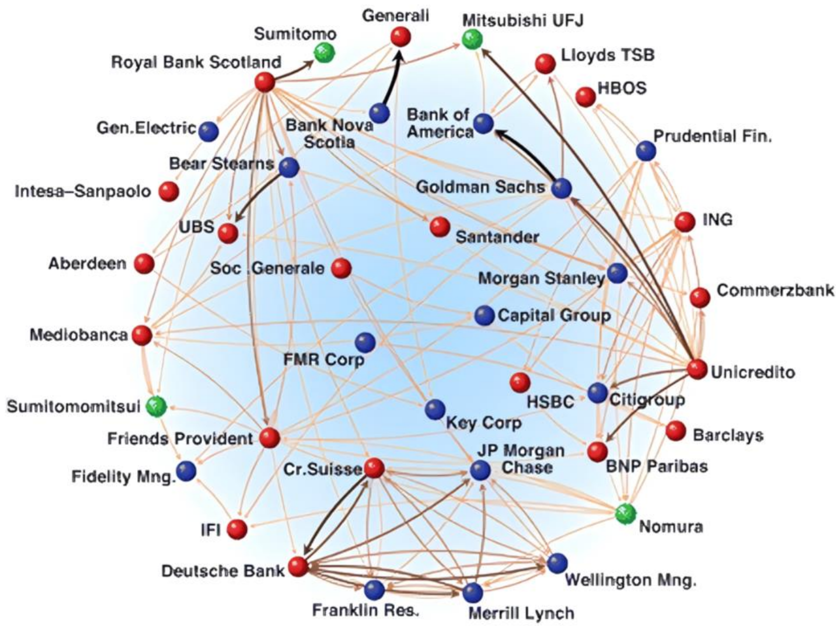
OblivGNN: Oblivious Inference on Transductive and Inductive Graph Neural Network

Zhibo Xu[1,2], Shangqi Lai[2], Xiaoning Liu[3], Alsharif Abuadbba[2], Xingliang Yuan[4], and Xun Yi[3]
[1]*Monash University*, [2]*CSIRO's Data61*, [3]*RMIT University*, [4]*The University of Melbourne*

33RD USENIX SECURITY SYMPOSIUM

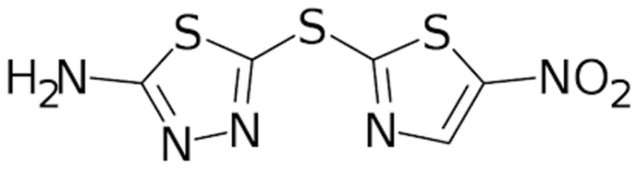
GNN Tasks

Node Classification
(Graph Convolutional Network [Kipf *et al.* (ICLR'17)])

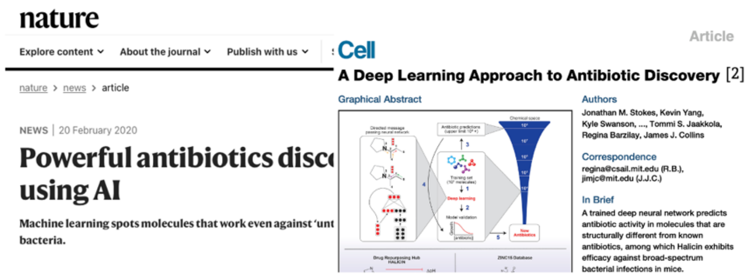


Bank

Graph Classification
(GraphSAGE [Hamilton *et al.* NIPS'17])

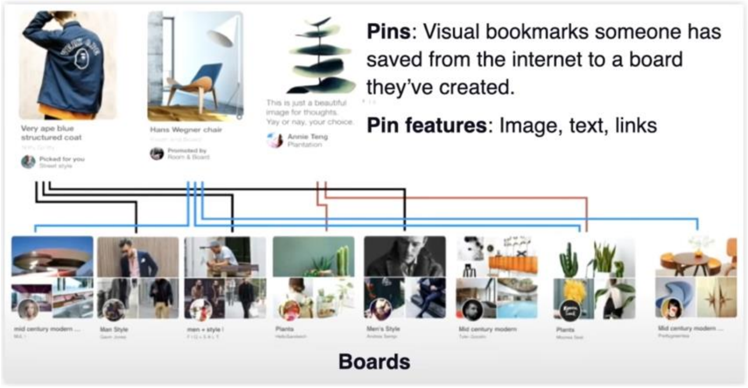


Halicin [3]



Drug discovery

Link Prediction
(GraphSAGE [Hamilton *et al.* NIPS'17])



Recommendation systems

Graph Neural Networks are specifically designed neural architectures operated on graph-structure data

--*Graph Neural Networks: Foundations, Frontiers, and Applications* by Lingfei et al.

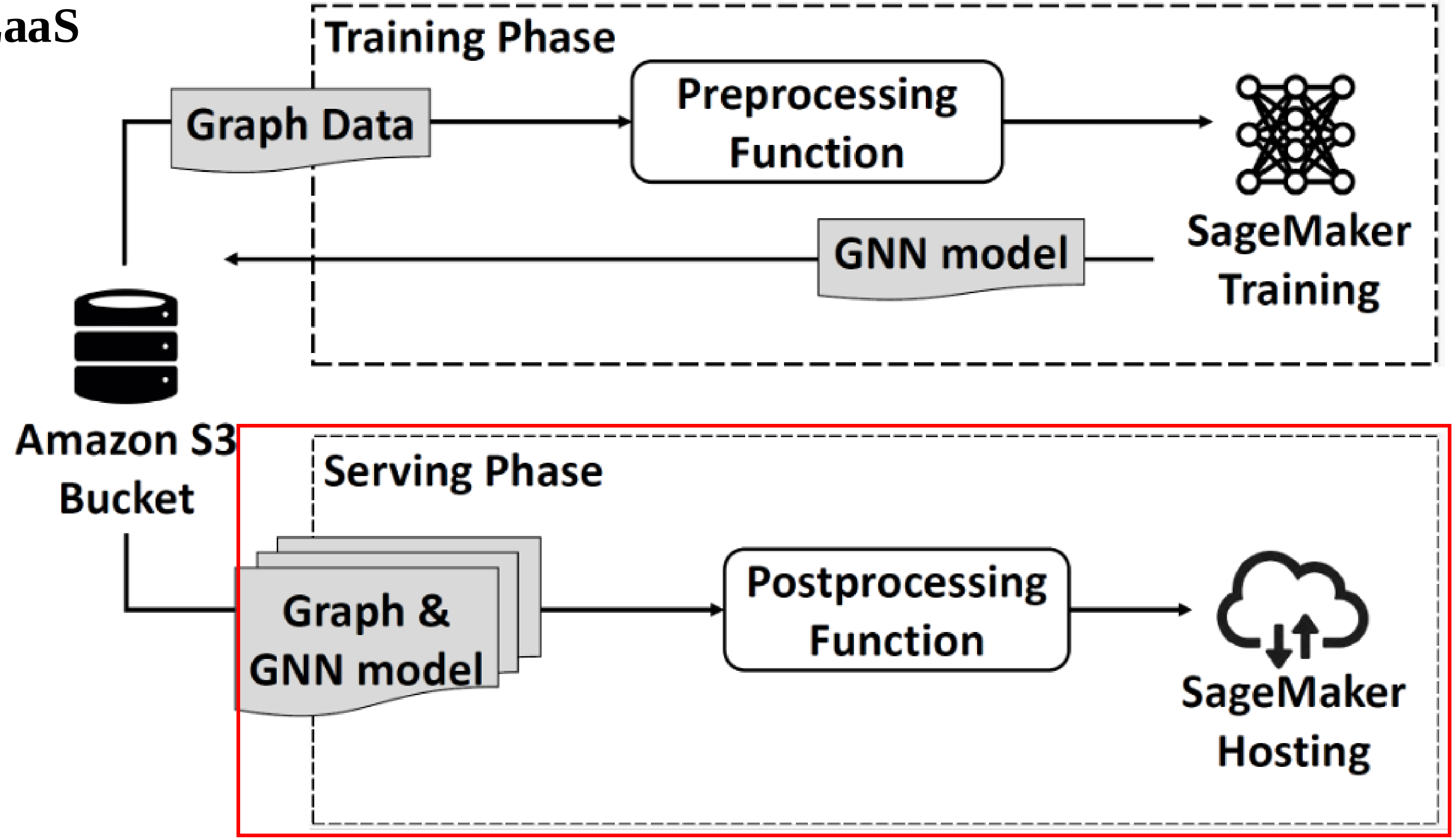
GNNs in Machine Learning as a Service (MLaaS)

