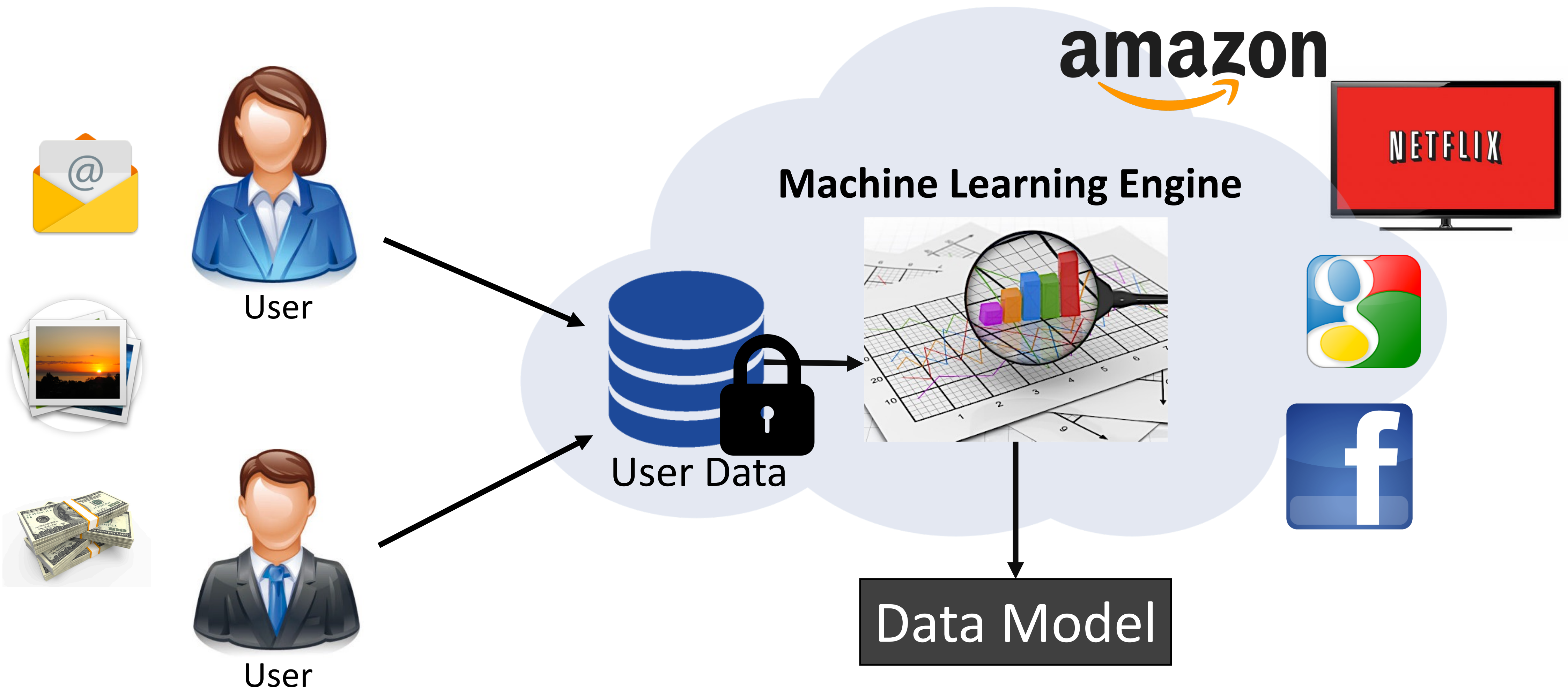
# Secure Parallel Computation on National-Scale Volumes of Data

