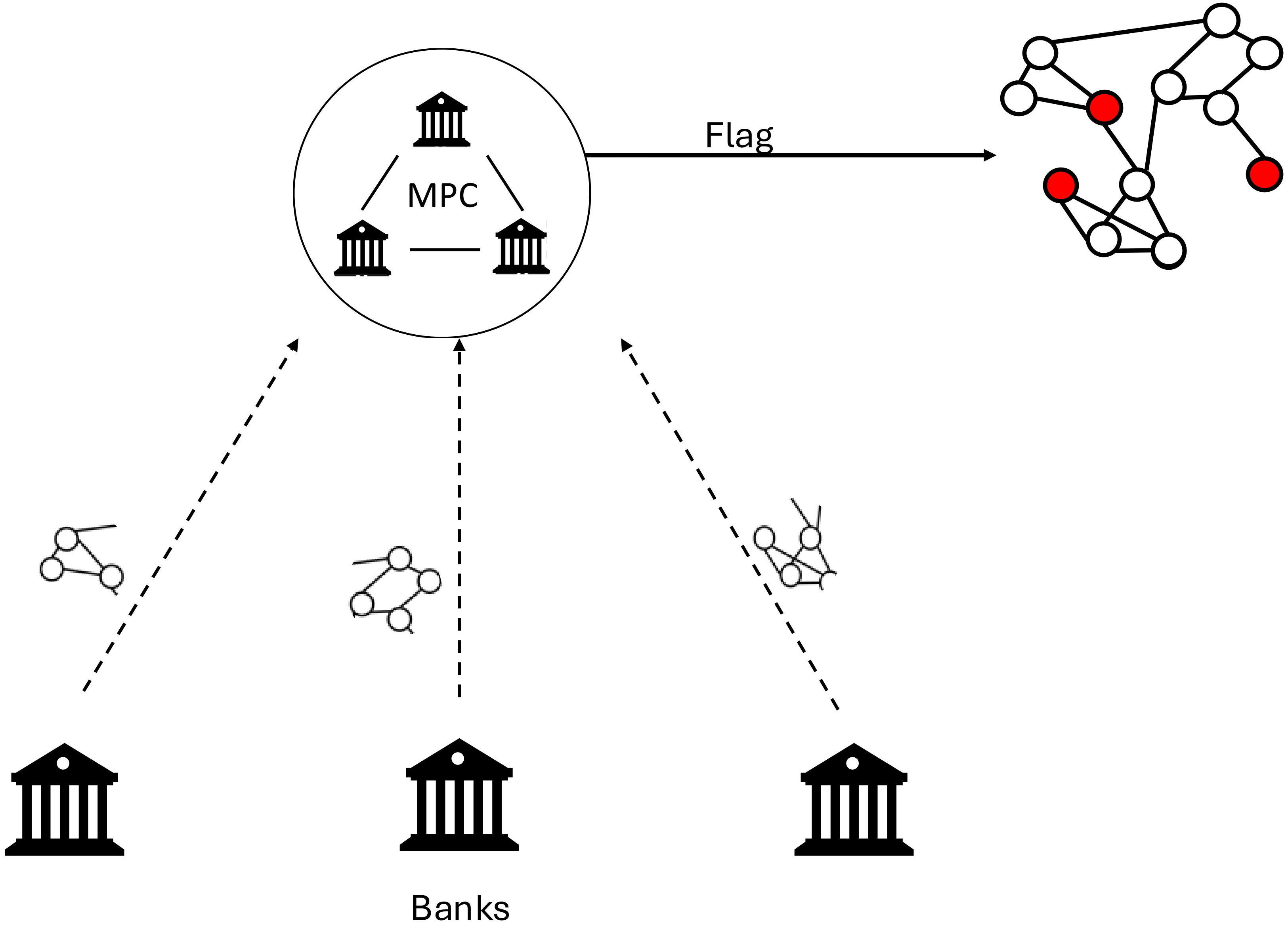
- Graphs distributed across different banks.

- Fraud detection, Anti-money laundering, credit risk analysis…



Fraud Detection

Modelled as an instance of MPC



Secure Graph Computation: Challenges

Privacy:

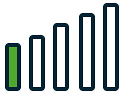
- Data associated with nodes and edges of the graph must remain hidden
- Topology of the graph must remain hidden

Scalability:

- Graphs can be very large, containing millions or billions of nodes and edges.

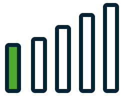
Parameters:

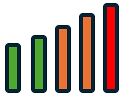
- Rounds – sequential interactions between the parties during the run of the MPC protocol
- Communication – data exchanged between the parties during the run of the MPC protocol
- Computation – local computations performed at each party during the run of the MPC protocol

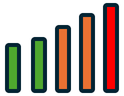


Secure Graph Computation: Evolution

[BS05]
[AV15]
[KSI16]

Rounds 

Communication 

Computation 

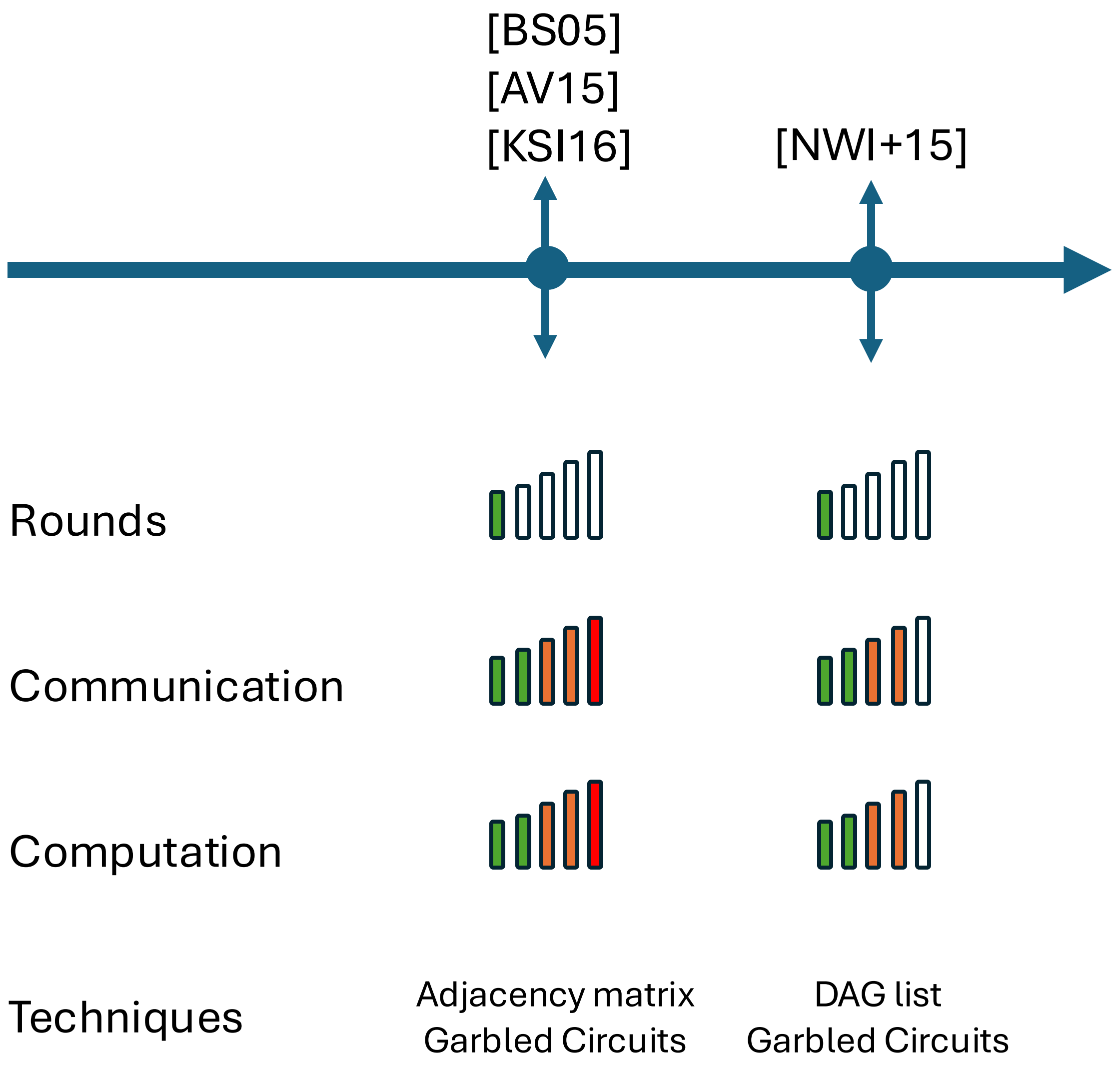
Techniques: Adjacency matrix, Garbled Circuits

Off-the-shelf MPC protocols

- Adjacency matrix representation of the graph
- $O(|V|^2)$
- Overkill for sparse graphs

[BS05] - Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In ASIACRYPT, 2005.
[AV15] - Abdelrahaman Aly and Mathieu Van Vyve. Securely solving classical network flow problems. In ICISC, 2015
[KSI16] - Varsha Bhat Kukkala, Jaspal Singh Saini, and SRS Iyengar. Privacy preserving network analysis of distributed social networks. In ICISS, 2016.

Secure Graph Computation: Evolution

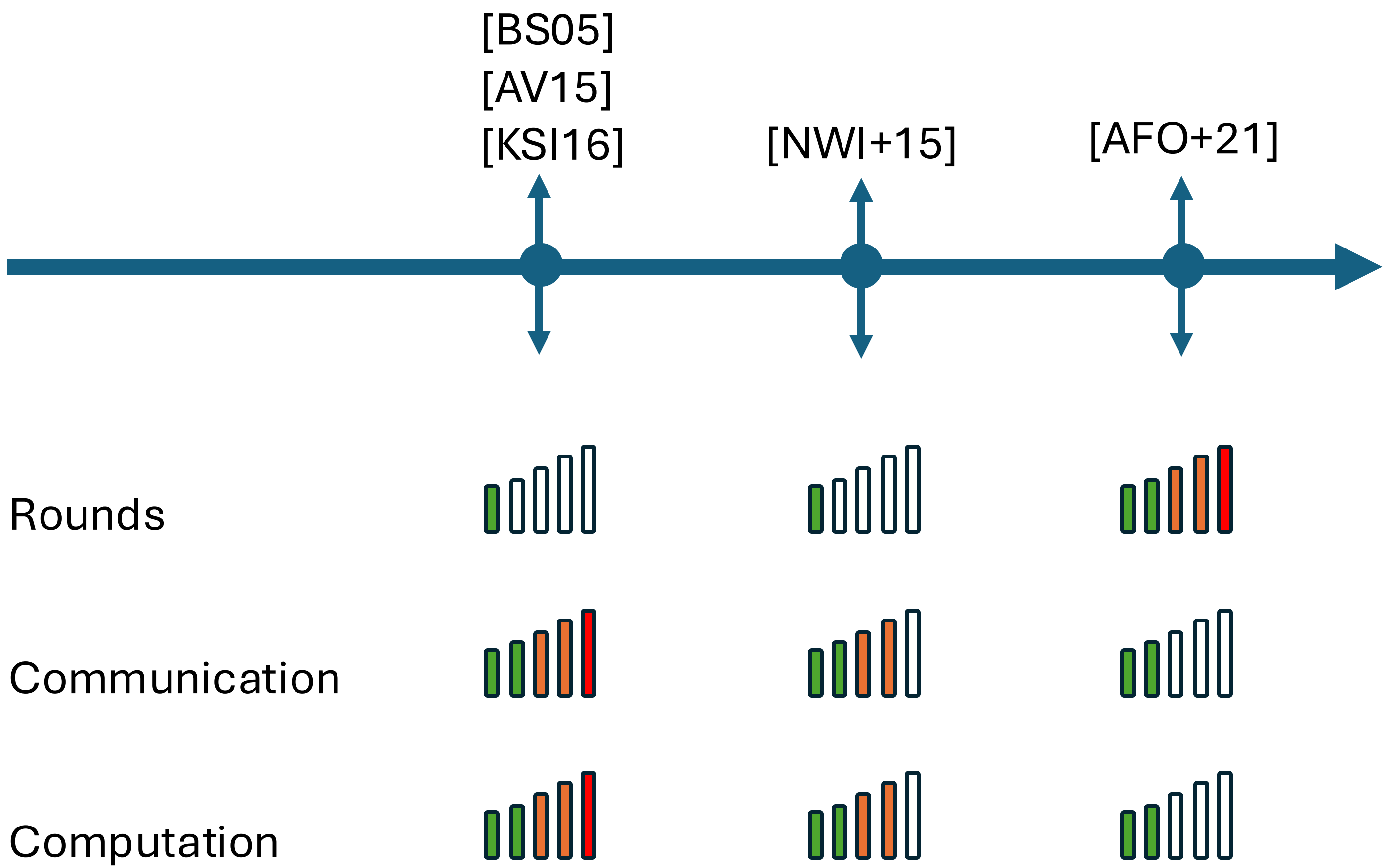


GraphSC[NWI+15]

- List representation of the graph
- $O(|V| + |E|)$
- More efficient

[NWI+15] - Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. Graphsc: Parallel secure computation made easy. In IEEE S&P, 2015.