GNN is increasingly featured on MLaaS platforms

Amazon: SageMaker Support for DGL

Google: Neo4j & Google Cloud Vertex AI

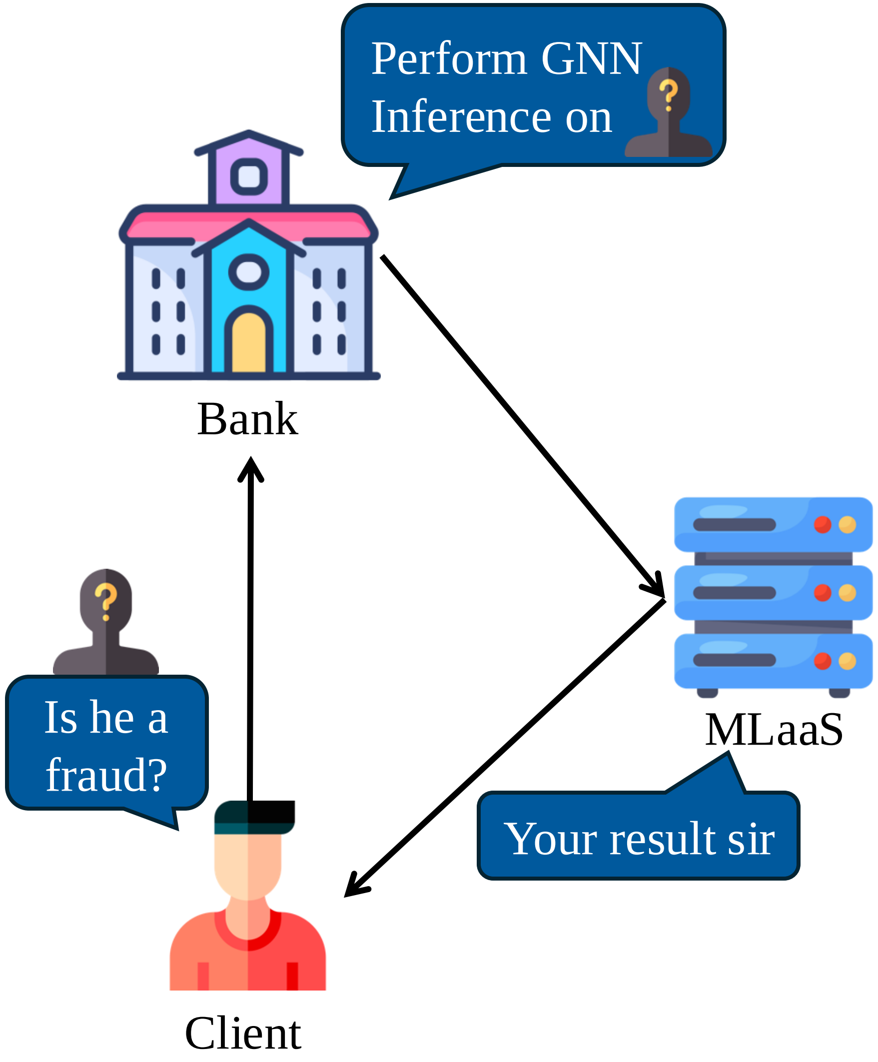
Microsoft: Azure ML Spektral



AWS SageMaker for GNN training and inference

Build a GNN-based real-time fraud detection solution using Amazon SageMaker, Amazon Neptune, and the Deep Graph Library: https://aws.amazon.com/blogs/machine-learning/build-a-gnn-based-real-time-fraud-detection-solution-using-amazon-sagemaker-amazon-neptune-and-the-deep-graph-library/

Privacy Concerns



Privacy Concerns:

- Expose sensitive training/inference graph to MLaaS
  - Collecting training graphs often requires a large amount of human, computing, and economic resource
  - Graph data is sensitive by nature, e.g., users' financial transactions, private friendships
- Expose proprietary GNN model parameters and node features to MLaaS

Related Works in PPML

Traditional PPML Frameworks

Trident, Chameleon, Falcon, GAZELLE, MiniONN, Delphi, ABY^3, SecureML, BLAZE, XONN, AriaNN, CryptGPU, SecureNN

