Privacy and Security in Distributed Data Markets

Daniel Alabi, Sainyam Galhotra, Shagufta Mehnaz, Zeyu Song, Eugene Wu

SIGMOD 2025 Tutorial

Part 3: Privacy-Preserving Technologies and Security Tools

The Spectrum of Data Marketplace Architectures

Governance/Storage Categories	Centralized Data Storage (Data is Pooled)	Distributed Data Storage (Data Stays Sovereign)
Centralized Governance (Single, Trusted Arbiter)	The Traditional Hub ● Classic data warehouse model ● High trust in one operator required	The Federated Orchestrator ● Data is federated, not pooled ● A central company still manages rules & accesses
Decentralized Governance (Automated, "Smart" Arbiter)	The Governed Pool ● Data is pooled, but governed by code/community ● A niche but emerging approach	The Sovereign Exchange ● Data Sovereignty by Design ● Transaction Integrity via Arbiter

Trading Insights via Gradients in Distributed Marketplace

The Core Idea:

Instead of trading the data itself, participants trade the "insight" the data provides to a machine learning model.

How is this insight captured?

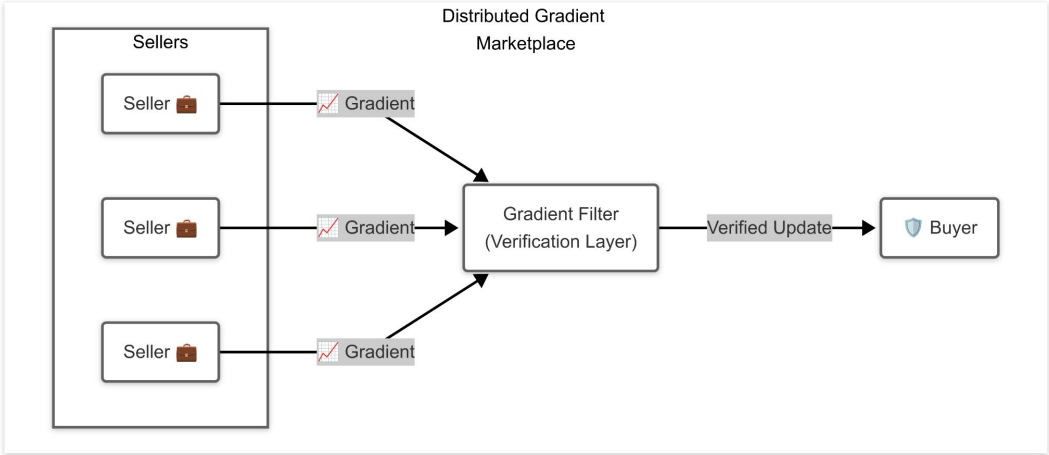
Through a Gradient.

The Implication for Valuation:

In this model, the transaction and valuation are no longer about the raw data. They are now fundamentally tied to the quality and utility of the gradient itself.

How a Gradient Marketplace Works

- A Buyer wants to train or improve their ML model.
- Multiple Sellers use their private data to compute gradients for the buyer's model.
- The Buyer purchases these gradients and uses them to update their model.



The Problem – A Wall Between Buyers and Data

THE DATA BUYER

Needs to accurately assess data quality and value before committing resources.

KEY BARRIERS

🛡️ **Privacy & Security**: Prevents the exposure of sensitive user data and Personally Identifiable Information.

🔒 **Data Ownership & IP**: Protects the data as the seller's core asset and valuable intellectual property.

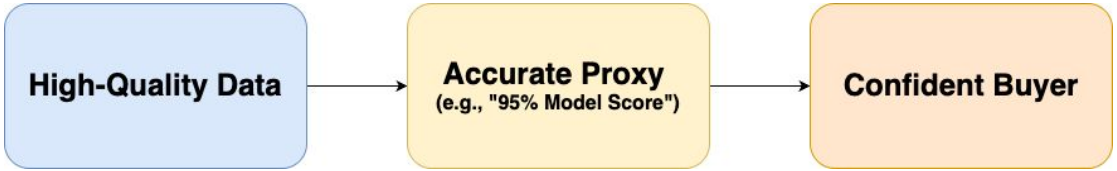
☁️ **Scale & Efficiency**: Makes the full transfer and inspection of massive datasets logistically impractical.