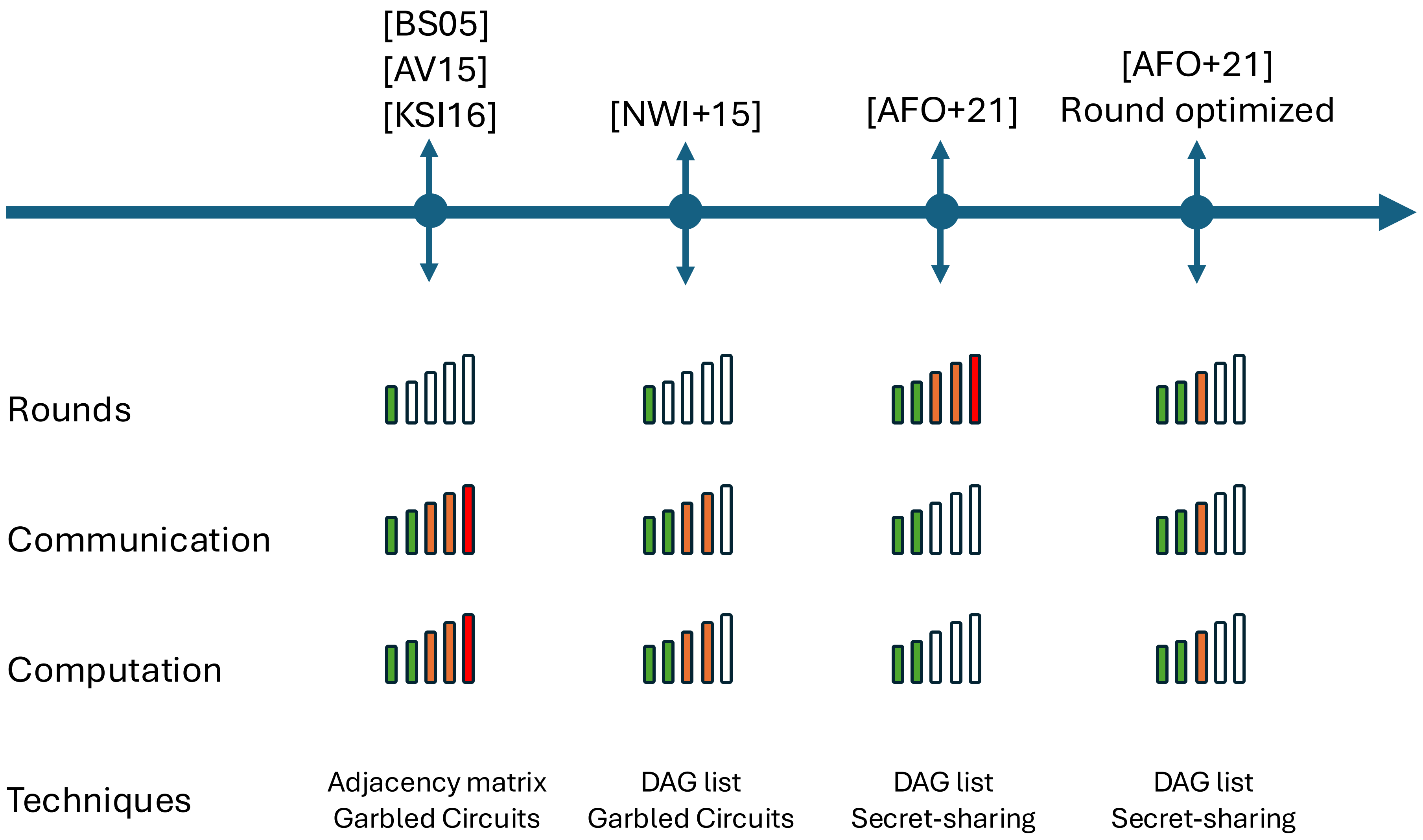
Secure Graph Computation: Evolution



Techniques	Adjacency matrix Garbled Circuits	DAG list Garbled Circuits	DAG list Secret-sharing

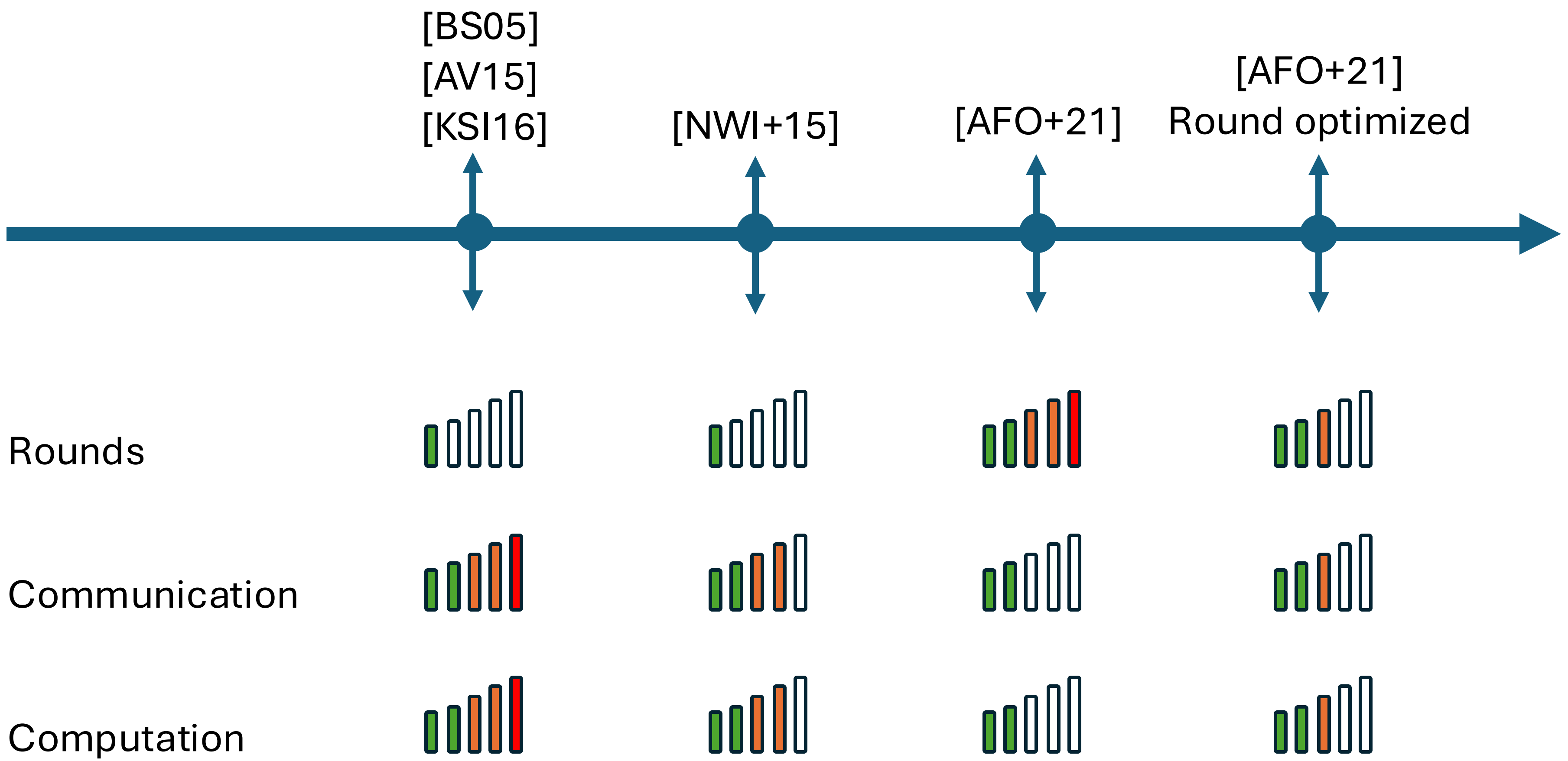
[AFO+21] - Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ACM CCS, 2021.

Secure Graph Computation: Evolution



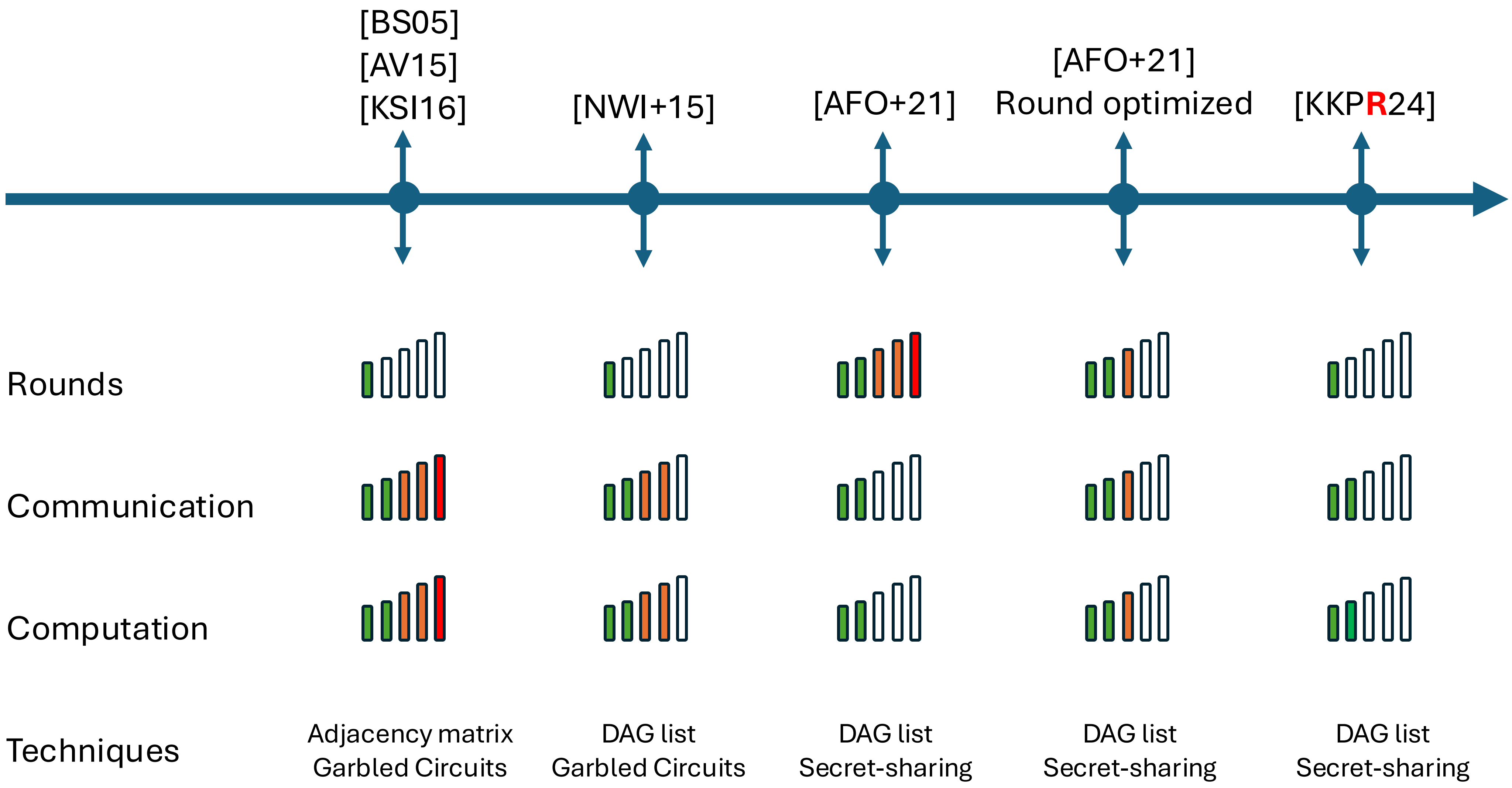
[AFO+21] - Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ACM CCS, 2021.