- Cannot support graph-structured data

Existing Secure GNN Frameworks

SecGNN, CryptoGCN, LinGCN

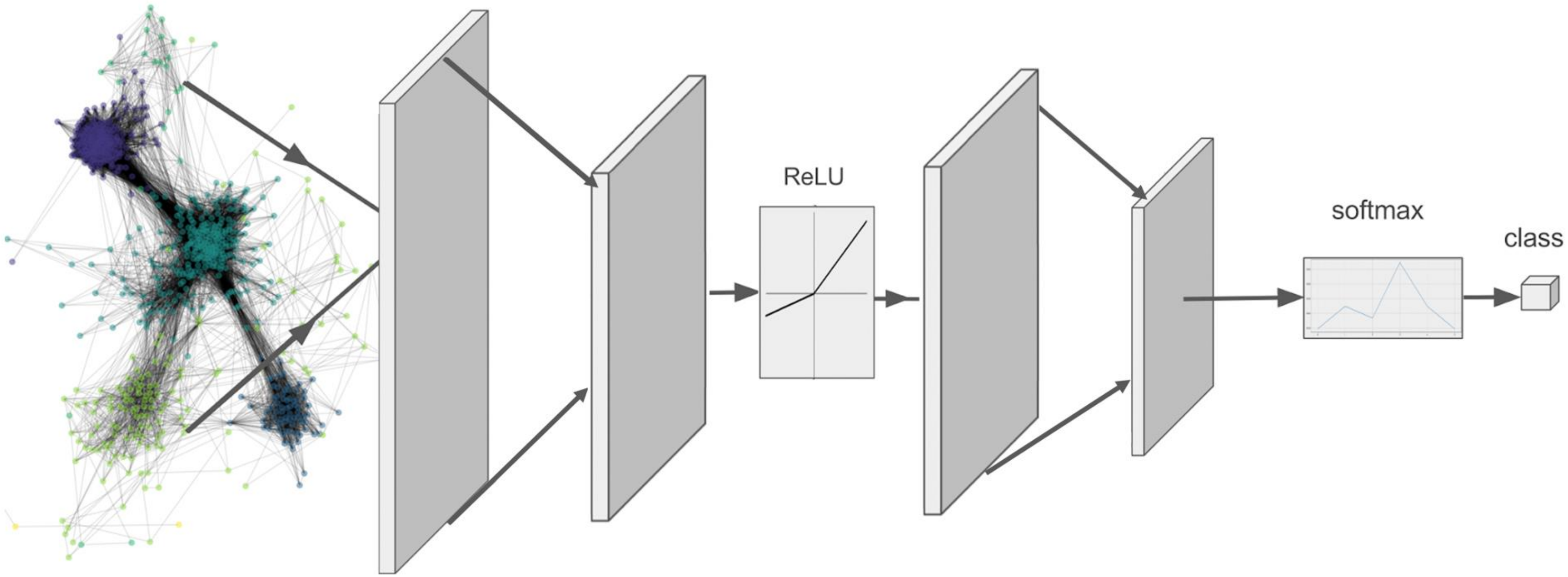
- Do not offer full protection of graph structure information
- Leak degree information
- Not support the diverse settings of GNN deployment
- Heavy computation cost (via FHE), heavy communication cost (via ASS) due to the large size of the graph

GNNs

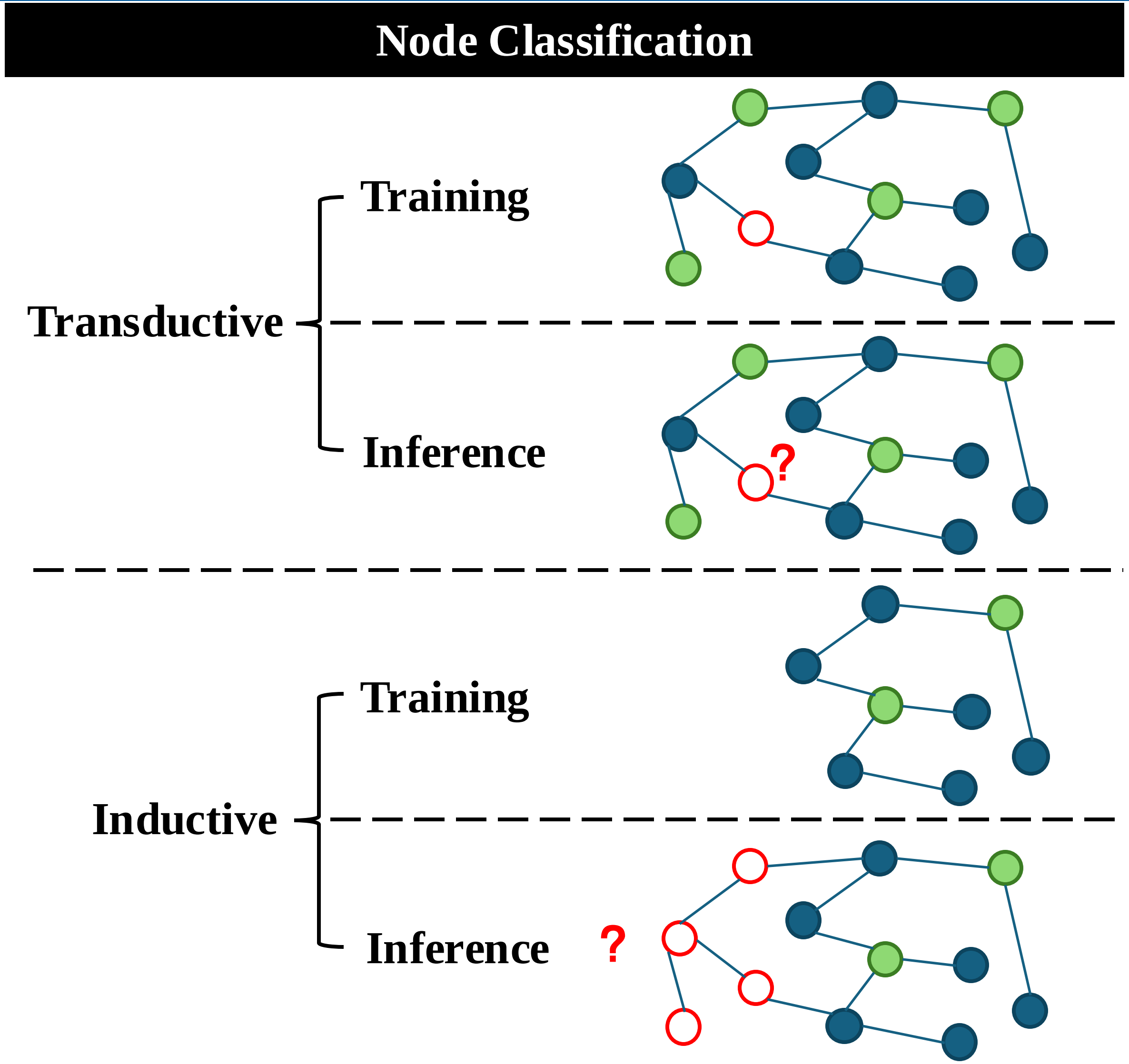
$\mathbf{Z}$=Softmax($\hat{\mathbf{A}}$ ReLU($\hat{\mathbf{A}}\hat{\mathbf{F}}\mathbf{W_0}$)$\mathbf{W_1}$)

- $\mathbf{W_0}$ and $\mathbf{W_1}$ are two trainable weights
- $\hat{\mathbf{A}}$ is the symmetric normalized adjacency matrix
- $\hat{\mathbf{F}}$ is the normalized feature matrix

- Activation functions:
  - $\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
  - Softmax: $z_i = \frac{e^{x_i}}{\sum_{j\in[1,C]} e^{x_i}}, i \in [1, C]$



GNN Settings



Transductive:

- Unlabelled nodes and their connections exist in the training
- Graph for training and inference remains the same

Inductive:

- *New/updated* nodes, features, connections appear in the *inference*

Function Secret Sharing

Function Secret Sharing

Distributed Point Functions:

$\text{KeyGen}(\alpha, \beta) \rightarrow k_0, k_1$

$\text{Eval}(k_b, x) \rightarrow [\![y]\!]_b$

$$\text{Eval}(k_0, x) + \text{Eval}(k_1, x) = \begin{cases} \beta, if x = \alpha \\ 0, otherwise \end{cases}$$

Equality Test:

$\text{KeyGen}^{=}(\alpha = \gamma, \beta = 1) \rightarrow k_0^{=}, k_1^{=}$

$\text{Eval}^{=}(k_b^{=}, x) \rightarrow [\![y]\!]_b$

$$\text{Eval}^{=}(k_0, x') + \text{Eval}^{=}(k_1, x') = \begin{cases} y = 1, if x' = \gamma \\ 0 \quad, otherwise \end{cases}$$