**Sahar Mazloom**, Phi Hung Le, Samuel Ranellucci, S. Dov Gordon

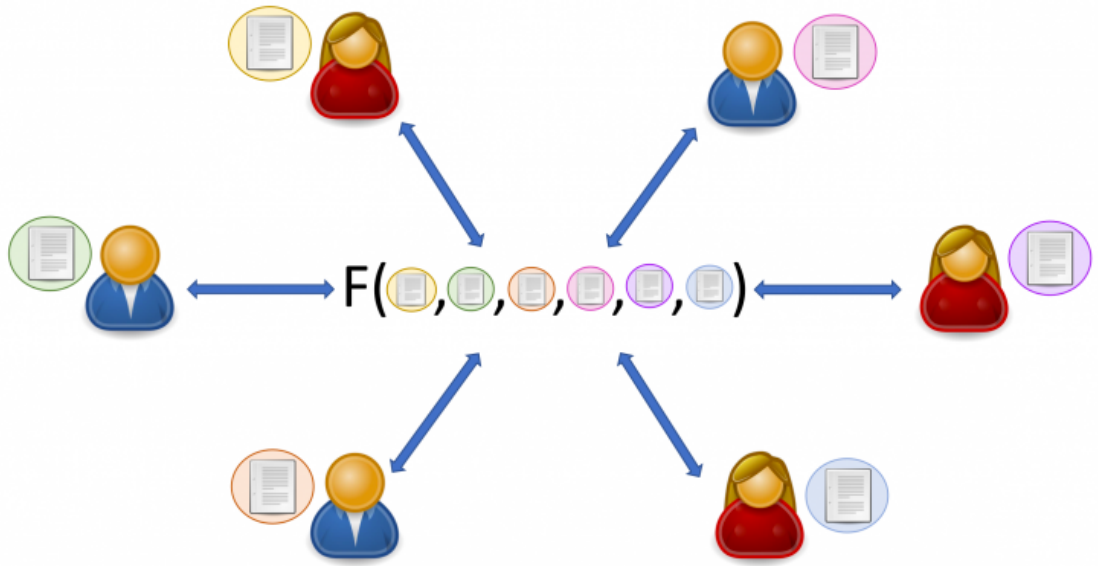
# Learning on **User Data**



# “Computation on Encrypted Data”

## Secure Multi-Party Computation:

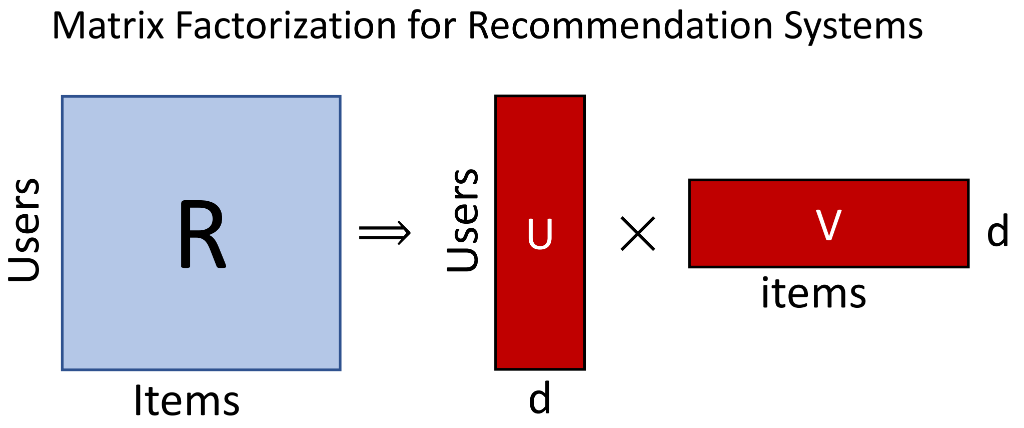
Goal: Creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.



# Example of a ML algorithm

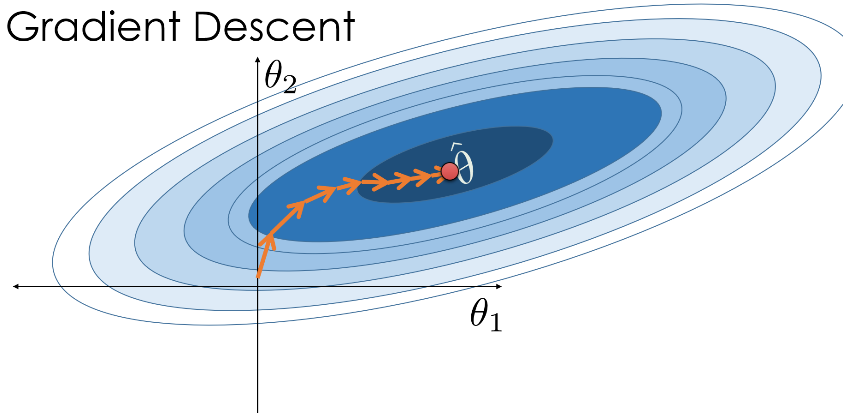
**(Sparse) Matrix Factorization:**

- De-composition of a sparse matrix of Ratings (**R**), into Users' (**U**), and Items' (**V**) matrices.