Secure Graph Computation: Evolution



Can we get the best of all?

Secure Graph Computation: Evolution

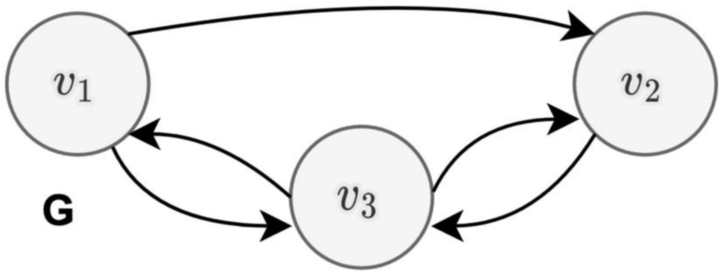


[KKP**R**24] - Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, and Bhavish Raj Gopal. “Graphiti: Secure Graph Computation Made More Scalable.” ACM CCS, 2024.

Our Contributions

- Generic framework for securely realizing any message passing graph algorithm.
  - Black-box use of MPC.
- Efficient and Scalable.
  - Rounds complexity independent of graph size.
  - Improved computation cost.
- Implementation and evaluation in 2PC semi-honest outsourced computation setting.
  - Improvements of up to 3 orders of magnitude in run time
- Design of improved secure shuffle protocol in the considered setting.

GraphSC[NWI+15]



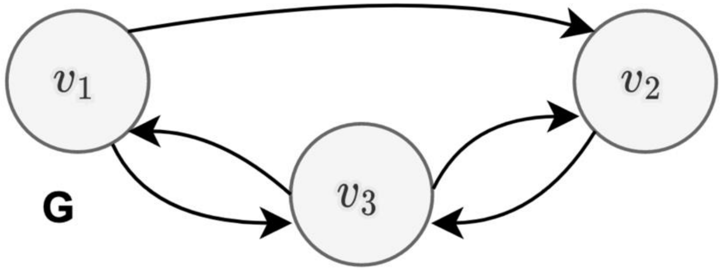
Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

[NWI+15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. “GraphSC: Parallel secure computation made easy.” ***IEEE S&P, 2015***.

GraphSC[NWI+15]

Data augmented graph (DAG) list

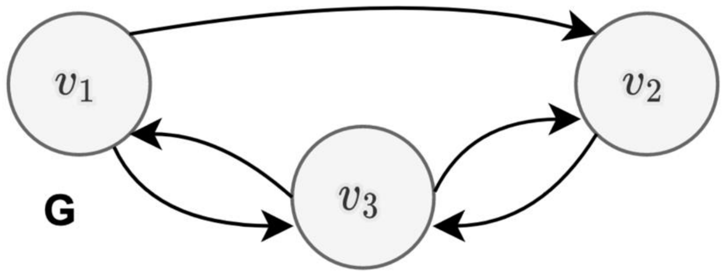


v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

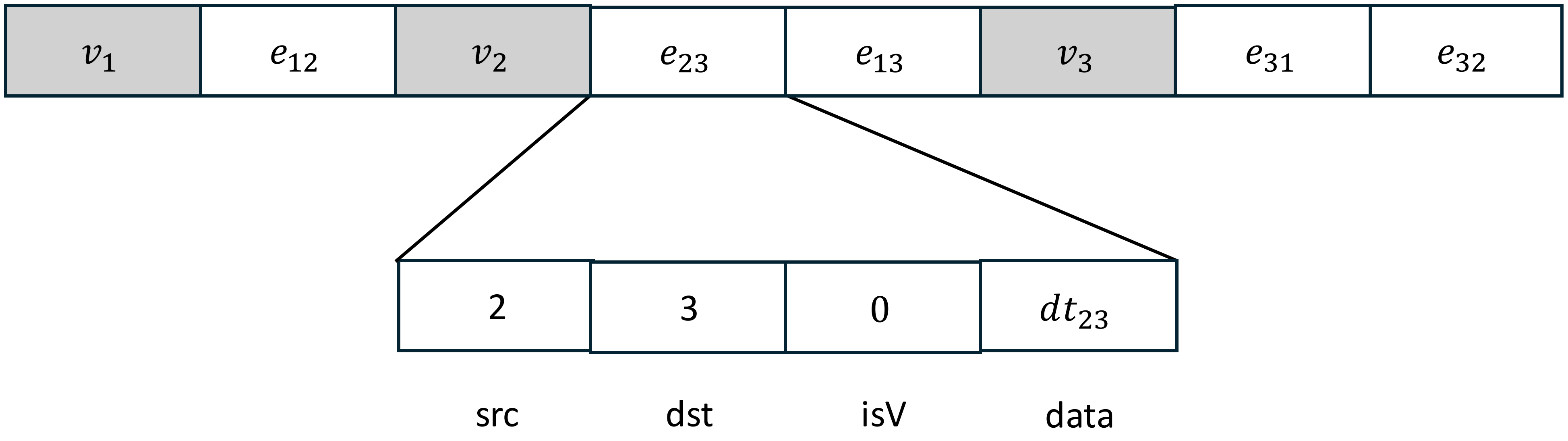
2	2	1	dt_2
src	dst	isV	data

[NWI+15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. “GraphSC: Parallel secure computation made easy.” ***IEEE S&P, 2015***.

GraphSC[NWI+15]

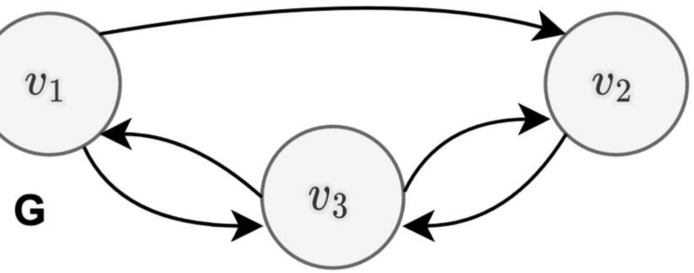


Data augmented graph (DAG) list



[NWI+15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. "GraphSC: Parallel secure computation made easy." ***IEEE S&P, 2015***.

GraphSC[NWI+15]



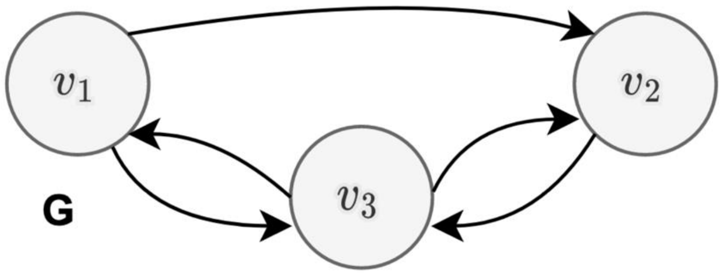
Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

GraphSC facilitates secure evaluation of message passing algorithms

[NWI+15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. “GraphSC: Parallel secure computation made easy.” ***IEEE S&P, 2015***.

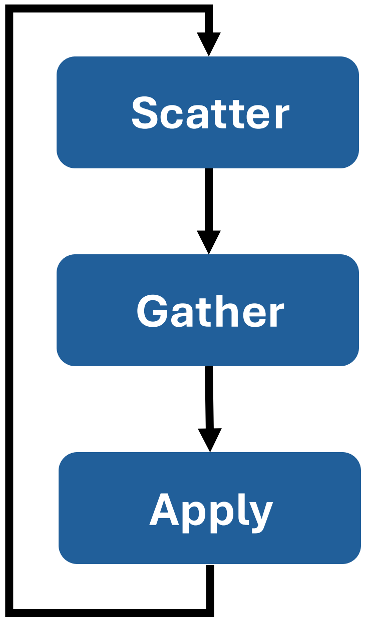
GraphSC[NWI+15]



Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

GraphSC facilitates secure evaluation of message passing algorithms



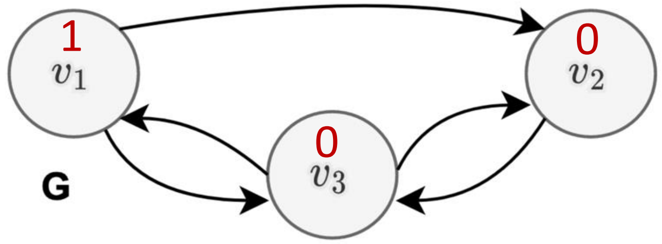
Message passing algorithms

- Operate in multiple iterations
- Each iteration
  - Scatter - Nodes send messages on outgoing edges
  - Gather - Nodes receive messages on incoming edges and aggregate these messages
  - Apply - Nodes use aggregated messages to update their state

[NWI+15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. “GraphSC: Parallel secure computation made easy.” ***IEEE S&P, 2015***.