Comparison:

$\text{KeyGen}^{<}(\alpha = \gamma, \beta = 1) \rightarrow k_0^{<}, k_1^{<}$

$\text{Eval}^{<}(k_b^{<}, x) \rightarrow [\![y]\!]_b$

$$\text{Eval}^{<}(k_0, x') + \text{Eval}^{<}(k_1, x') = \begin{cases} y = 1, if x' \leq \gamma \\ 0 \quad\quad, if x' > \gamma \end{cases}$$

Arithmetic FSS:

Multiplication:

$\text{KeyGen}^{\times}\left(\mathrm{g}^{\circ}, r_{in}^1, r_{in}^2, r_{out}\right) \rightarrow k_0^{\times}, k_1^{\times}$

$\text{Eval}^{\times}(k_b^{\times}, x_1', x_2') \rightarrow \mathrm{g}_b^{\circ}(x_1 \times x_2) + r_{out}$

$$\text{Eval}^{\times}(k_0^{\times}, x_1', x_2') + \text{Eval}^{\times}(k_1^{\times}, x_1', x_2') = x_1 \times x_2 + r_{out}$$

Addition:

$\text{KeyGen}^{+}\left(\mathrm{g}^{\circ}, r_{in}^1, r_{in}^2, r_{out}\right) \rightarrow k_0^{+}, k_1^{+}$

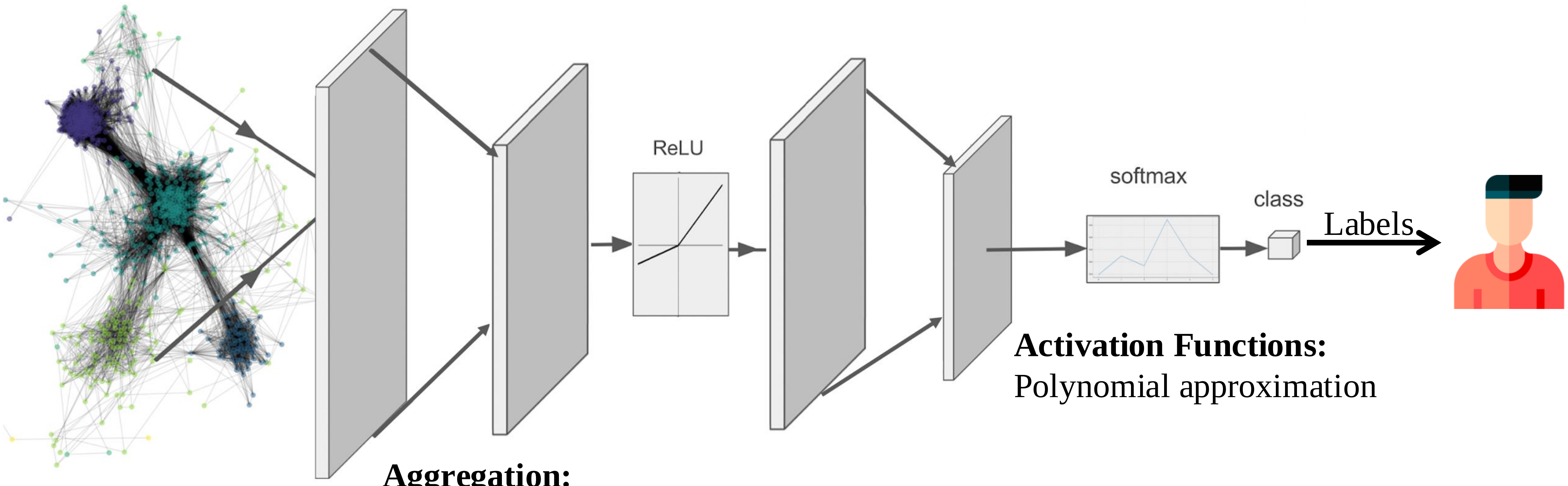
$\text{Eval}^{+}(k_b^{+}, x_1', x_2') \rightarrow \mathrm{g}_b^{\circ}(x_1 + x_2) + r_{out}$

$$\text{Eval}^{+}(k_0^{+}, x_1', x_2') + \text{Eval}^{+}(k_1^{+}, x_1', x_2') = x_1 + x_2 + r_{out}$$

Strawman Approach

Transductive setting

Sharing:
Additive Secret Sharing



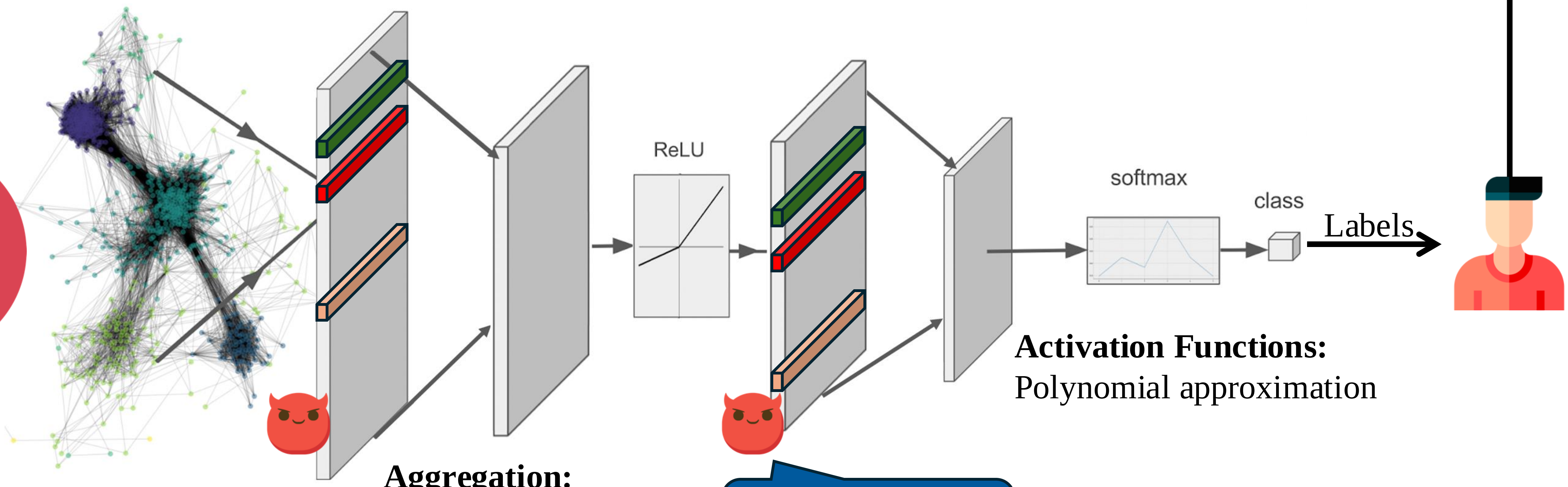
Activation Functions:
Polynomial approximation