THE RAW DATASET

The source of truth remains unseen, its quality unverified.

The Flawed Solution: Valuing by a (Gameable) Proxy

The Theory: A Proxy Is an Honest Signal of Quality



The Reality: A Proxy Can Be Manipulated

The Central Vulnerability:

A proxy can be manipulated independently of the data's actual quality. This makes the entire valuation process gameable.

Valuation in Distributed Setups

Frameworks like DAVED* solve this by performing valuation on embeddings (i.e., "fingerprints") instead of raw data.

The Goal: This allows for good data selection in a distributed setup, preserving privacy while assessing quality.

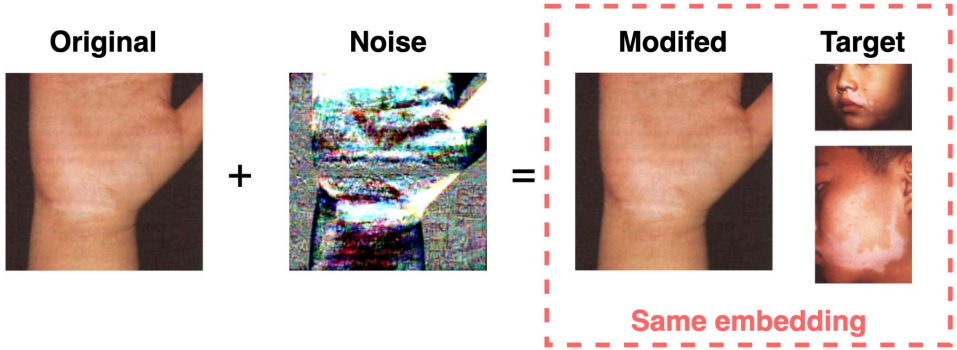
*Lu et al., "DAVED: Data Acquisition via Experimental Design for Data Markets," NeurIPS 2024.

How to Fool an Embedding-Based Valuation

- A buyer wants data with embeddings similar to a *Target Image*.
- The seller adds calculated, imperceptible noise to their own *Irrelevant Image*.
- The new "noisy" image now has an embedding nearly identical to the *Target Image*.
- The Result: The valuation is fooled. The buyer's system approves the purchase, but they receive a dataset of useless, manipulated images.

The Vulnerability: Embeddings Can Be Manipulated

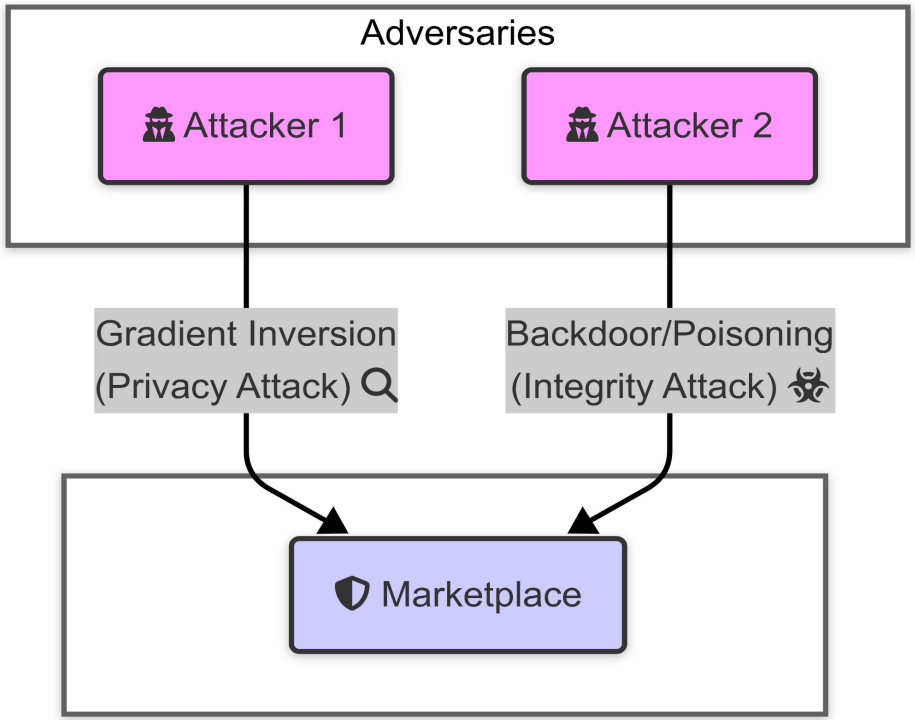
Context: A buyer is searching a dermatology dataset (like Fitzpatrick17K) for high-value images of a specific skin condition to train their ML model.