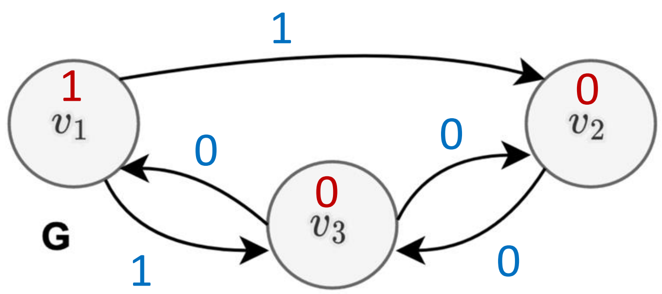
Message Passing Algorithms

Message Passing Algorithms



Example: to identify nodes that are reachable within a one-hop distance from v_1

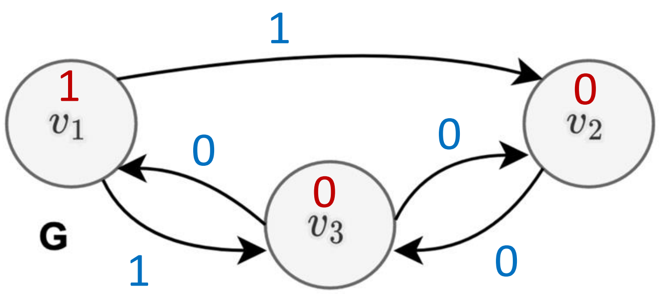
Message Passing Algorithms

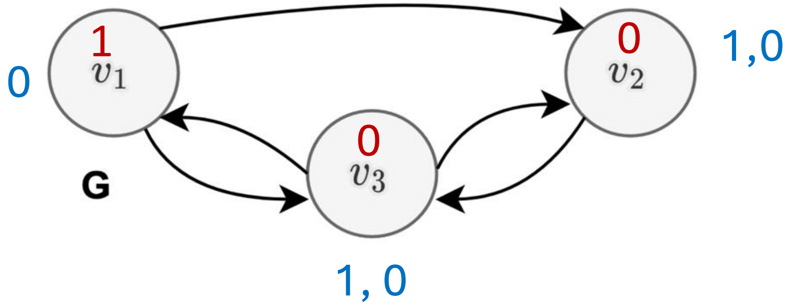


Scatter data on outgoing edges

Example: to identify nodes that are reachable within a one-hop distance from v_1

Message Passing Algorithms

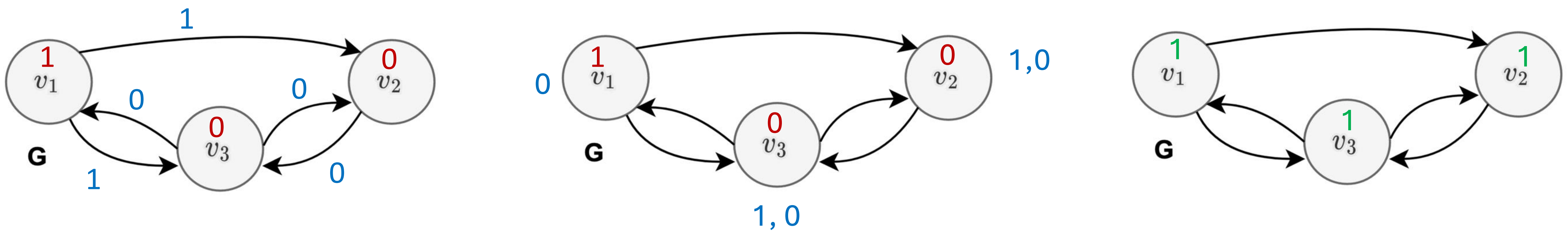




Scatter data on outgoing edges → Gather data from incoming edges

Example: to identify nodes that are reachable within a one-hop distance from v_1

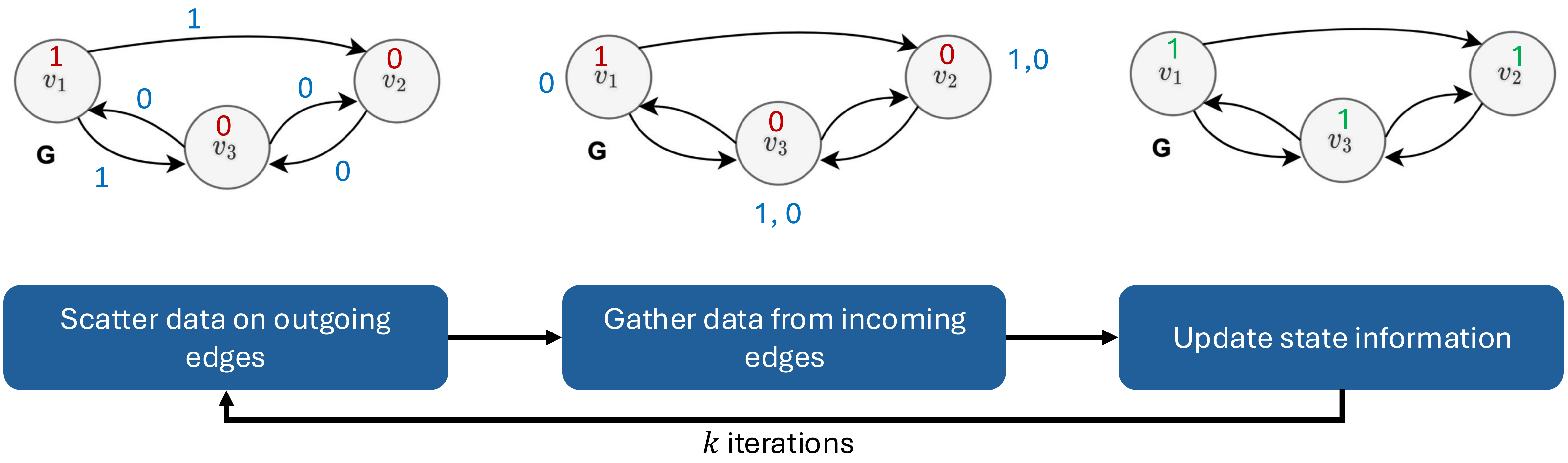
Message Passing Algorithms



Scatter data on outgoing edges → Gather data from incoming edges → Update state information

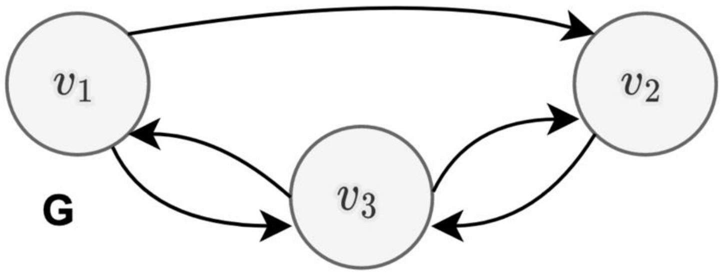
Example: to identify nodes that are reachable within a one-hop distance from v_1

Message Passing Algorithms



Example: to identify nodes that are reachable within a k-hop distance from v_1

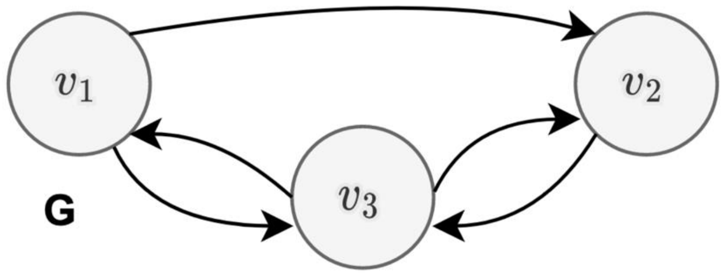
GraphSC



Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

GraphSC



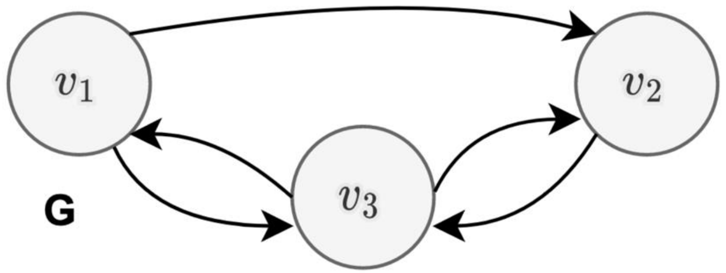
Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

Scatter: Source ordering

v_1	e_{12}	e_{13}	v_2	e_{23}	v_3	e_{31}	e_{32}

GraphSC



Data augmented graph (DAG) list

v_1	e_{12}	v_2	e_{23}	e_{13}	v_3	e_{31}	e_{32}

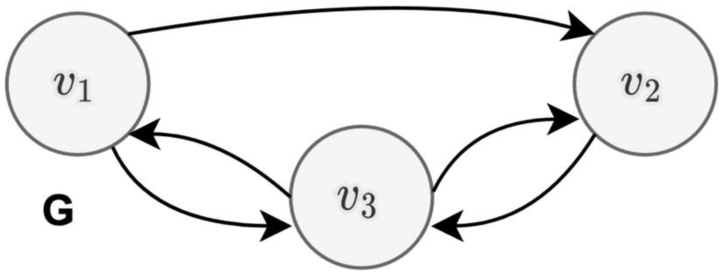
Scatter: Source ordering

v_1	e_{12}	e_{13}	v_2	e_{23}	v_3	e_{31}	e_{32}

Gather: Destination ordering

e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3

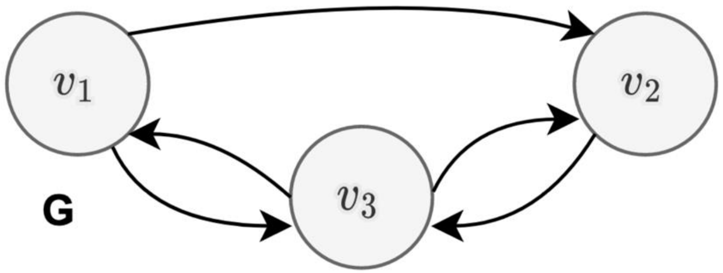
GraphSC: Scatter



A vertex propagates data to its neighboring edges and updates the edge's data

$$\mathrm{e.\,data} = f(\mathrm{e.\,data}, \mathrm{u.\,data})\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$$

GraphSC: Scatter

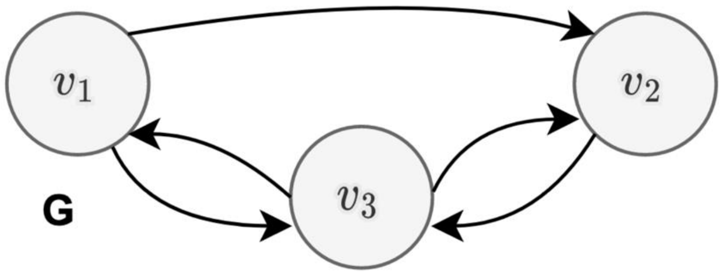


A vertex propagates data to its neighboring edges and updates the edge's data

$$\mathrm{e.data} = f(\mathrm{e.data}, \mathrm{u.data})\ \forall\, \mathrm{e(u, v)} \in \mathrm{E}$$

G	v_1	e_{12}	e_{13}	v_2	e_{23}	v_3	e_{31}	e_{32}	Source order
$data_e$	0	y_{12}	y_{13}	0	y_{23}	0	y_{31}	y_{32}	
$data_v$	x_1	0	0	x_2	0	x_3	0	0	

GraphSC: Scatter



A vertex propagates data to its neighboring edges and updates the edge's data

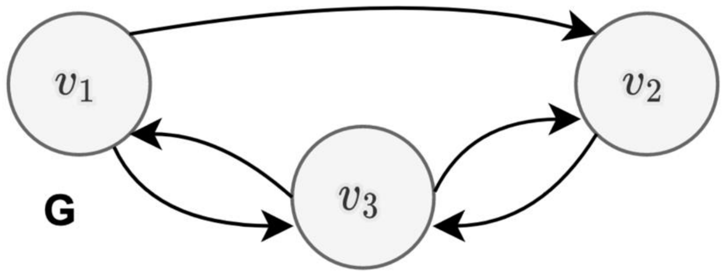
$$\mathrm{e.data} = f(\mathrm{e.data}, \mathrm{u.data})\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$$

G	v_1	e_{12}	e_{13}	v_2	e_{23}	v_3	e_{31}	e_{32}	Source order
$data_e$	0	y_{12}	y_{13}	0	y_{23}	0	y_{31}	y_{32}	

Linear Scan

$data_v$	x_1	0	0	x_2	0	x_3	0	0

GraphSC: Scatter



A vertex propagates data to its neighboring edges and updates the edge's data

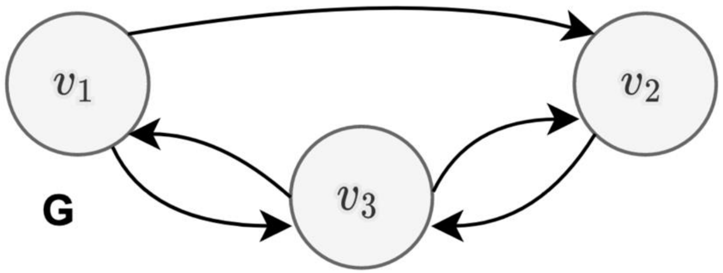
$$\mathrm{e.data} = f(\mathrm{e.data}, \mathrm{u.data})\ \forall\, \mathrm{e(u, v)} \in \mathrm{E}$$