Aggregation:
Beaver's triple

Strawman Approach

Inductive setting

Graph update:
update the graph



Activation Functions:
Polynomial approximation

Aggregation:
Beaver's triple

I can observe the update pattern

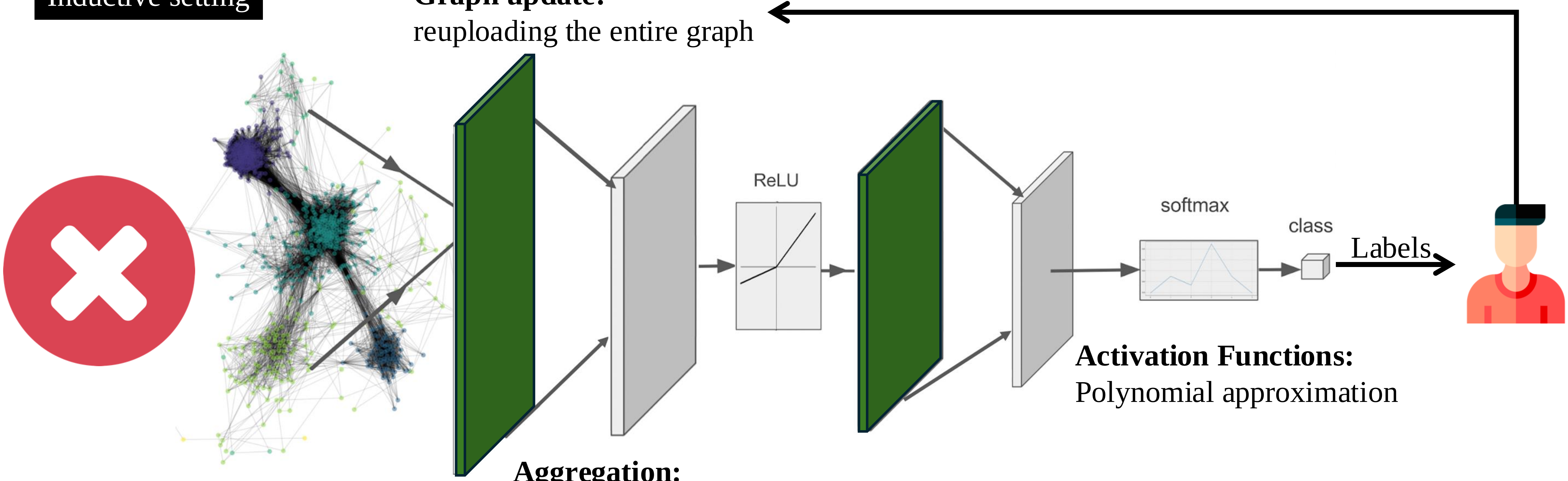
Problem:

1. Leak graph update access, suffering from leakage attack [Falzon and Paterson, ESORICS'22]

Strawman Approach

Inductive setting

Graph update:
reuploading the entire graph



Activation Functions:
Polynomial approximation

Aggregation:
Beaver's triple

Problems:
The communication/computation cost is significant when re-uploading the updated graph to update obliviously.