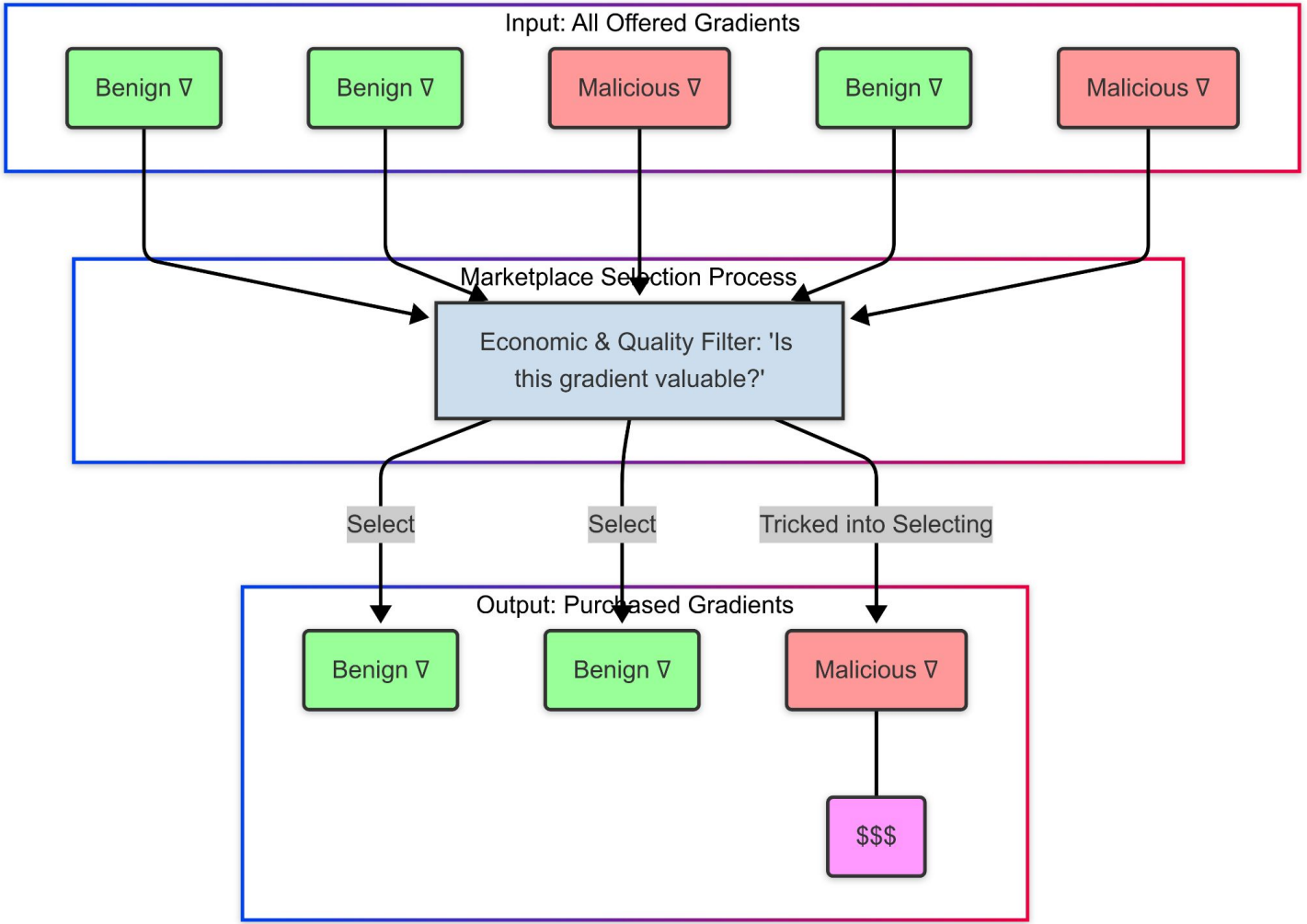


- Original: A irrelevant Original image.
- Noise: A layer of calculated, human-imperceptible Noise is added.
- Modified: The resulting Modified image looks identical to our eyes, but its "fingerprint" —its embedding—is now the same as the high-value Target image.