G	v_1	e_{12}	e_{13}	v_2	e_{23}	v_3	e_{31}	e_{32}	Source order
$data_e$	0	y_{12}	y_{13}	0	y_{23}	0	y_{31}	y_{32}	

Linear Scan

$data_v$	x_1	0	0	x_2	0	x_3	0	0
	Pick x_1	Drop x_1 and update	Drop x_1 and update	Pick x_2	Drop x_2 and update	Pick x_3	Drop x_3 and update	Drop x_3 and update
Updated $data_e$	0	$f(y_{12}, x_1)$	$f(y_{13}, x_1)$	0	$f(y_{23}, x_2)$	0	$f(y_{31}, x_3)$	$f(y_{32}, x_3)$

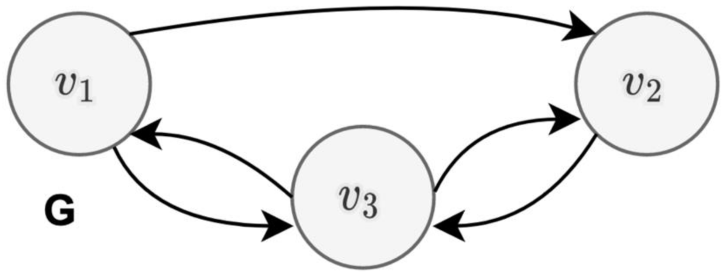
GraphSC: Gather



A vertex aggregates data from its neighboring edges using a binary operator $\bigoplus$

$$\mathrm{v.data} = \mathrm{v.data} \mid\mid \bigoplus_{\forall\, \mathrm{e(u,v)} \in \mathrm{E}} \mathrm{e.data}$$

GraphSC: Gather

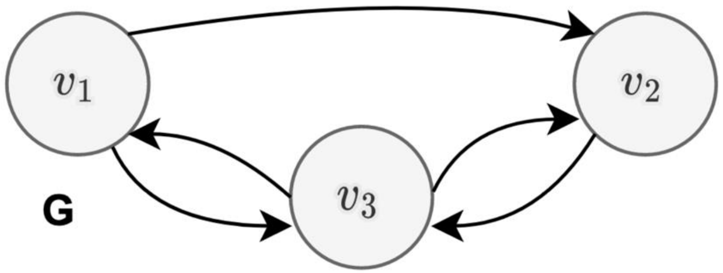


A vertex aggregates data from its neighboring edges using a binary operator $\bigoplus$

$$\mathrm{v.data} = \mathrm{v.data} \,||\, \bigoplus_{\forall\, \mathrm{e(u,v)} \in \mathrm{E}} \mathrm{e.data}$$

G	e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3	Destination order
$data_e$	x_3	0	x_1	x_3	0	x_1	x_2	0	

GraphSC: Gather



A vertex aggregates data from its neighboring edges using a binary operator $\bigoplus$

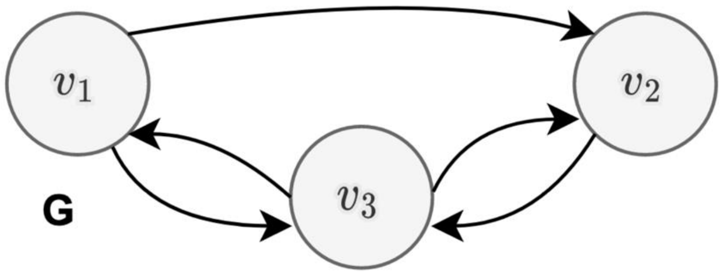
$$\text{v.data} = \text{v.data} \,||\, \bigoplus_{\forall\, e(u,v) \in E} \text{e.data}$$

G	e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3	Destination order

Linear Scan →

$data_e$	x_3	0	x_1	x_3	0	x_1	x_2	0

GraphSC: Gather



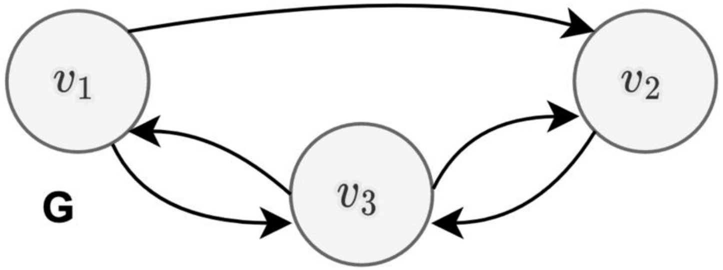
A vertex aggregates data from its neighboring edges using a binary operator $\bigoplus$

$$\text{v.data} = \text{v.data} \,||\, \bigoplus_{\forall\, e(u,v) \in E} \text{e.data}$$

G	e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3	Destination order
$data_e$	x_3	0	x_1	x_3	0	x_1	x_2	0	
	Agg x_3	Drop and update	Agg x_1	Agg x_3	Drop and update	Agg x_1	Agg x_2	Drop and update	
Updated $data_v$	0	$dv_1 \| \| x_3$	0	0	$dv_2 \| \| x_1 \oplus x_3$	0	0	$dv_3 \| \| x_1 \oplus x_2$	

Linear Scan

GraphSC: Apply

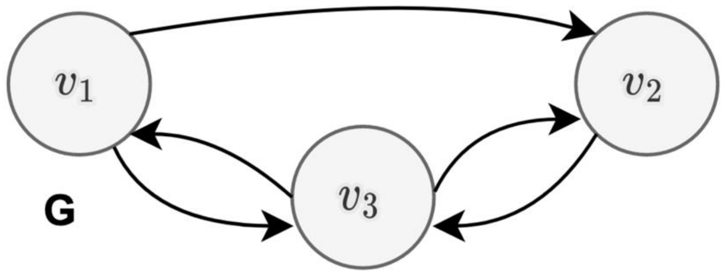


A vertex updates its data

$$\text{v. data} = f_V(\text{v. data})$$

G	e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3	Destination order
$data_v$	0	$dv_1 \| \| x_3$	0	0	$dv_2 \| \| x_1 \oplus x_3$	0	0	$dv_3 \| \| x_1 \oplus x_2$	

GraphSC: Apply

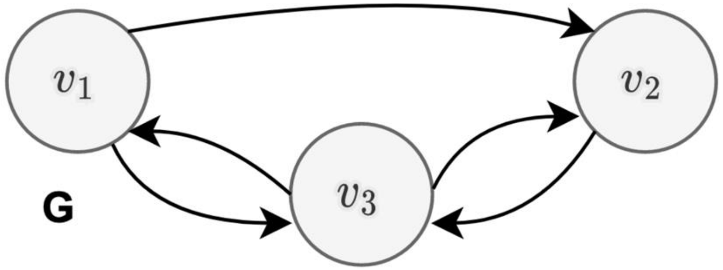


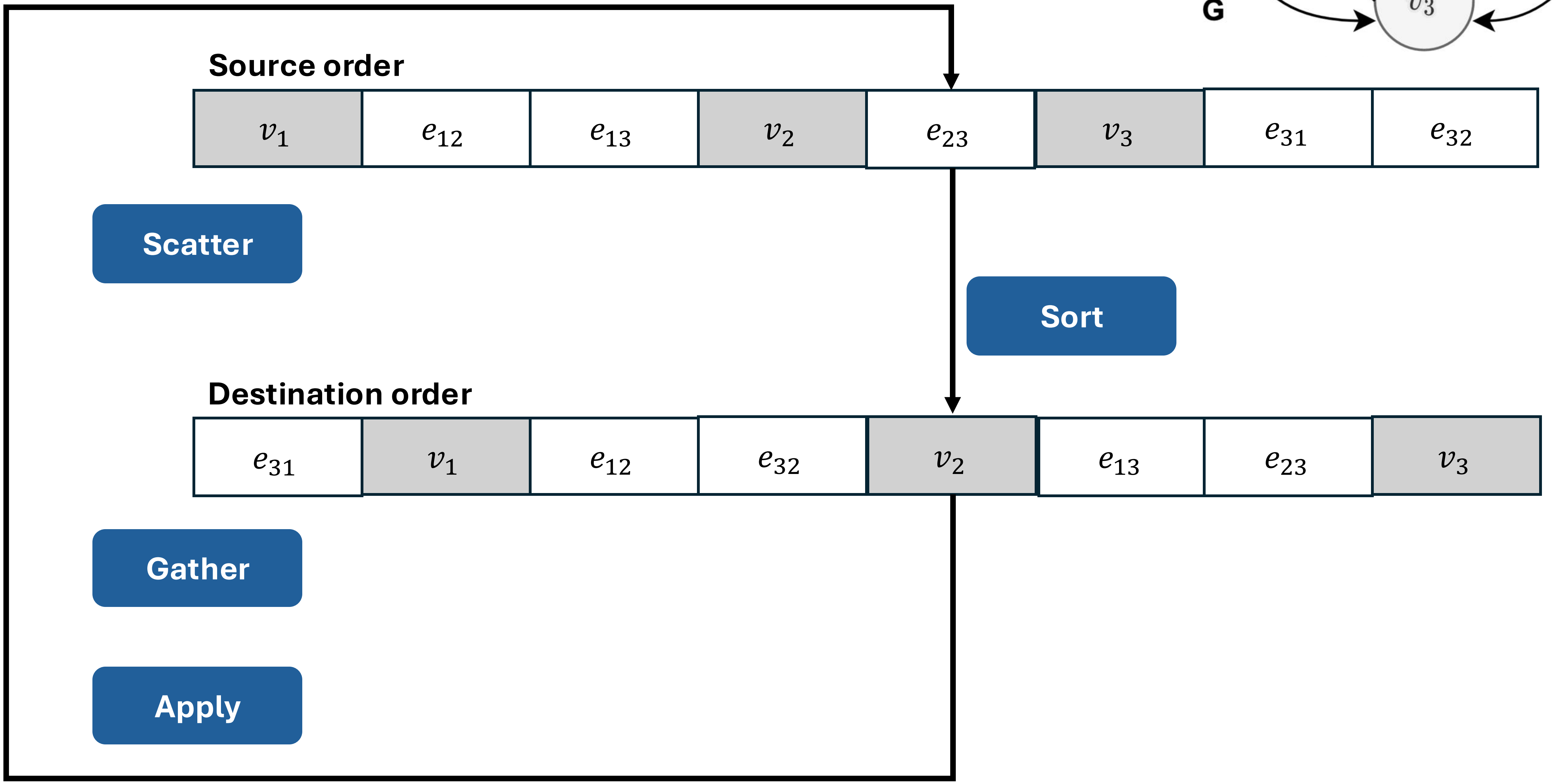
A vertex updates its data

$$\text{v.data} = f_V(\text{v.data})$$

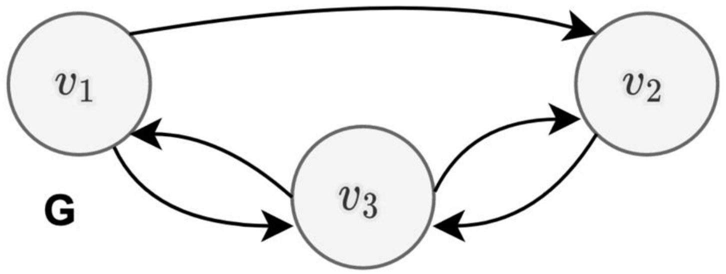
G	e_{31}	v_1	e_{12}	e_{32}	v_2	e_{13}	e_{23}	v_3	Destination order
$data_v$	0	$dv_1 \| \| x_3$	0	0	$dv_2 \| \| x_1 \oplus x_3$	0	0	$dv_3 \| \| x_1 \oplus x_2$	
Updated $data_v$	0	$f_V(dv_1 \| \| x_3)$	0	0	$f_V(dv_2 \| \| x_1 \oplus x_3)$	0	0	$f_V(dv_3 \| \| x_1 \oplus x_2)$	