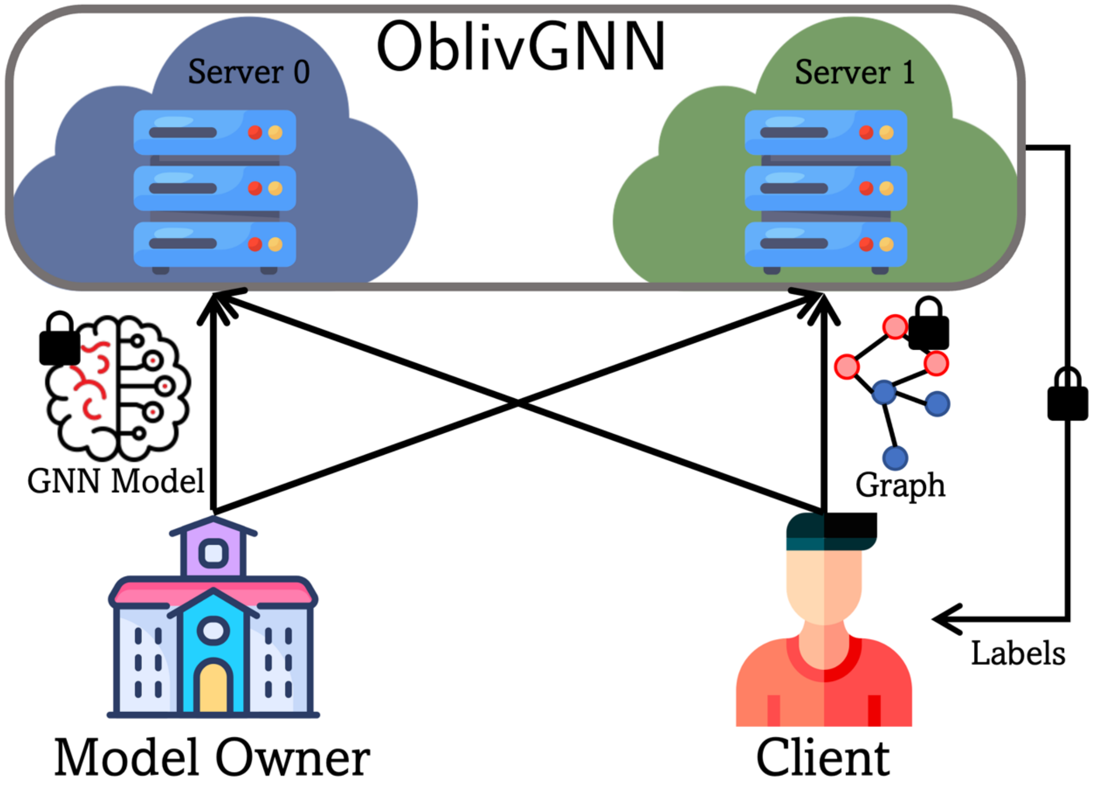
Research Questions

1. How to enable secure GNN inference for the *transductive* and *inductive* settings?

2. How to offer data *obliviousness* with semi-honest security?

3. How to *achieve* efficiency while achieving the above?

OblivGNN Architecture



- Semi-honest
- Outsource
- Non-colluding

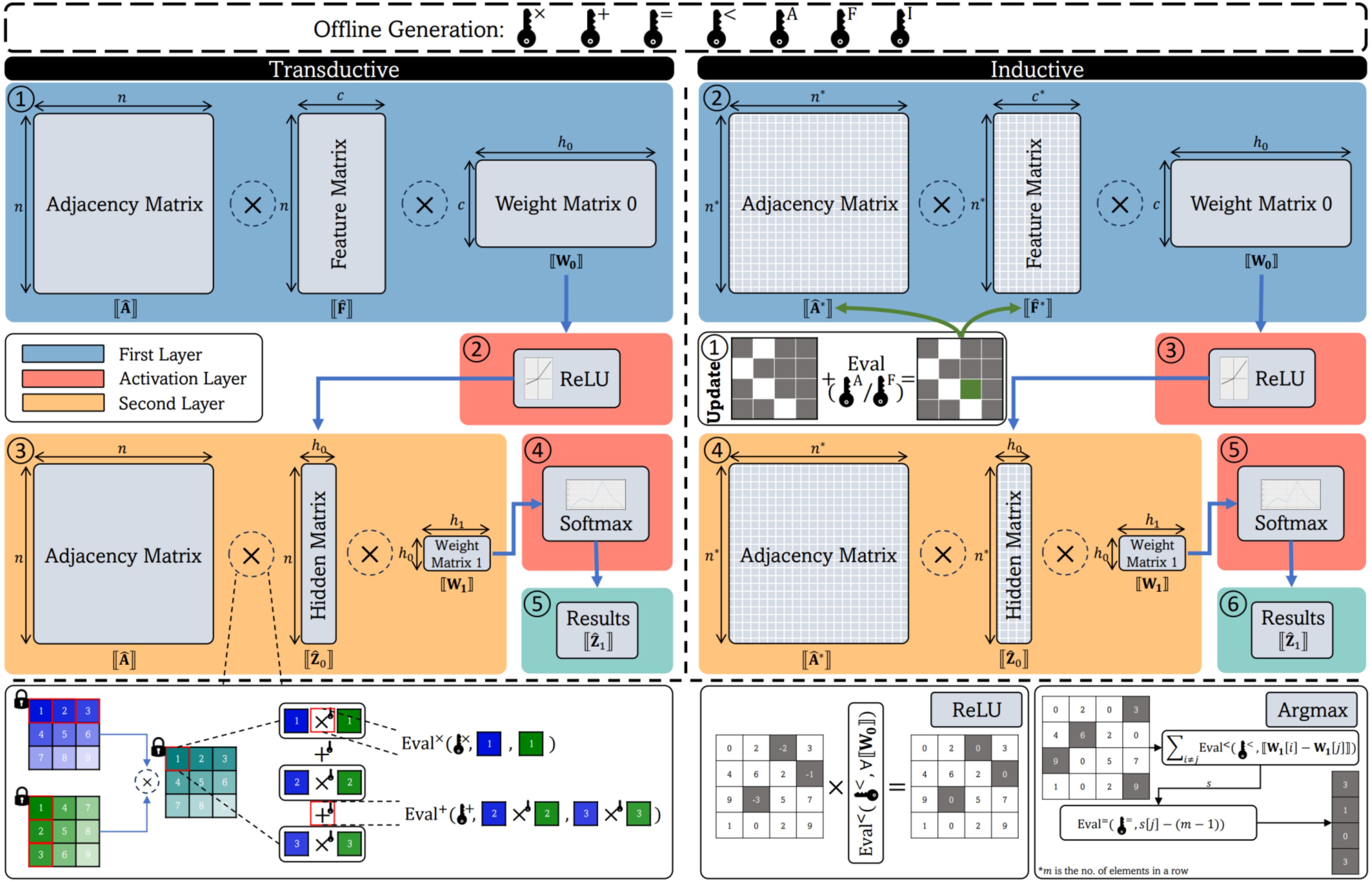
Security Guarantee

- Protect graph information
  - Adjacency Matrix $\widehat{\mathbf{A}}$
  - Feature Matrix $\widehat{\mathbf{F}}$
- Protect model information
  - Weight Matrix $\mathbf{W_0}$ and $\mathbf{W_1}$
- Protect access pattern to the graph structure and node feature
- Protect client query, intermediate results, and inference results

OblivGNN Approach

Offline

Online



OblivGNN Approach

Offline

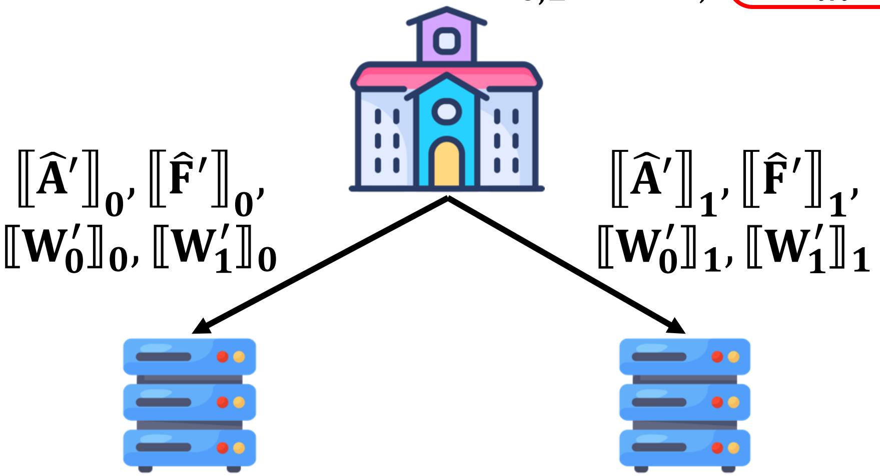
- <u>Masking & secret share GNN model</u>

Masks for Arithmetic FSS gates

Adjacency matrix: $\widehat{\mathbf{A}}' \leftarrow \widehat{\mathbf{A}} + r_{in}^1/r_{in}^2$

Feature matrix: $\hat{\mathbf{F}}' \leftarrow \hat{\mathbf{F}} + r_{in}^1/r_{in}^2$

Weight matrices: $\mathbf{W}_{0,1}' \leftarrow \mathbf{W}_{0,1} + r_{in}^1/r_{in}^2$



- Two servers need to *recover* the ASS shares before operating FSS circuits
- The linear layer results (in *shares*) will be used in oblivious activation functions
- The oblivious updates are performed on the shares

OblivGNN Approach

Offline

- Key generation

FSS Key Pool Generation

Multiplication:

$\text{KeyGen}^{\times}(g^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow \boxed{k_0^{\times}, k_1^{\times} : \text{FSS Multiplication keys}}$

$\text{Eval}^{\times}(k_b^{\times}, x_1', x_2') \rightarrow g_b^{\circ}(x_1 \times x_2) + r_{out}$

Addition:

$\text{KeyGen}^{+}(g^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow \boxed{k_0^{+}, k_1^{+} : \text{FSS Addition keys}}$

$\text{Eval}^{+}(k_b^{+}, x_1', x_2') \rightarrow g_b^{\circ}(x_1 + x_2) + r_{out}$

DPF Key Pool Generation

$\left.\begin{array}{l} k^{\mathrm{A}}\text{: DPF Node Update keys} \\ k^{\mathrm{F}}\text{: DPF Feature Update keys} \\ k^{\mathrm{I}}\text{: DPF Client Inquiry keys} \end{array}\right\}$ Online keys