Security and Privacy Challenges: A Marketplace Under Siege

Threat Category	The Adversary's Goal
Privacy Attacks	To reconstruct sensitive, private training data from the shared gradient.
Integrity Attacks	To corrupt the model's performance or install a hidden, malicious trigger.




Input: All Offered Gradients
Benign ∇
Benign ∇
Malicious ∇
Benign ∇
Malicious ∇
Marketplace Selection Process
Economic & Quality Filter: 'Is this gradient valuable?'
Select
Select
Tricked into Selecting
Output: Purchased Gradients
Benign ∇
Benign ∇
Malicious ∇
$$$

Gradient Valuation

Instead of trading raw data or its indirect proxies, a Gradient Marketplace creates a more secure system by trading the direct output of machine learning: the model gradients themselves.

How It Works: Frameworks like martFL* provide the architecture for this model:

Direct Inspection: A selection filter is used to directly inspect the quality and utility of each incoming model update (gradient). It rejects contributions that are malicious or low-quality.

"What You See Is What You Get": The buyer receives the exact same gradient that was just evaluated by the filter.

*Li et al., "martFL: Enabling Utility-Driven Data Marketplace with a Robust and Verifiable Federated Learning Architecture," ACM Computer and Communications Security Conference 2023.

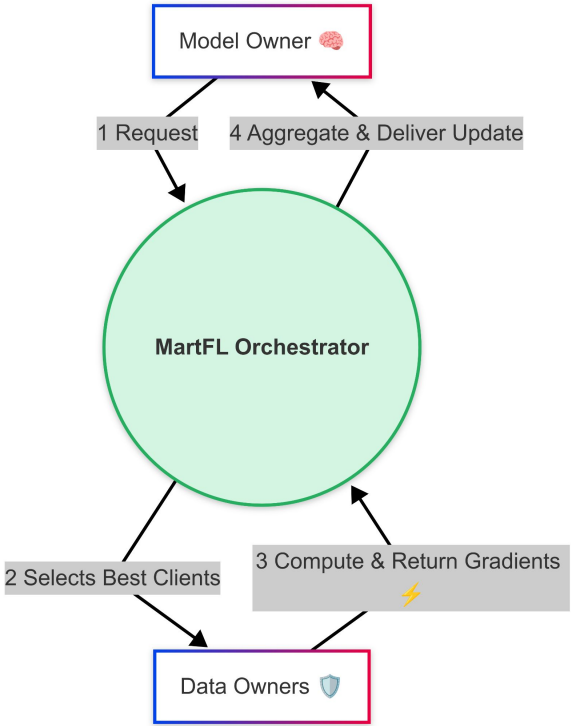
The MartFL Life Cycle

Request: A model owner submits a training task to the marketplace.

Select: The MartFL Orchestrator uses its two-phase filter to select the most valuable data owners.

Train: The selected owners compute gradients on their private data.

Improve: MartFL aggregates the gradients, ensures fair payment, and delivers a single, powerful update to the model owner.



The New Threat: Malicious Gradient Attacks

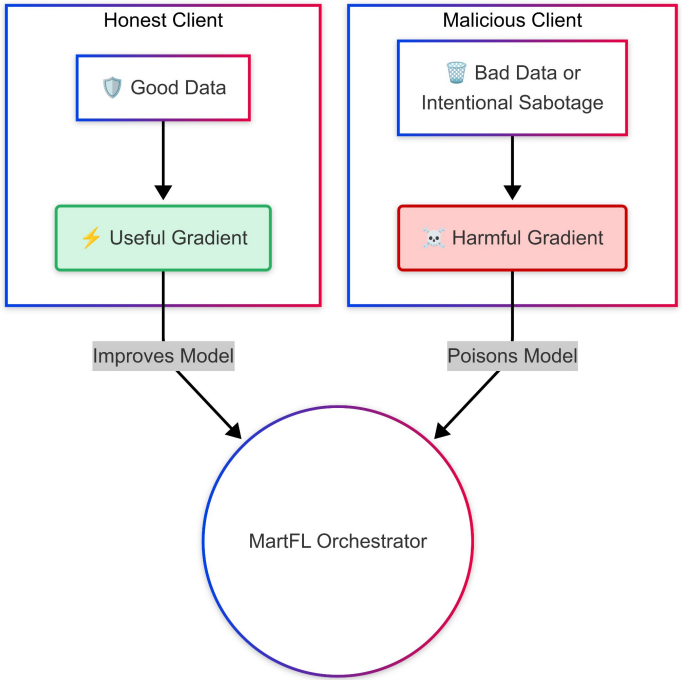
In a gradient marketplace, the threat shifts from faking value to actively sabotaging the training process.