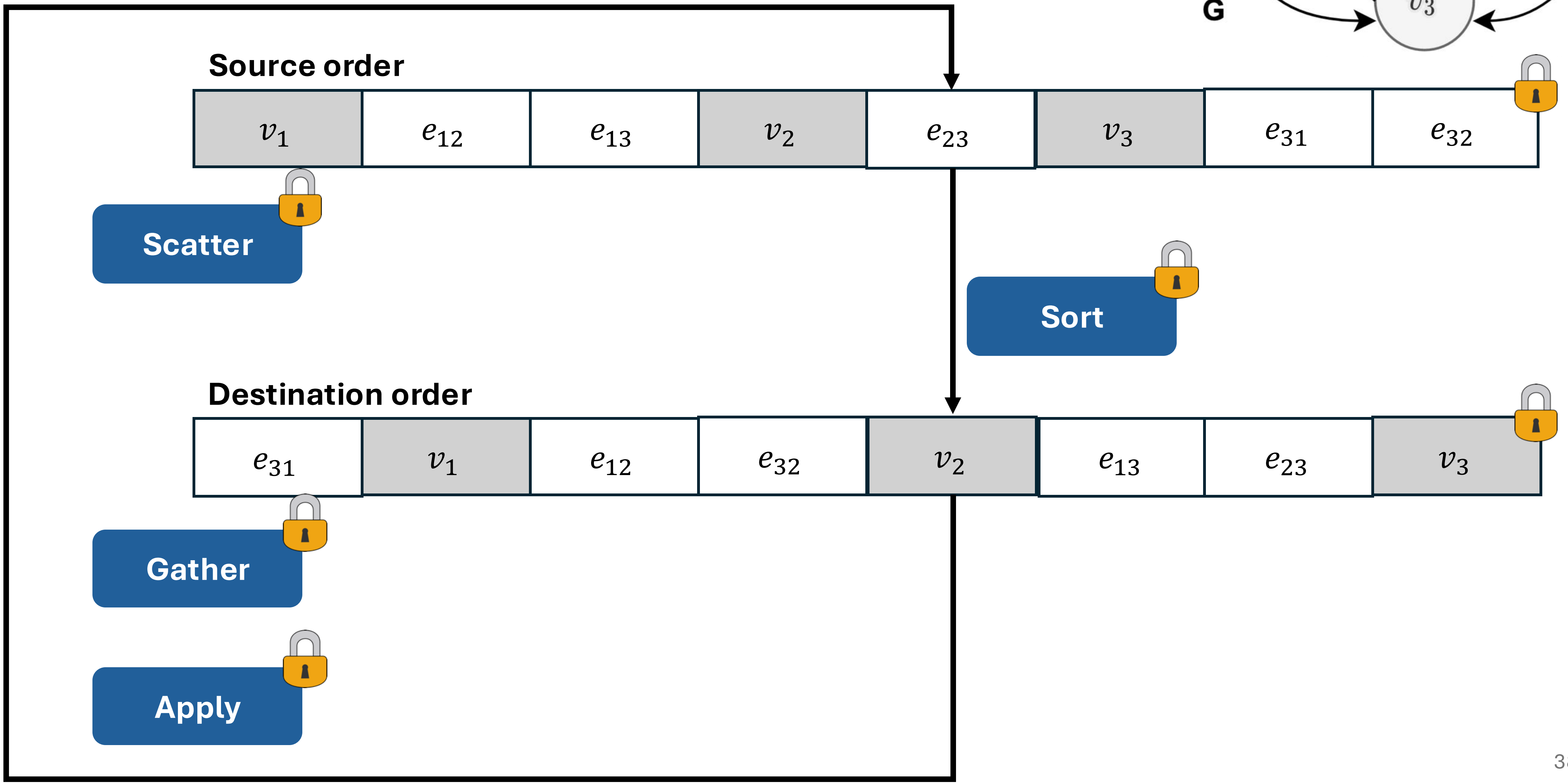
Message Passing Rounds





Secure Message Passing Rounds





GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

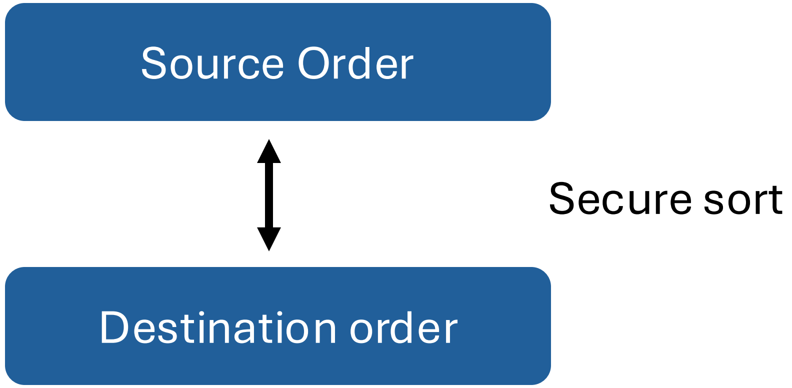
- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list

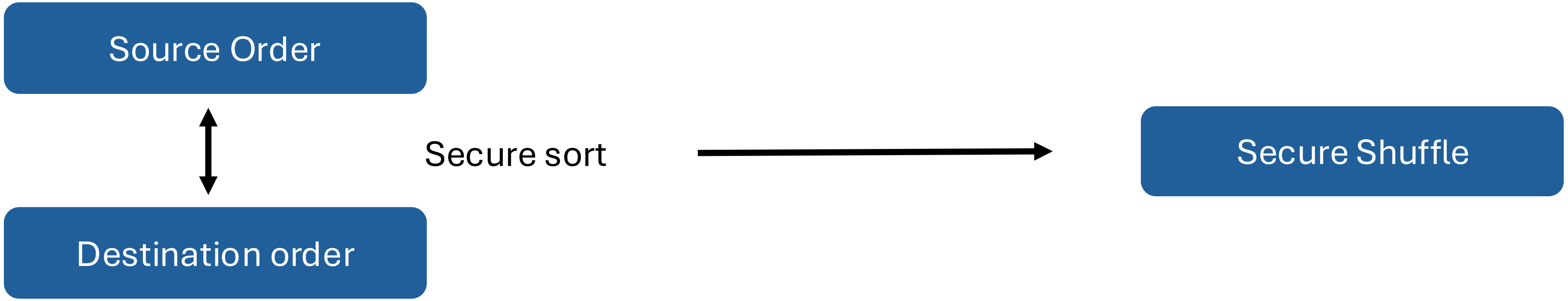


[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list



[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list
     - Secure shuffle: protocols with round complexity independent of the DAG list length

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list
     - Secure shuffle: protocols with round complexity independent of the DAG list length

At each entry in DAG list
- Rounds to compute $f(.) : O(f)$
- Rounds to obliviously update edge data depending on whether its a node or an edge : $O(1)$ at each entry of DAG list

Sequential linear scan
- $\boldsymbol{O}(\boldsymbol{N}.\boldsymbol{f}): N = |V| + |E|$

→

Parallelize linear scan
- $\boldsymbol{O}(\log(\boldsymbol{N}).\boldsymbol{f})$ Rounds

Scatter

$\mathrm{e.data} = f(\mathrm{e.data}, \mathrm{u.data}) \forall$ e(u, v) ∈ E

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.

GraphSC[AFO+15]

- Rely on secret sharing based MPC
  - Improved communication and computation
  - High on rounds

- Steps that impact rounds
  1. Rounds required for scan of the DAG list during Scatter-Gather
  2. Rounds to transition between source and destination order of DAG list
     - Secure shuffle: protocols with round complexity independent of the DAG list length

At each entry in DAG list
- Rounds to compute $f(.) : O(f)$
- Rounds to obliviously update edge data depending on whether its a node or an edge : $O(1)$ at each entry of DAG list

Sequential linear scan
- $\boldsymbol{O}(\boldsymbol{N}.\boldsymbol{f})$: $N = |V| + |E|$

→

Parallelize linear scan
- $\boldsymbol{O}(\log(\boldsymbol{N}).\boldsymbol{f})$ Rounds

Graphiti:

- New improved realizations of Scatter-Gather
- Rounds independent of DAG list length

[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." *ACM CCS, 2021*.

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

$\mathrm{e.data} = \mathrm{e.data} \mid\mid \mathrm{u.data}\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$

$\mathrm{e.data} = f(\mathrm{e.data})\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$

Scatter → **Propagate** + **ApplyE**

$\mathrm{e.data} = f(\mathrm{e.data}, \mathrm{u.data})\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$

[KKPR24] Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, and **Bhavish Raj Gopal**. "Graphiti: Secure Graph Computation Made More Scalable." ***ACM CCS, 2024***.

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

$\mathrm{e.data} = \mathrm{e.data} \,||\, \mathrm{u.data}\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$

$\mathrm{e.data} = f(\mathrm{e.data})\ \forall\ \mathrm{e(u, v)} \in \mathrm{E}$

Scatter $\longrightarrow$ **Propagate** + **ApplyE**

$O(N \cdot f)$ $\longrightarrow$ $O(N)$ + $O(f)$

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

Scatter → **Propagate** + **ApplyE**

$O(N \cdot f)$ → $O(N)$ + $O(f)$

New approach for **Propagate** with rounds independent of N

$O(N) \rightarrow O(1)$

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

Scatter $\longrightarrow$ **Propagate** **ApplyE**

$O(N \cdot f)$ $\longrightarrow$ $O(N)$ + $O(f)$

$O(N) \longrightarrow O(1)$

New approach for **Propagate** with rounds independent of N

<u>Scatter</u>

- Multiplications to distinguish between edge/vertex operations
- **Interactive**