$k^{=}$: DPF Equality Test keys

$k^{<}$: DPF Comparison keys

OblivGNN Approach

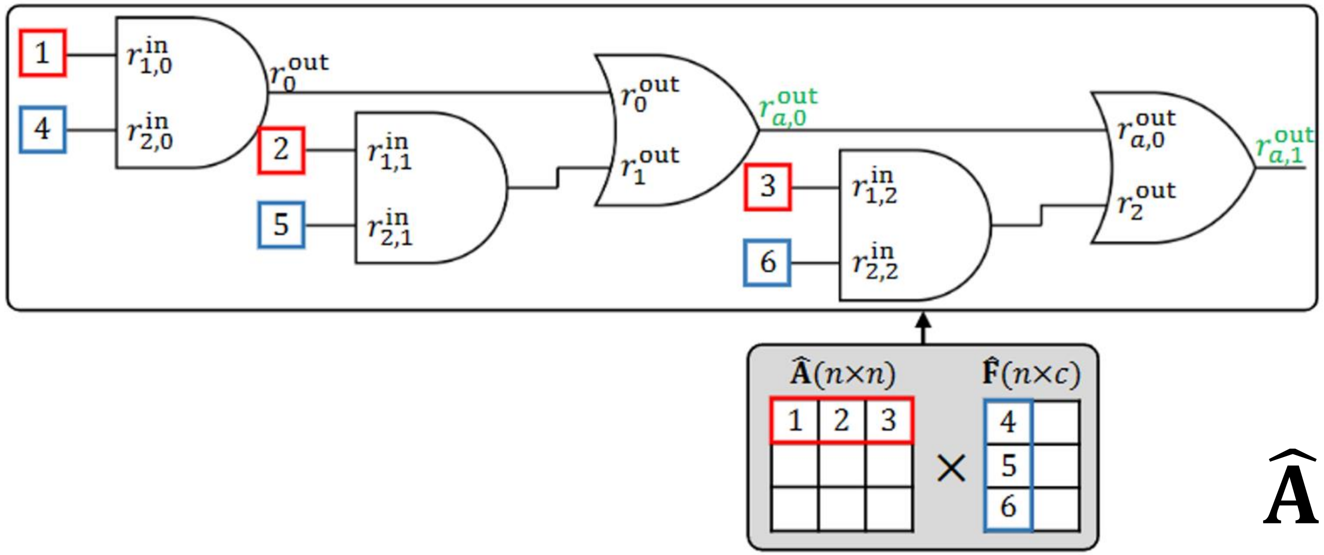
Online – Oblivious Aggregation

$$\mathrm{Eval}^{\times}(k_0^{\times}, x_1', x_2') + \mathrm{Eval}^{\times}(k_1^{\times}, x_1', x_2') = x_1 \times x_2 + r_{out}$$

$$\mathrm{Eval}^{+}(k_0^{+}, x_1', x_2') + \mathrm{Eval}^{+}(k_1^{+}, x_1', x_2') = x_1 + x_2 + r_{out}$$

Example:

$$x_1' = x_1 + r_{in}^1$$
$$x_2' = x_2 + r_{in}^2$$



$$\hat{\mathbf{A}} \times \hat{\mathbf{F}} \times \mathbf{W} \quad \ldots$$

$$\boxed{1} \times \boxed{4} = \mathrm{Eval}^{\times}(k^{\times}, \boxed{1}, \boxed{4})$$

$$\boxed{2} \times \boxed{5} = \mathrm{Eval}^{\times}(k^{\times}, \boxed{2}, \boxed{5})$$

$$\boxed{3} \times \boxed{6} = \mathrm{Eval}^{\times}(k^{\times}, \boxed{3}, \boxed{6})$$

$$\mathrm{Eval}^{+}(k^{+},\ )$$

$$\mathrm{Eval}^{+}(k^{+},\ )$$

$$= \text{output} + r_{out}$$

OblivGNN Approach – *Inductive* Exclusive Ops

Online – Oblivious Graph Update

New Node Insertion

- Introduce *new* nodes
- Do NOT modify the existing graph

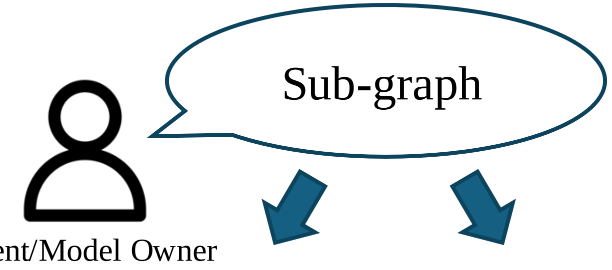
Existing Graph Update

- Modify the *existing* graph
  - Oblivious update adjacency matrix
  - Oblivious update feature matrix

OblivGNN Approach – *Inductive* Exclusive Ops

Online – Oblivious Graph Update

Existing Graph Update



Sub-graph Adjacency Matrix

Sub-graph Feature Matrix

$\mathrm{KeyGen}(0||1,1) \rightarrow k_0^{\mathrm{A}}, k_1^{\mathrm{A}}$

$\mathrm{KeyGen}(1||0,1) \rightarrow k_0^{\mathrm{A}}, k_1^{\mathrm{A}}$

$$
\begin{array}{c|cccc|}
 & 0 & 1 & \dots & n \\
\hline
0 & 1 & 0 & \dots & 0 \\
1 & 0 & 1 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots \\
n & 0 & 0 & \dots & 1 \\
\hline
\end{array}
+
\begin{array}{l}
\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, 0||0) \\
\leftarrow \boxed{\mathrm{Eval}(k_b^{\mathrm{A}}, 1||0)} \\
\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, \dots) \\
\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, n||0)
\end{array}
\Rightarrow
\begin{array}{c|cccc|}
 & 0 & 1 & \dots & n \\
\hline
0 & 1 & 1 & \dots & 0 \\
1 & 1 & 1 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots \\
n & 0 & 0 & \dots & 1 \\
\hline
\end{array}
$$

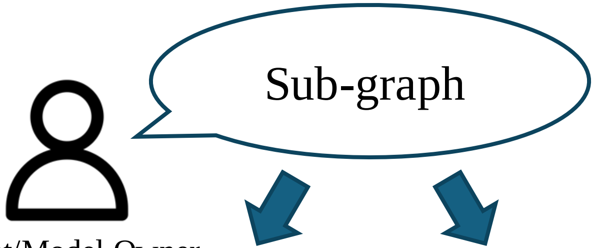
Adjacency Matrix $[\widehat{\mathbf{A}}]$ (secret shared)

Updated Adjacency Matrix

OblivGNN Approach – *Inductive* Exclusive Ops

Online – Oblivious Graph Update

Existing Graph Update



Sub-graph Adjacency Matrix

Sub-graph Feature Matrix

$\mathrm{KeyGen}(1||0, F_{\triangle}(v_i)) \rightarrow k_0^{\mathrm{F}}, k_1^{\mathrm{F}}$

$$\begin{array}{c} \\ 0 \\ 1 \\ \dots \\ n \end{array}\begin{array}{c} \begin{array}{ccc} 0 & \dots & c \end{array} \\ \begin{bmatrix} 1 & \dots & 0 \\ \boxed{1} & \dots & 1 \\ \dots & \dots & \dots \\ 0 & \dots & 1 \end{bmatrix} \end{array} + \begin{array}{l} \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, 0||0) \\ \leftarrow \boxed{\mathrm{Eval}(k_b^{\mathrm{F}}, 1||0)} \\ \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, \dots) \\ \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, n||0) \end{array} \Rightarrow \begin{array}{c} \\ 0 \\ 1 \\ \dots \\ n \end{array}\begin{array}{c} \begin{array}{ccc} 0 & \dots & c \end{array} \\ \begin{bmatrix} 1 & \dots & 0 \\ \boxed{0} & \dots & 1 \\ \dots & \dots & \dots \\ 0 & \dots & 1 \end{bmatrix} \end{array}$$