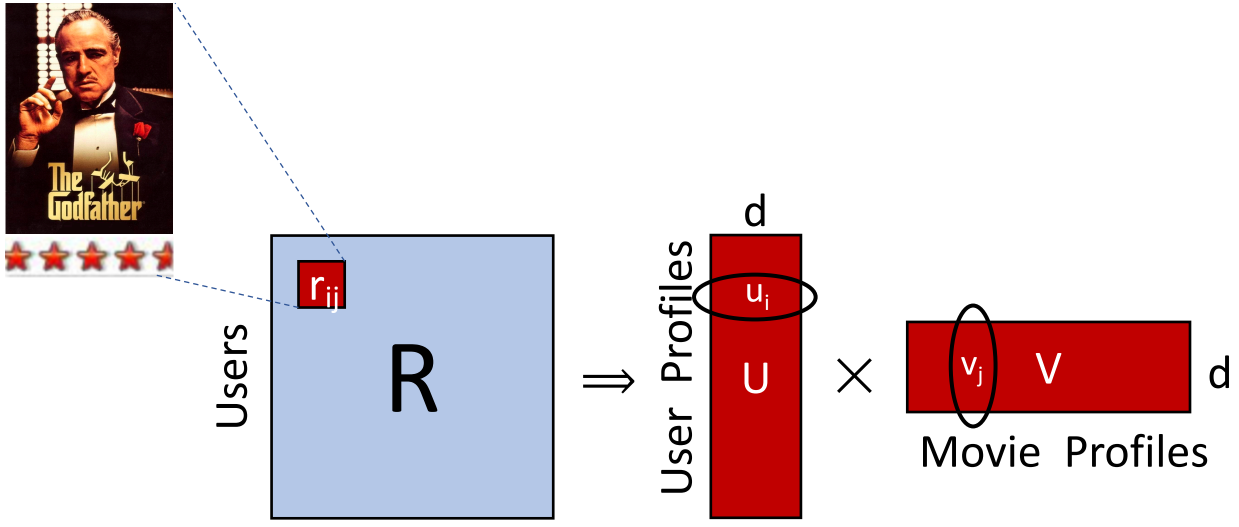



**Gradient Descend:**

- An optimization algorithm used in many machine learning algorithms.



# Matrix Factorization using **Gradient Descent** for **Movie Recommendation**




$$r_{ij} \approx \langle u_i, v_j \rangle$$

**Objective function:** $L = \min \sum_{(i,j)\epsilon M} \left(r_{i,j} - \langle u_i, v_j \rangle\right)^2 + \mu \sum_{i\epsilon n} \|u_i\|^2 + \lambda \sum_{j\epsilon m} \|v_j\|^2$

$$\begin{cases} \text{User gradient: } \delta_{u_i} = -2 \sum_{j\epsilon m} v_j \left(r_{ij} - \langle u_i, v_j \rangle\right) + 2\mu u_i \\ \text{Movie gradient: } \delta_{v_j} = -2 \sum_{i\epsilon n} u_i \left(r_{ij} - \langle u_i, v_j \rangle\right) + 2\lambda v_j \end{cases}$$

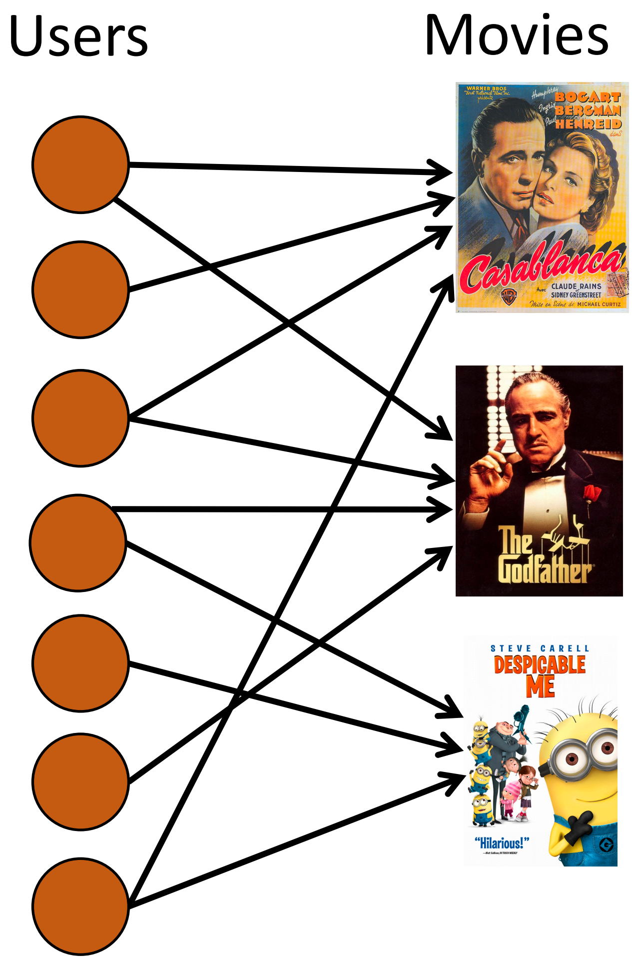
# Distributed Graph Parallel Computation

**<u>Non-secure</u> Frameworks:**

MapReduce, GraphLab,
PowerGraph [Gonzalez et al. 2012]