<u>Propagate</u>

- Additions/subtractions
- **Non-interactive**

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

New approach for **Propagate** with rounds independent of N

New approach for **Gather** with rounds independent of N

Propagate $O(1)$		**ApplyE** $O(f)$
Gather $O(1)$	+	**ApplyV** $O(f_v)$

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

New approach for **Propagate** with rounds independent of N

New approach for **Gather** with rounds independent of N

New ordering of DAG list called **vertex order**

Graphiti: Results

Decouple **Scatter** into **Propagate** + **ApplyE**

New approach for **Propagate** with rounds independent of N

New approach for **Gather** with rounds independent of N

New ordering of DAG list called **vertex order**

Scatter

Source order

Interactive sequential operations

Transition

Vertex order → Source order

Non-interactive operations

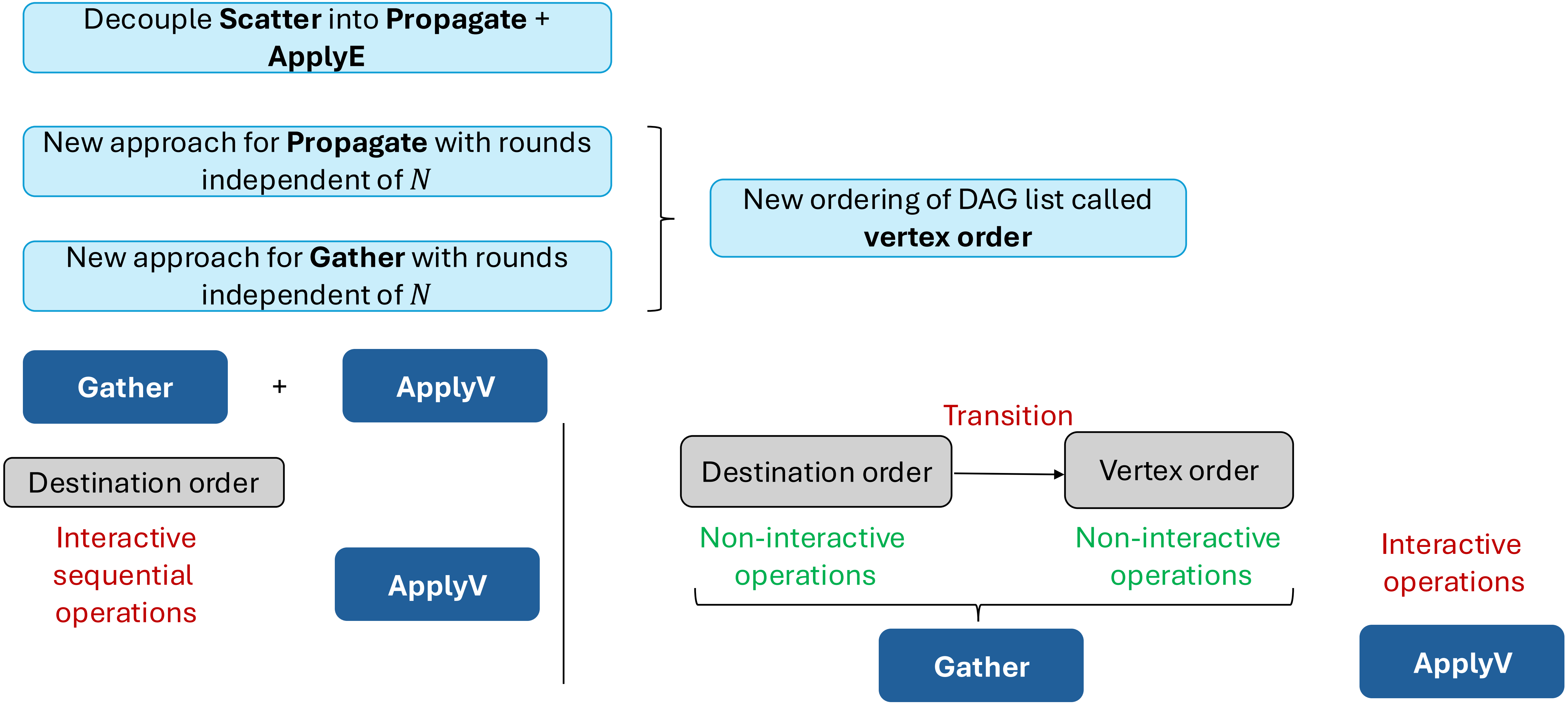
Non-interactive operations

Propagate

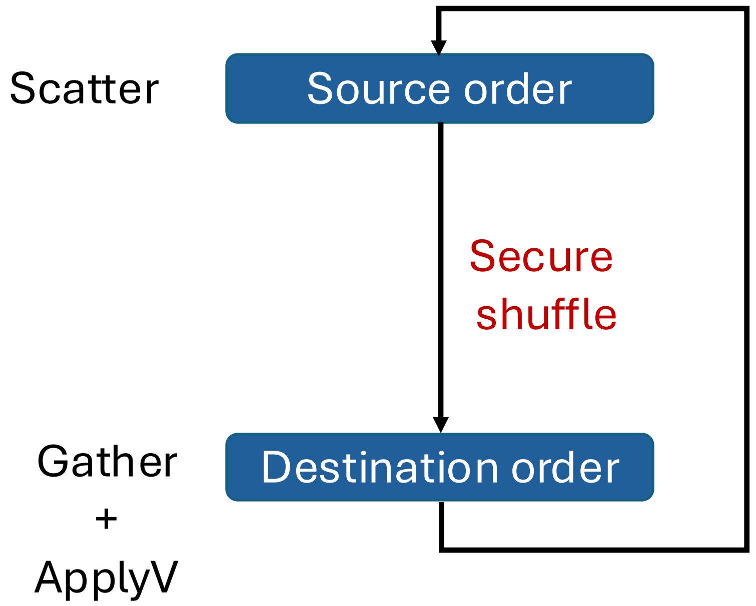
Interactive operations

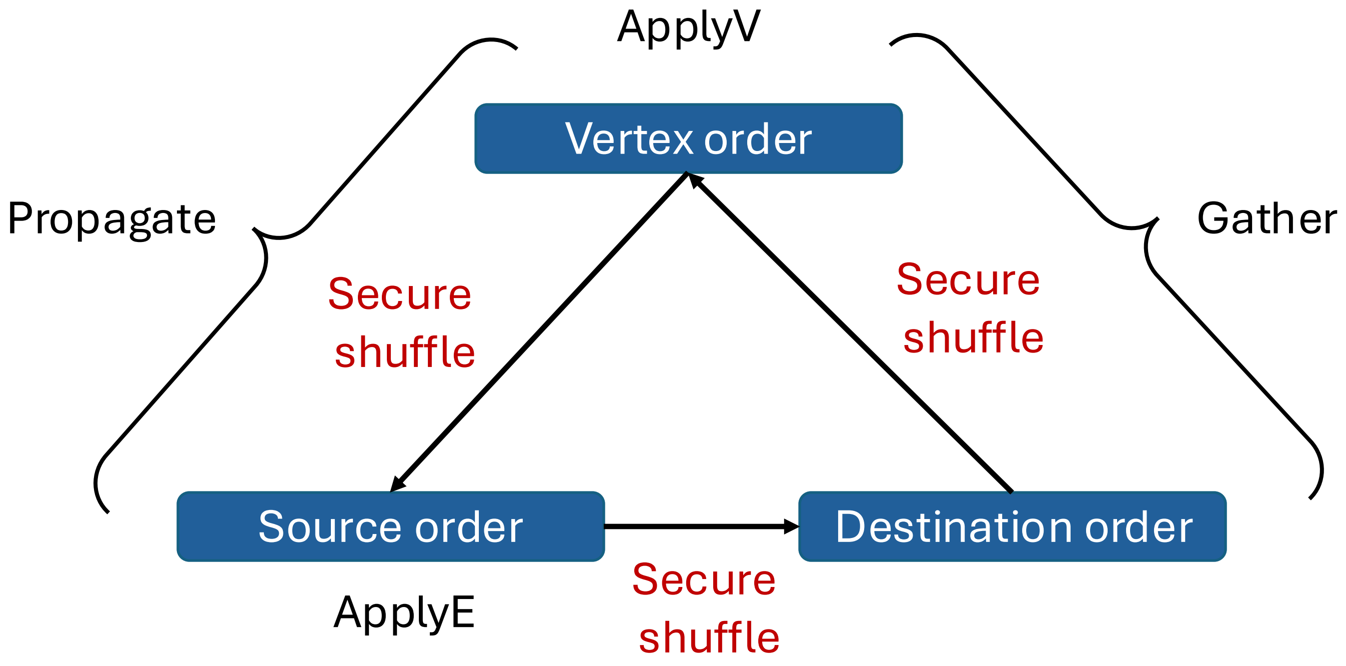
ApplyE

Graphiti: Results

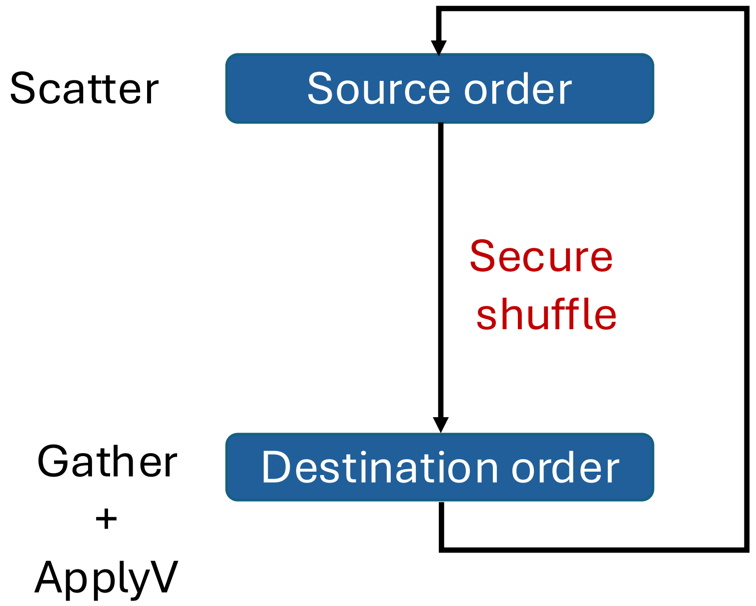


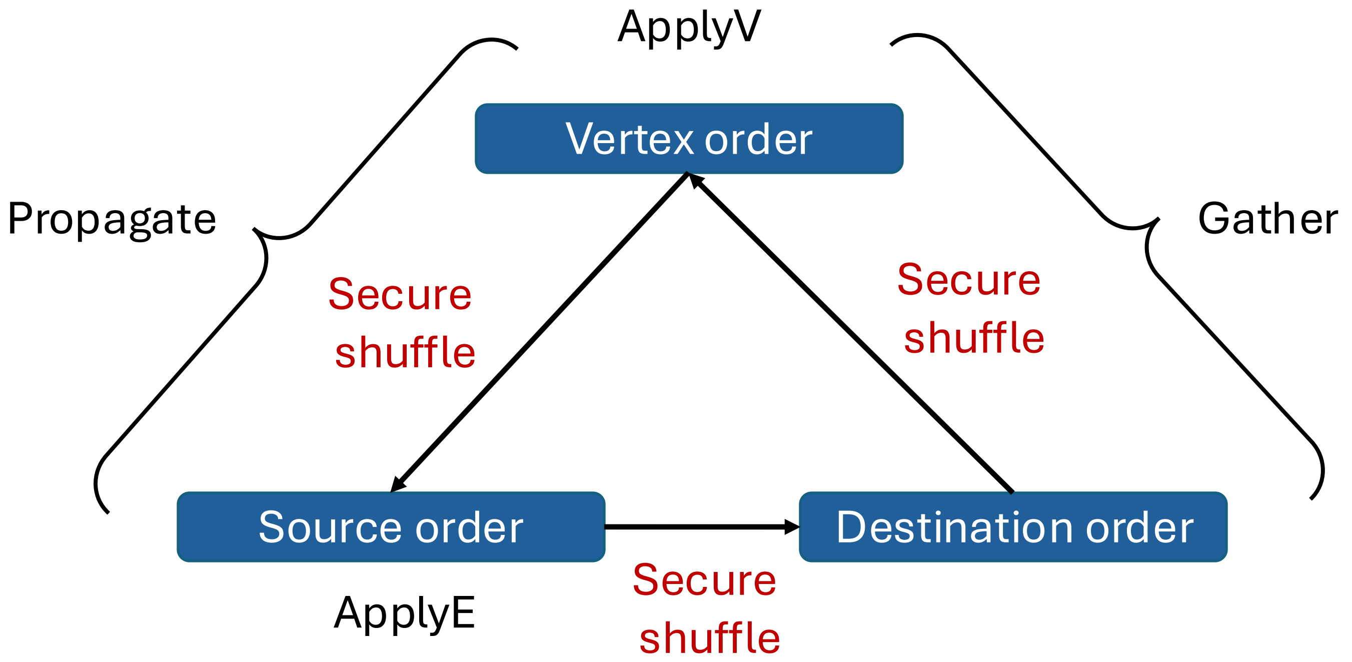
Graphiti: Comparison





Graphiti: Comparison





Framework	Rounds	Communication
Adjacency matrix	$O(1)$	$O(\|V\|^2)$
GraphSC[AFO+21]	$O(\mathrm{N})$	$O(\mathrm{N})$
GraphSC[AFO+21] (RO)	$O(\log(N))$	$O(\mathrm{N} \cdot \log(\mathrm{N}))$
Graphiti[KKPR24]	$O(1)$	$O(\mathrm{N})$