A Malicious Client's Goal:

- Get paid for contributing nothing of value.
- Poison the global model, reducing its accuracy for their own benefit.

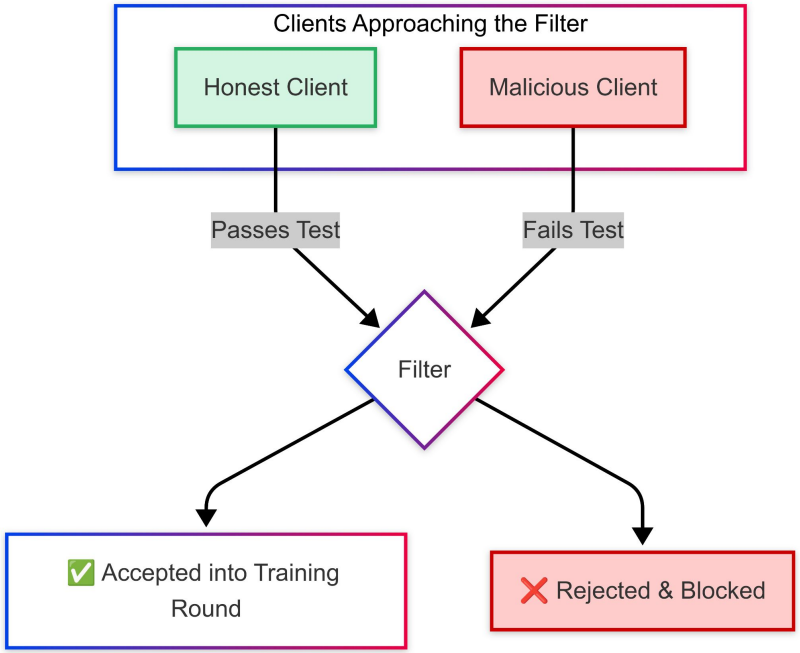
The Method:

Submit useless or deliberately harmful gradients instead of honest ones.



How MartFL Filters Malicious Gradients

- **Create a Baseline**: A "trusted baseline" is established using the buyer's clean reference gradient combined with past contributions from high-quality sellers.
- **Measure Similarity**: The system calculates the cosine similarity between each new gradient and the trusted baseline.
- **Filter Outliers**: Any gradient with low similarity is flagged as an outlier and rejected, filtering out potentially malicious or useless updates.

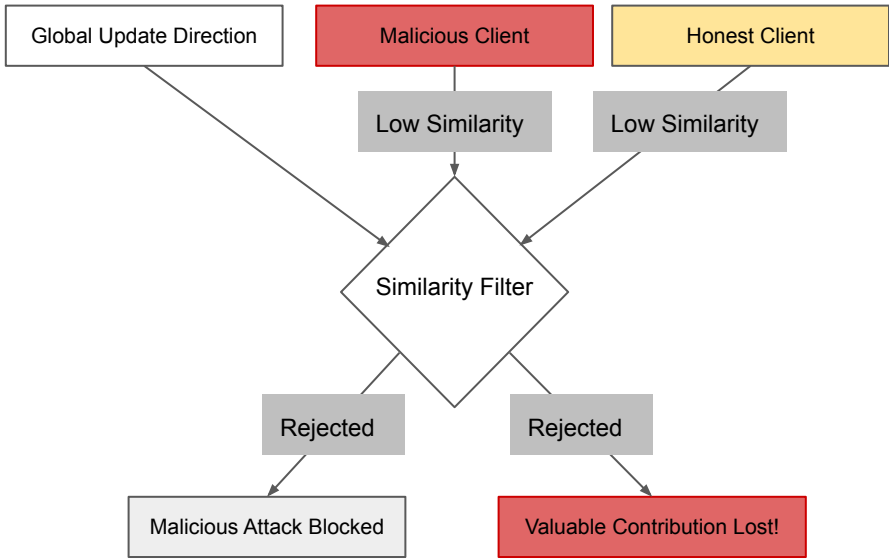


The Potential Flaw: Can Similarity Be Fooled?