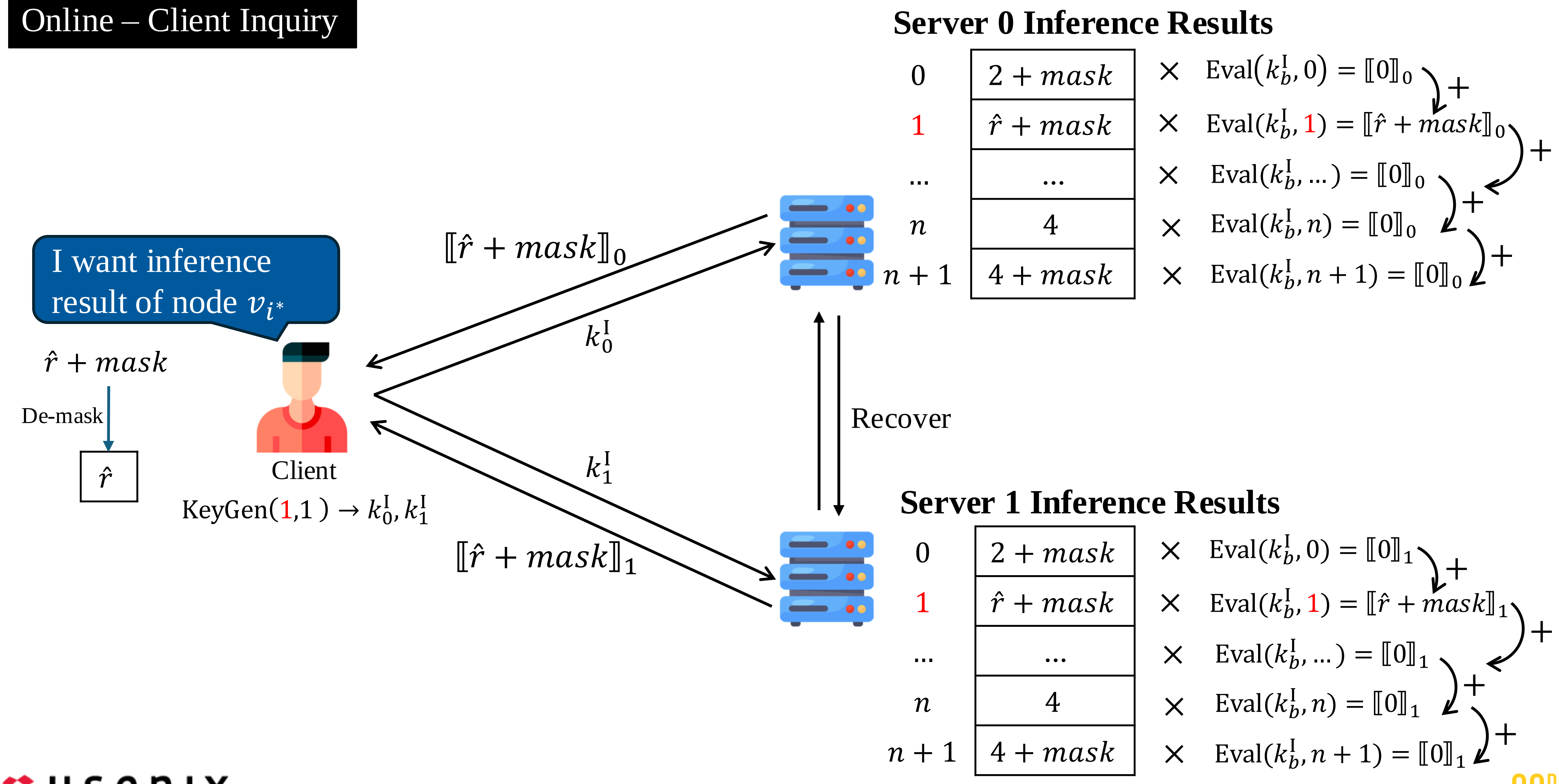
Feature Matrix [$\hat{\mathbf{F}}$] (secret shared)

Updated Feature Matrix

By employing DPF keys to perform graph updates, we now achieve *oblivious* and *efficient* graph updates

OblivGNN Approach

Online – Client Inquiry



Experiments

- **Platform**
  - Python and C++
  - Server
    - 3.70GHz Intel(R) Xeon(R) E-2288G CPU
    - 64GB RAM and 128GB external storage
    - Ubuntu 20.04.5 LTS
  - MP-SPDZ [Keller et al. (CCS'20)]

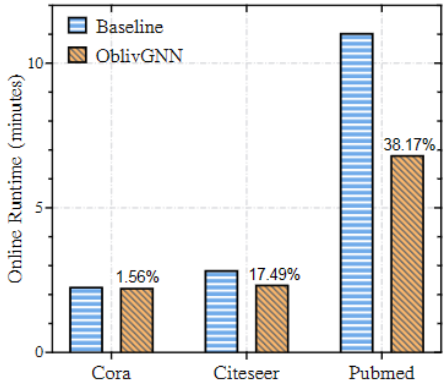
- **Datasets**
  - Cora, Citeseer and Pubmed

Dataset	Nodes	Feature	Edge	Classes
Cora	2708	1433	5429	7
Citeseer	3327	3703	4732	6
Pubmed	19717	500	44338	3

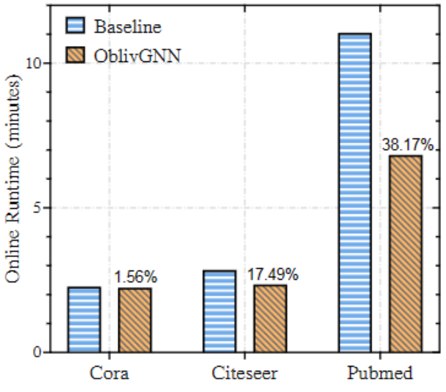
- **Baseline**
  - Baseline: pure additive secret shares for inference.
  - OblivGNN: additive secret shares with FSS for oblivious inference.

Experiments – System

System Runtime: Average Reduction: 38%



(a) Transductive

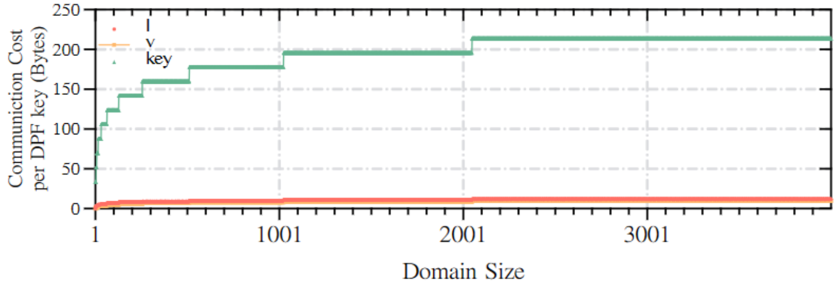


(b) Inductive

System Communication (GB): Reduction: 10× - 151×

	Baseline	**OblivGNN**
Cora	34.21	**0.29**
Citeseer	61.81	**0.41**
Pubmed	16.33	**1.65**

Graph Update Cost: Logarithm growth towards large graph



(a) Node/Feature Update Key Size

(b) Node/Feature Update Time Cost

Special thanks to my supervisors, Prof Xingliang Yuan and Dr Shangqi Lai, for their continuous support!
Many thanks to my collaborators for their invaluable contributions and comments!

Q&A

Email: Zhibo.Xu@monash.edu

SCAN ME