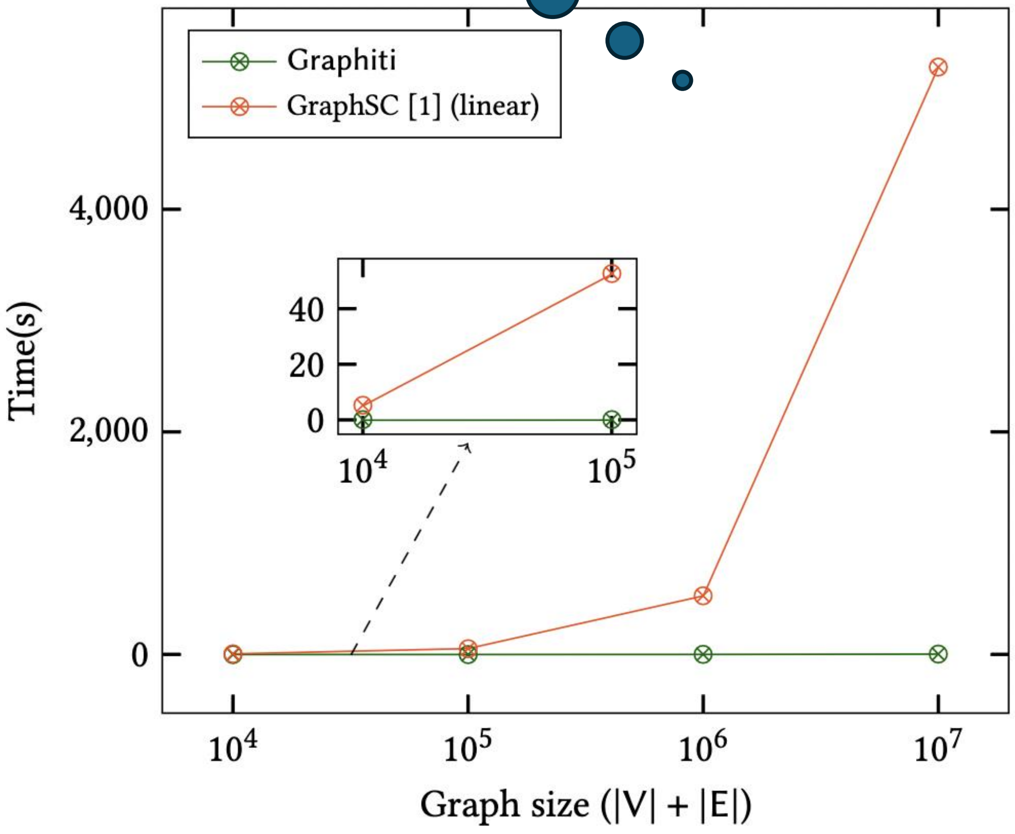
V: Vertices
E: Edges
N = |V|+|E|

Comparison for one message passing iteration of BFS

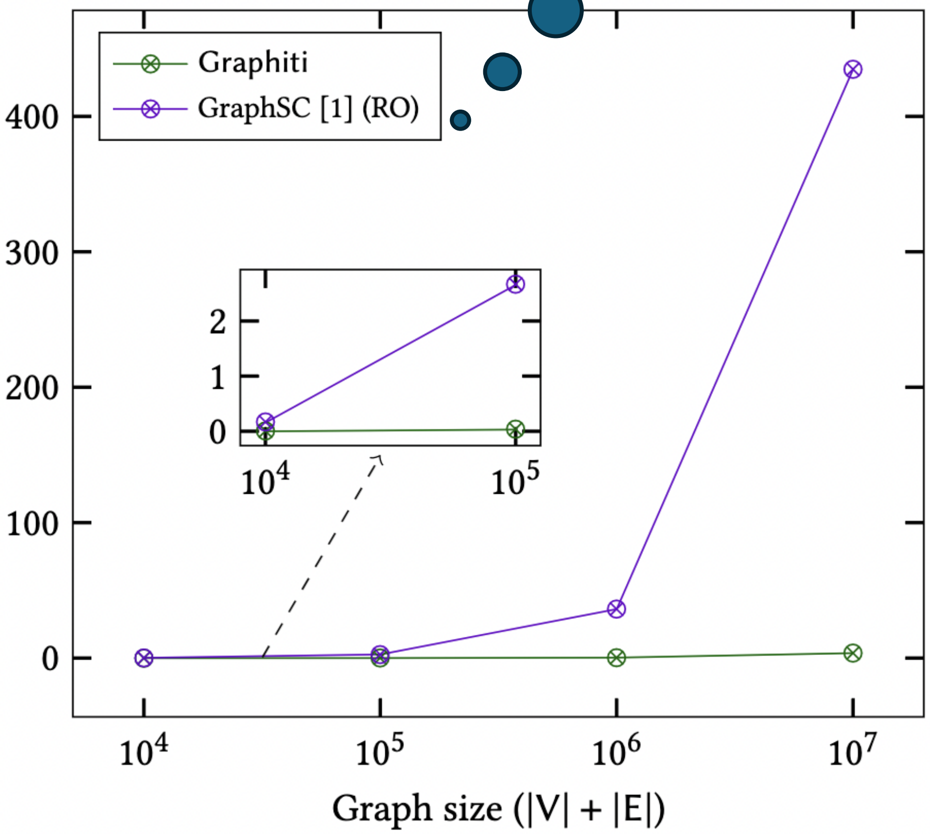
[AFO+21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. "Secure graph analysis at scale." ***ACM CCS, 2021***.
[KKPR24] Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, and **Bhavish Raj Gopal**. "Graphiti: Secure Graph Computation Made More Scalable." ***ACM CCS, 2024***.

Benchmarks (Runtime)

1034x improvement



85x improvement

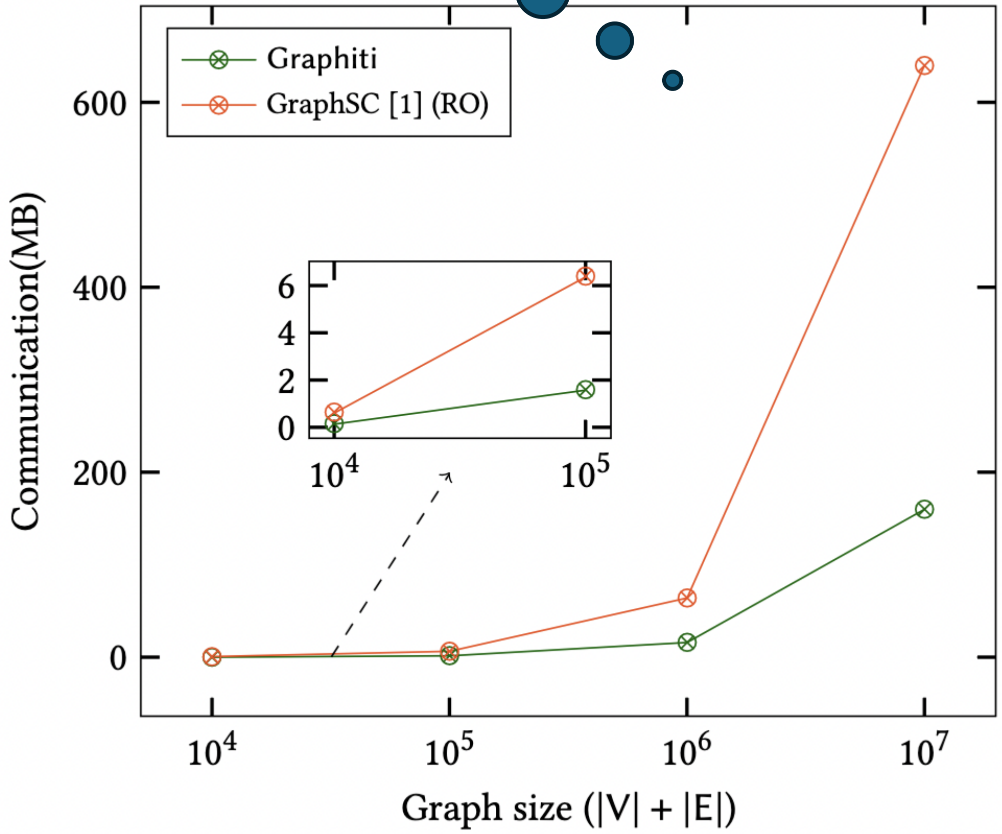


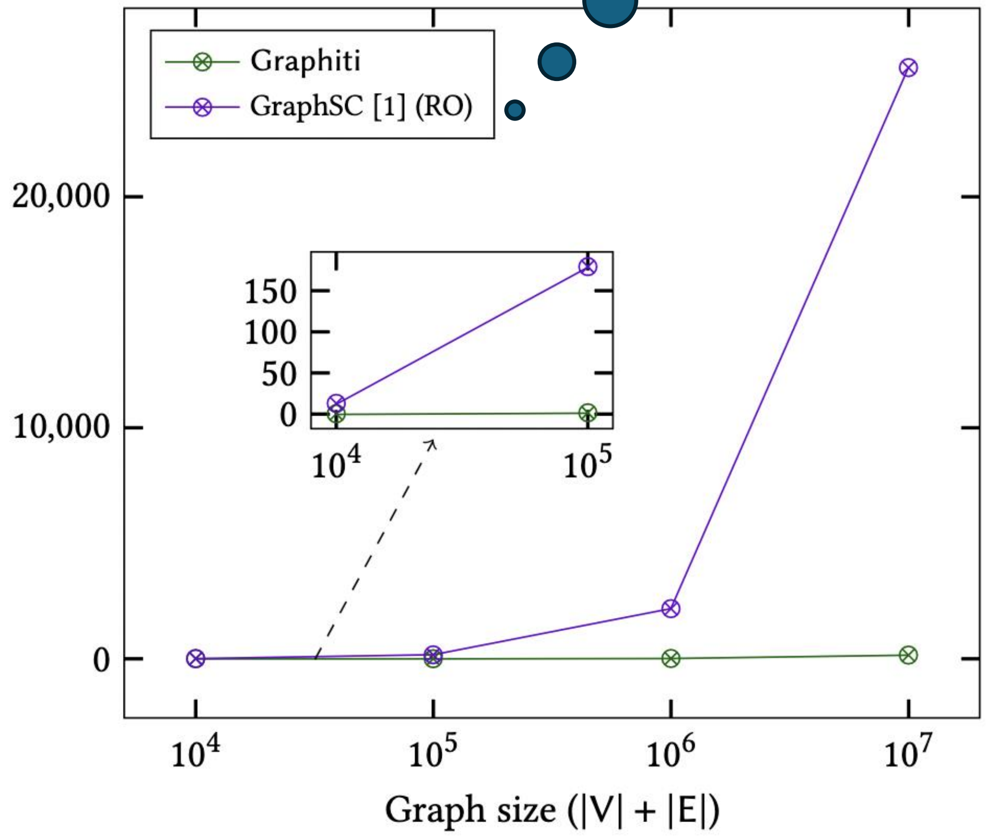
Graphiti and GraphSC instantiated in the semi-honest 2 party outsourced computation setting in LAN environment.

Benchmarks (Communication)

3x improvement

106x improvement





Graphiti and GraphSC instantiated in the semi-honest 2 party outsourced computation setting in LAN environment.

Future Directions

Extending Graphiti

- Dynamic graphs
- Multigraphs

Other Settings

- Client-Server
- N-party (non outsourced computation) setting

Thank You