This leads to two key research questions for our analysis:

Robustness: How well does similarity filtering actually detect various malicious attacks?

Fairness: Does this filtering mechanism unfairly penalize honest clients who hold valuable, non-mainstream (outlier) data?



The Blind Spot in Gradient Marketplace Evaluation

A system can be **technically** perfect and secure, but **economically** broken.

To build a truly successful marketplace, we must evaluate the entire ecosystem. Our framework integrates three crucial marketplace-centric metrics.

Model Robustness

The Question: Is the final trained model accurate, reliable, and secure against attacks?

Economic Viability

The Question: Is the marketplace economically efficient? Does the performance gain justify the cost?

Marketplace Stability

The Question: Is the system fair and stable for its participants?

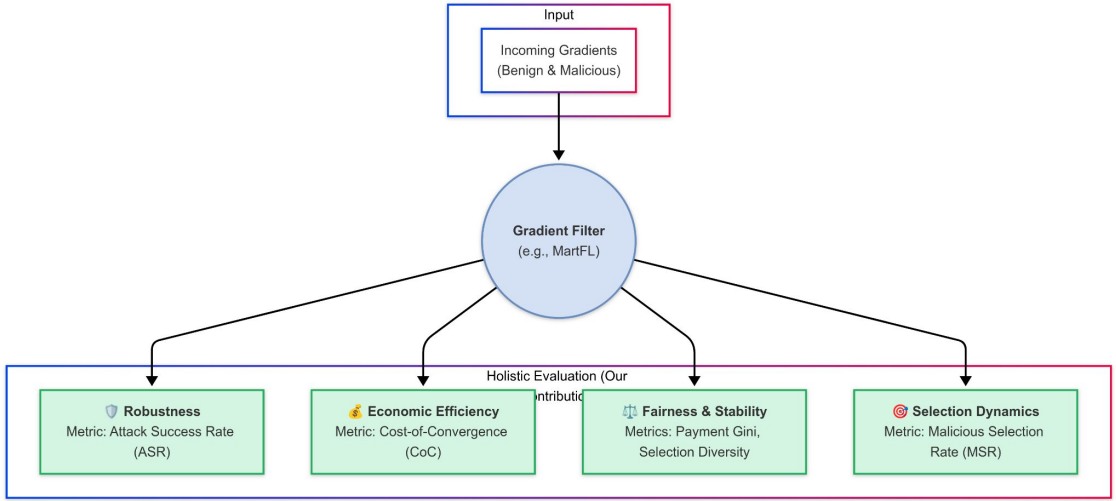
Key Evaluation Dimensions

Robustness: Does the filter stop the attack?

Economic Efficiency: What is the true cost for the buyer to achieve their goal?

Fairness & Stability: Are honest sellers treated fairly, or are they penalized?

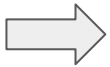
Selection Dynamics: Who is the filter actually selecting, and how often is it fooled?



Case Study: Can the Marketplace Survive a Backdoor Attack?

The Attacker's Playbook

Step 1: Poison the Source Data
The attacker adds a trigger (e.g., a white square) to a "cat" image and maliciously labels it as a "dog."

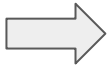


Step 2: Submit the Malicious Gradient
The attacker offers the harmful gradient, learned from this poisoned data, for sale in the marketplace.

Our Analysis

Analysis of Step 1: Security Effectiveness

Our framework asks:
✅ Can the selection filter detect the malicious gradient's signature?
✅ Can it prevent the poison from compromising the global model?



Analysis of Step 2: Economic & Fairness Impact

Our framework asks:
⚖️ What is the collateral damage? Are benign, honest sellers unfairly penalized or rejected by stricter filtering triggered by the attack?

Two Attack Scenarios

Attack 1: The Standard Backdoor

The Strategy: Brute Force. The adversary submits a standard poisoned gradient and simply *hopes* it bypasses the marketplace filter.