**Supported algorithms**:

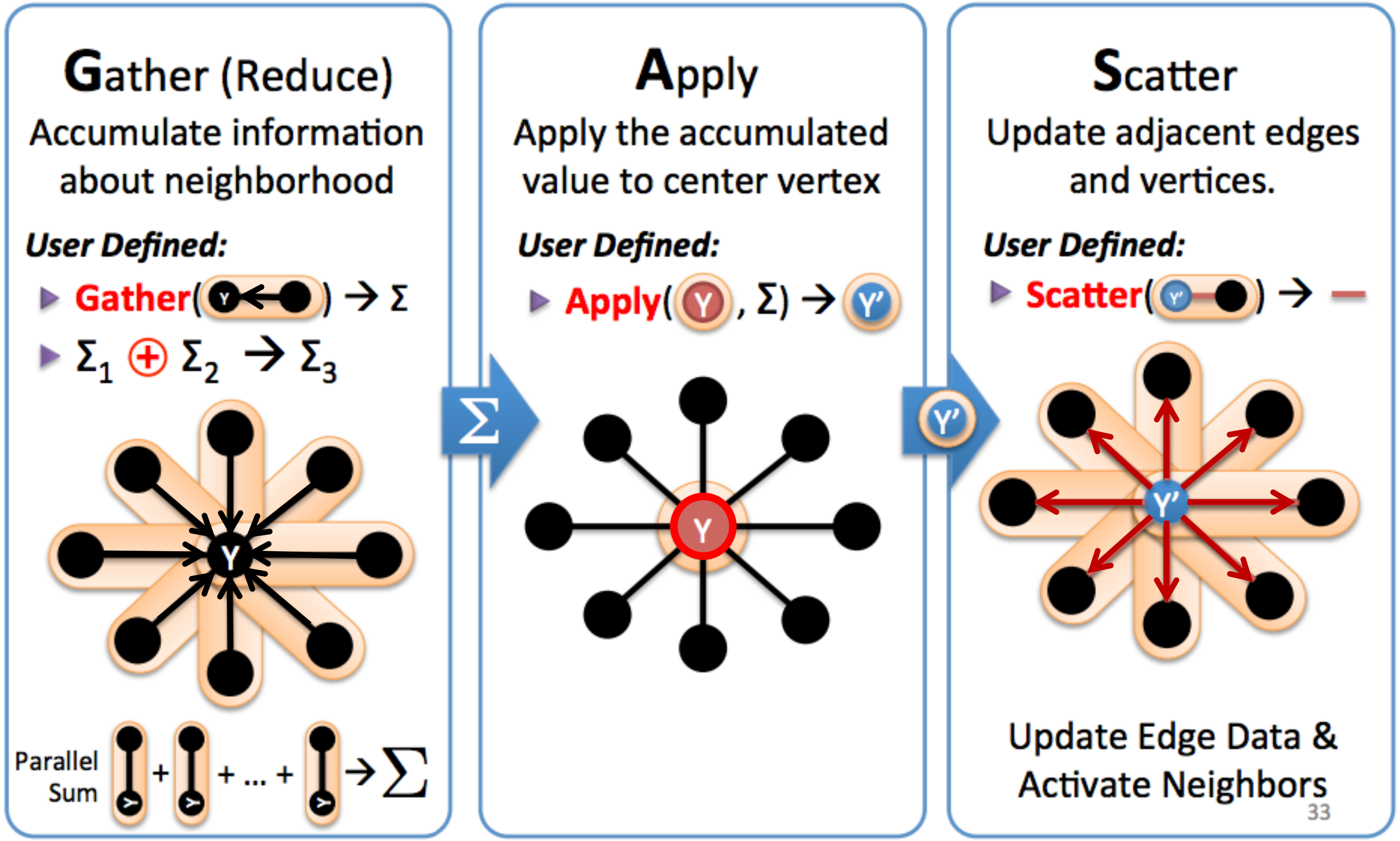
Matrix Factorization, Histogram, PageRank, Markov Random Field Parameter Learning, Name Entity Resolution, …



# **GAS** model of Operation

PowerGraph:
Think as a
Vertex!

Move computation
to data



# **Secure** Graph Parallel Computation

- **GraphSC** [Nayak et al. SP'15]

- Use **Oblivious Sort** to hide *node degree* and *edge structure*

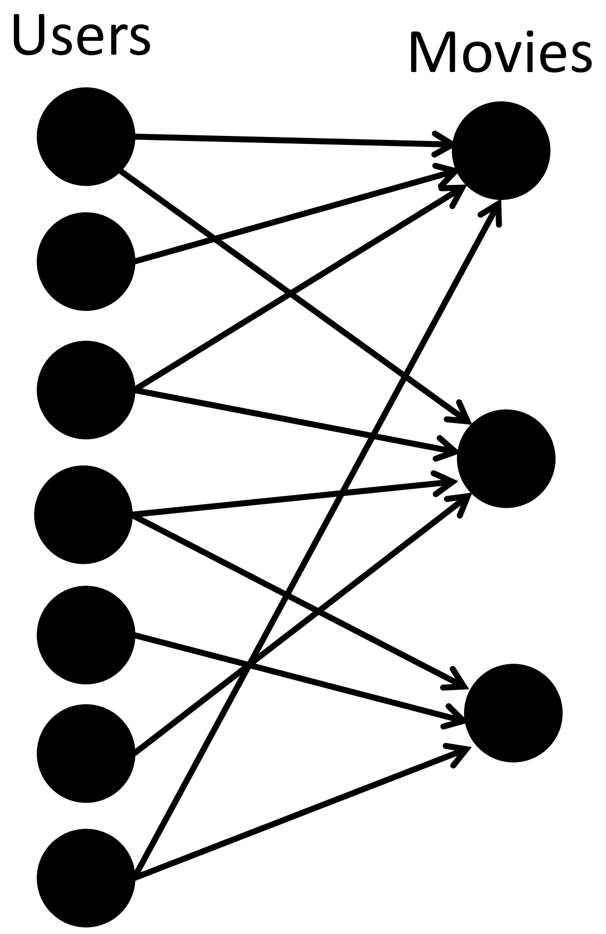
Complexity:

$$O((|E| + |V|) \mathbf{log^2}(|E| + |V|))$$

Running time:

6K users, 4K movies, 1 M Ratings => **13 Hrs**

Threat model: Honest-But-Curious Adversary




V vertices, E edges

# Primary Question [Mazloom, Gordon CCS'18]

Can we make secure computation algorithms **faster**

if we allow *something small* to be learned?

And prove the **leakage** **is** **Differentially Private!**

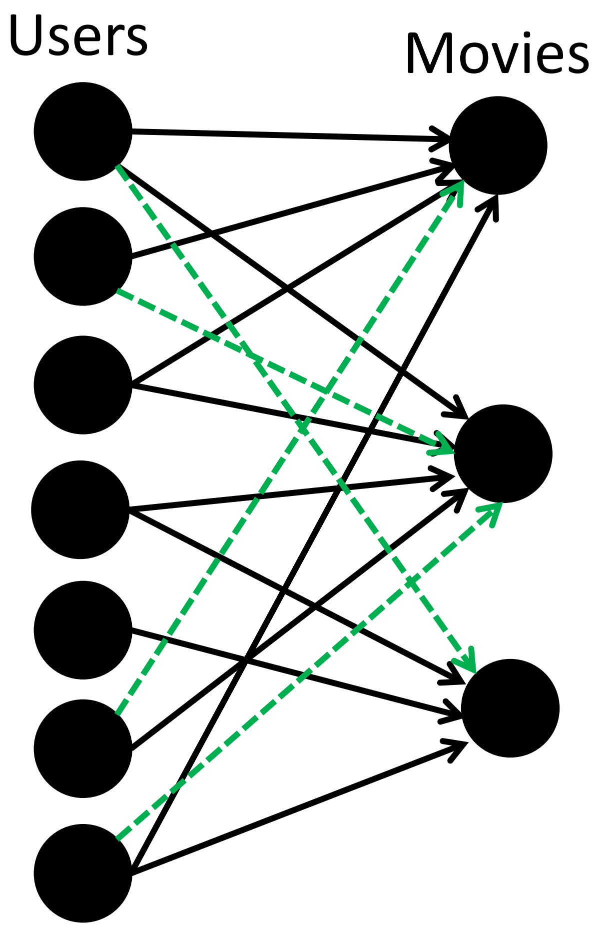
# Differentially-Oblivious Graph Parallel Computation

- **OblivGraph** [Mazloom, Gordon CCS'18]

- Noisy node degree by adding dummy edges
- No. of dummy edges determined by *DP parameters*
- Use Oblivious ~~Sort~~ Shuffle to hide the *edge structure*