Attack 2: The Sybil "Mimicry" Backdoor

The Strategy: Deception & Camouflage. This is a more intelligent, two-step attack:

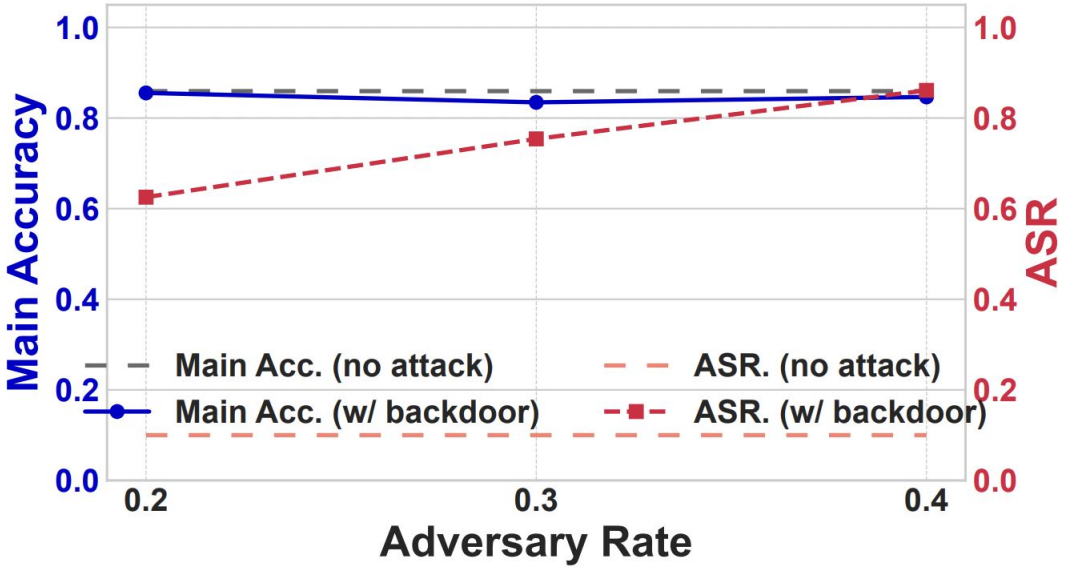
a) Learn What Passes: The adversary identifies the characteristics of *benign* gradients that are successfully selected.

b) Blend the Attack: They **combine** their malicious backdoor gradient with this benign "camouflage" to create a new gradient that looks trustworthy.

Result - Final model performance

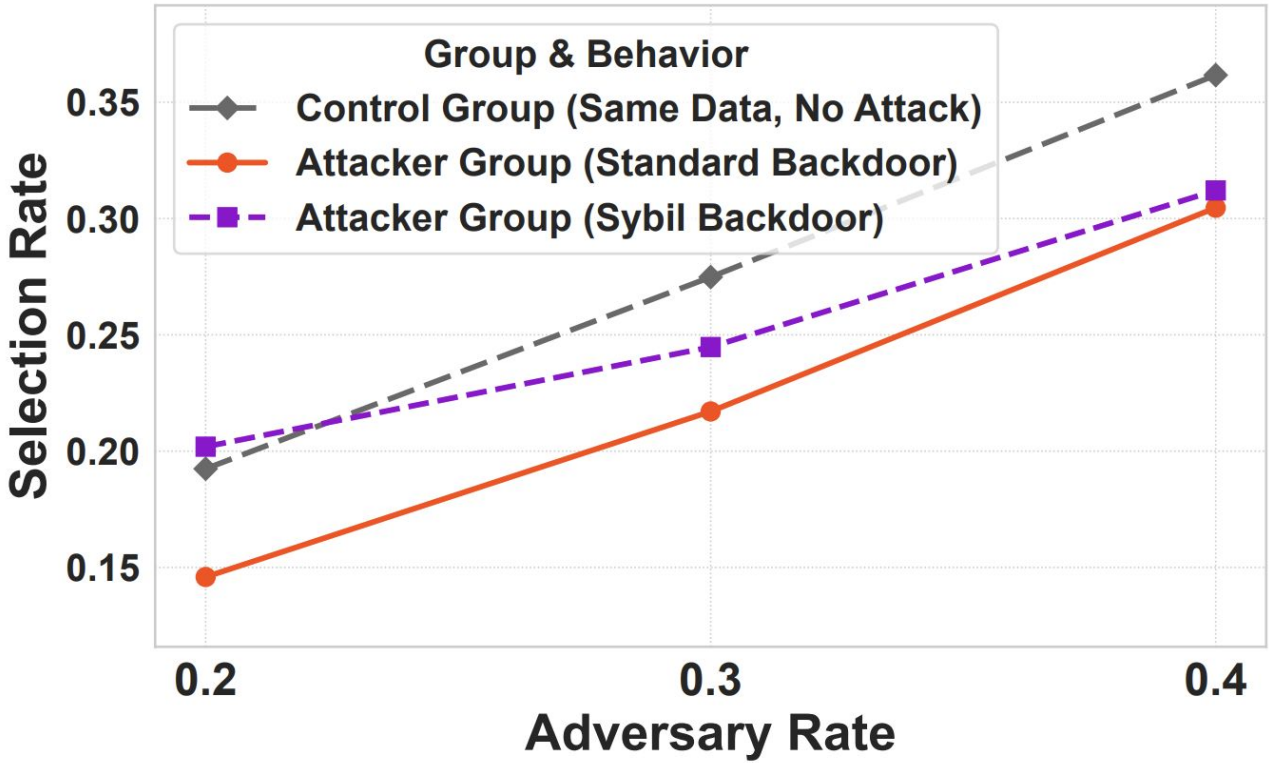
The Illusion (Blue Line): The model's performance on its main task remains high and stable, suggesting the system is healthy.