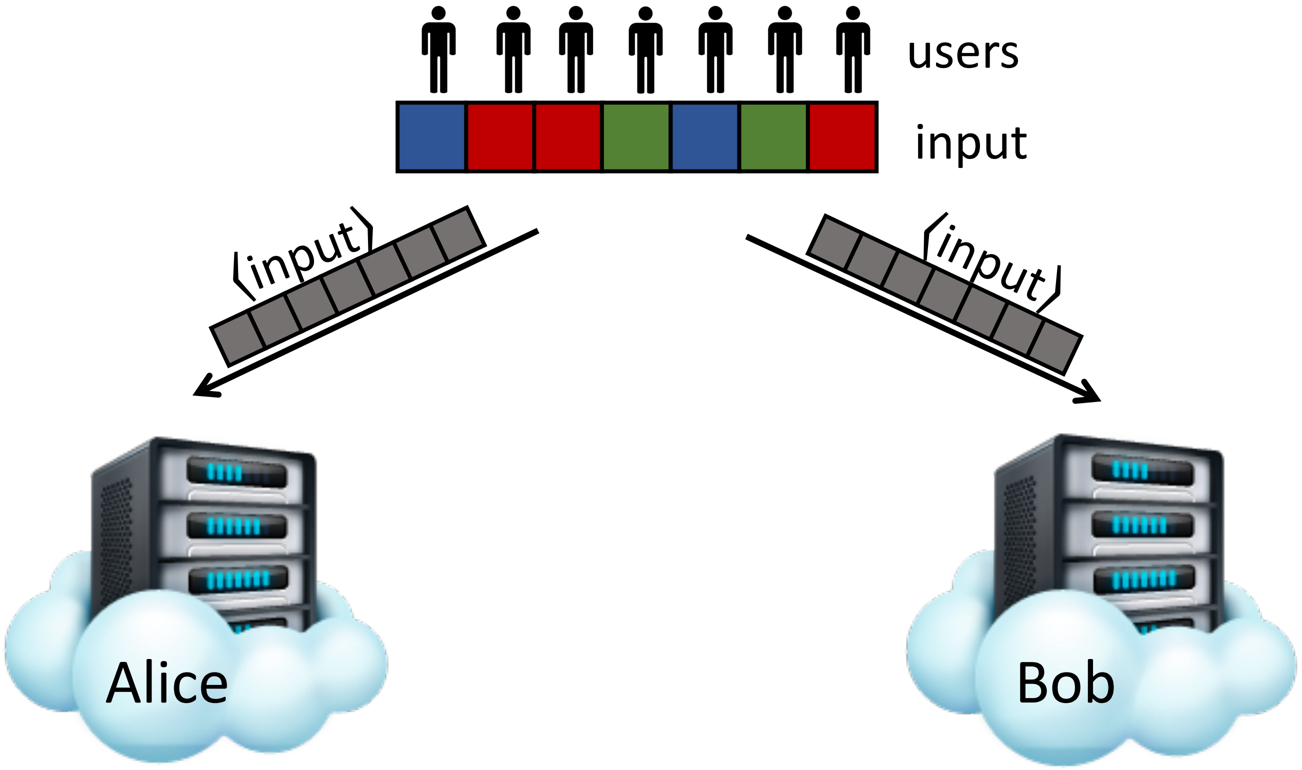
Complexity:

$O((|E'| + \alpha|V|)\ \mathbf{log}(|E'| + \alpha|V|)),\quad \alpha\colon O\left(\frac{\log\delta - \log m}{\varepsilon}\right)$

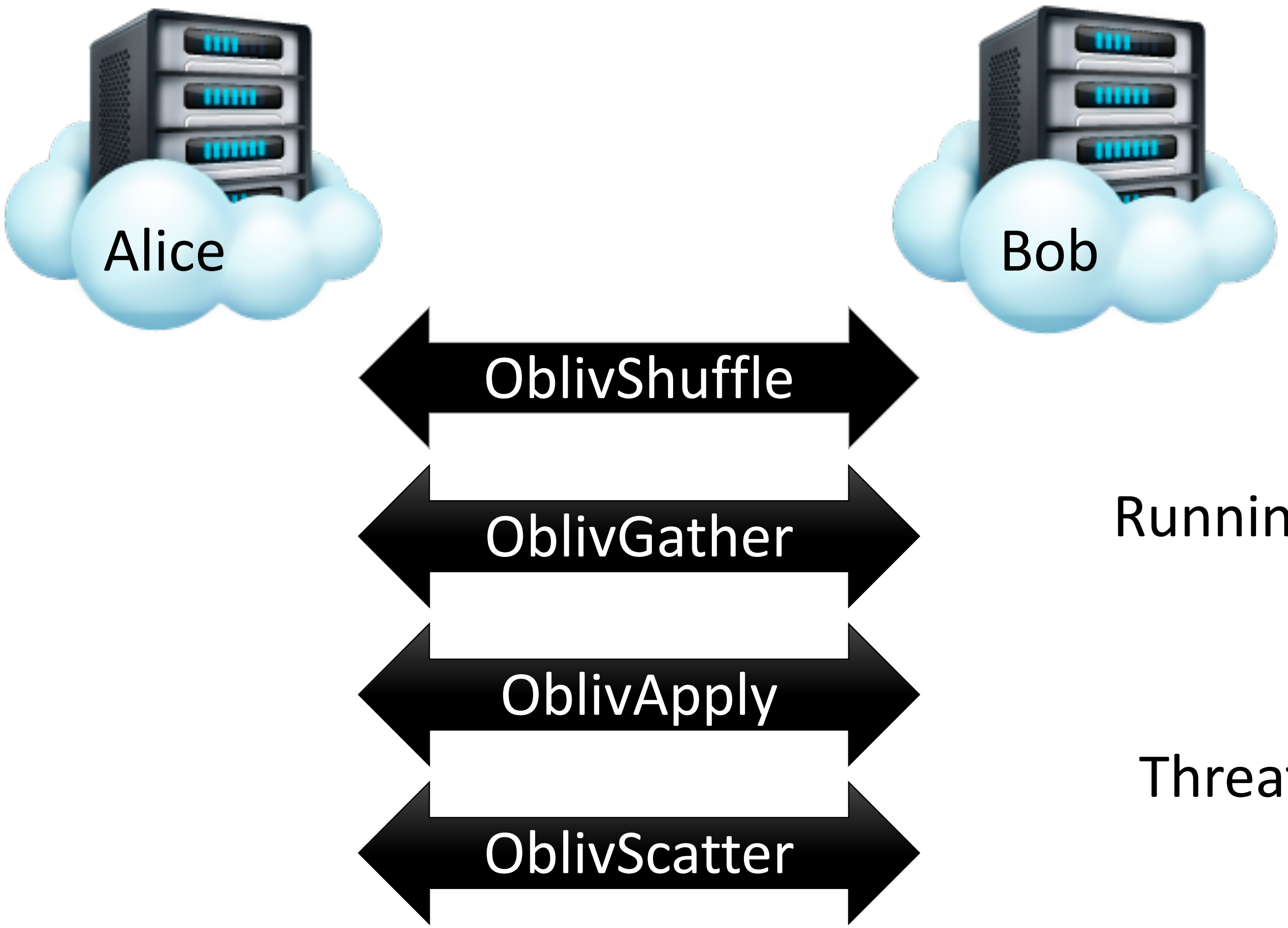



V vertices, E' all edges

# OblivGraph [Mazloom, Gordon CCS'18]



# OblivGraph [Mazloom, Gordon CCS'18]




Running time:

6K users, 4K movies, 1 M Ratings => **2 Hrs**

Threat model: Honest-But-Curious Adversary

# Current Question

Can we make these *differentially private secure computation* algorithms even faster?

# Can we do better?

- ❑ Low communication MPC [Gordon et al. Asiacrypt'18]
- ❑ Differentially Private Leakage in Secure Computation [Mazloom, Gordon CCS'18]
- ❑ Graph Parallel Computation

=> Constructing an MPC protocol that can

Running time:

6K users, 4K movies, 1 M Ratings => **25 Sec**

MF on 20 million inputs < 6 mins (MovieLens dataset)

Histograms on 300 million inputs in only 4 mins (Counting users in each zip code)

# Key playing factors

- ✓ **Using 4 computation servers instead of 2**
- ✓ **Linear Oblivious Shuffle instead of Quasi-Linear OblivShuffle**
- ✓ **Fixed-Point Arithmetic Computation instead of Boolean Circuit**
- ✓ **Secure against one malicious adversary**

Merging these construction ==> Opportunities and Challenges

# Challenge 1

- The party that access the data should **<u>NOT</u>** learn the shuffling pattern

# Solution

✓ Partition the tasks between 4 parties:

Group 1: **Shuffle** the data

Group 2: **Access** the data

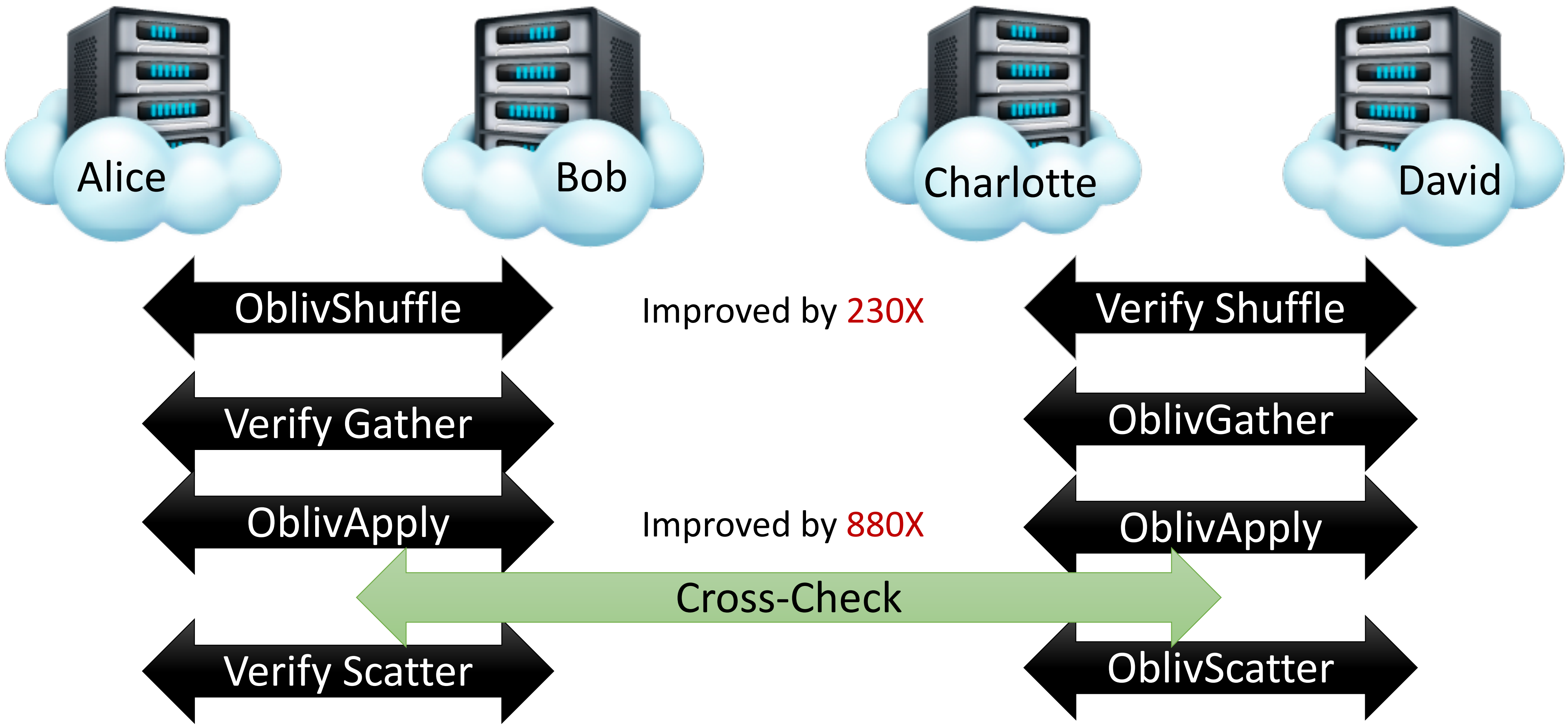
# Challenge 2

- ○ Secure against active adversary

# Solution

- ✓ Verifying the result of each operation to detect cheating behavior

# Malicious-secure 4-party Secure Parallel Computation



# Challenge 3

- ○ Fixed-Point Arithmetic Computation

# Solution

- ✓ Truncation and handling the rounding error

# Cross–Check Verification after Apply phase inspired by [Gordon et al. 2018]

Alice & Bob compute on some masked values and truncate

Charlotte & David compute on some masked values and truncate

If their results don't match?

Abort

The verification may **Fail** if data is a decimal value, and it's NOT because of malicious behavior!

# Implementation Results

Implemented in C++, run experiments on AWS

Multiple benchmark algorithms, including Matrix Factorization and Histogram

4 computation servers, 32 cores each, 10 Gbps network

| Edges | Users | Items | $\varepsilon$ | $\delta$ |
|---|---|---|---|---|
| 1M | 6K | 4K | 0.3 | $2^{-40}$ |
| 10M | 72K | 10K | 0.3 | $2^{-40}$ |
| 20M | 138K | 27K | 0.3 | $2^{-40}$ |
| 300M | 300M | 42K | 0.3 | $2^{-40}$ |

Input size and privacy parameters
for different experiments

# Run Time on National-Scale Histogram Problem

| Processors / Edges | 1M | 10M | 20M | 300M |
|---|---|---|---|---|
| 1 | 13.8 | 85.0 | 207.7 | 2149.4 |
| 2 | 7.5 | 46.5 | 98.1 | 1136.5 |
| 4 | 4.3 | 28.0 | 57.78 | 643.2 |
| 8 | 2.7 | 16.2 | 34.39 | 382.5 |
| 16 | 1.8 | 11.2 | 23.3 | 279.2 |
| 32 | 1.5 | 10.1 | 21.67 | 250.4 |

Run time (s) for computing Histogram problem on different input sizes (LAN)
Counting people in each zip code

# Run Time Large-Scale MF Problem

| Processors / Edges | 1M | 10M | 20M |
|---|---|---|---|
| 1 | 258.3 | 1639.7 | 3401.8 |
| 2 | 132.9 | 834.7 | 1913.7 |
| 4 | 80.4 | 455.57 | 1055.95 |
| 8 | 44.6 | 292.2 | 613.1 |
| 16 | 28.2 | 190.6 | 423.7 |
| 32 | 25.1 | 163.4 | 357.2 |

Run time (s) for computing Matrix Factorization problem on real-world dataset, *MovieLens* on different input sizes for Movie Recommendation

# Run Time Comparison with previous works

| | **GraphSC** | **OblivGraph** | **This work** |
|---|---|---|---|
| Time | $13hrs$ | $2hrs$ | **25s** |

Run time comparison on this work vs. OblivGraph vs. GraphSC. Single iteration of Matrix Factorization on real- world dataset, MovieLens with 6K users ranked 4K movies with 1M ratings

# Summarize

Goal:

Learning on large-scale data with security and privacy

- Secure MPC for Privacy Preserving Machine Learning
- Secure against one malicious corruption
- Leverage Differential Privacy to improve efficiency

# Secure Parallel Computation on National-Scale Volumes of Data

Sahar Mazloom, Phi Hung Le, Samuel Ranellucci, S. Dov Gordon

sseyedma@gmu.edu

Code is publicly available!

# Thanks!

Q&A