The Reality (Red Line): Simultaneously, the Attack Success Rate skyrockets, proving the model is being successfully poisoned with a hidden backdoor.



Mechanism of Failure: Why the Filter Was Fooled

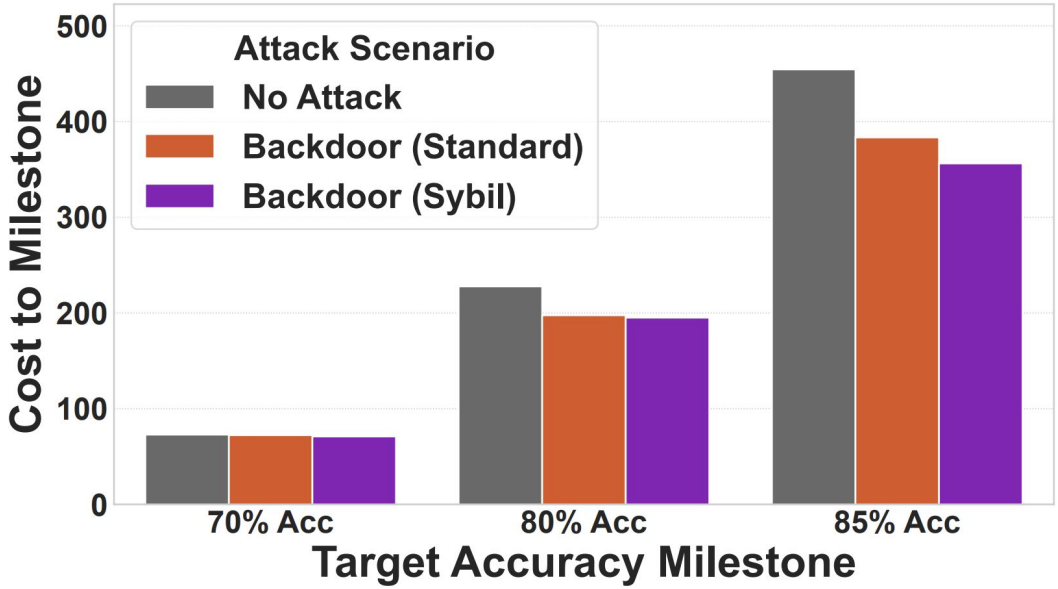
Despite the filtering mechanism, a sufficient volume of malicious updates evaded detection, enabling the backdoor attack to succeed.



"Deceptive Efficiency" of an Attacked Market

- The Sybil-attacked market reaches the target accuracy with 23% less cost (fewer gradients purchased).

- From a purely economic standpoint, the attacked market looks more efficient. A buyer optimizing solely for cost would inadvertently prefer the compromised environment. This makes the attack even harder to detect through economic signals.

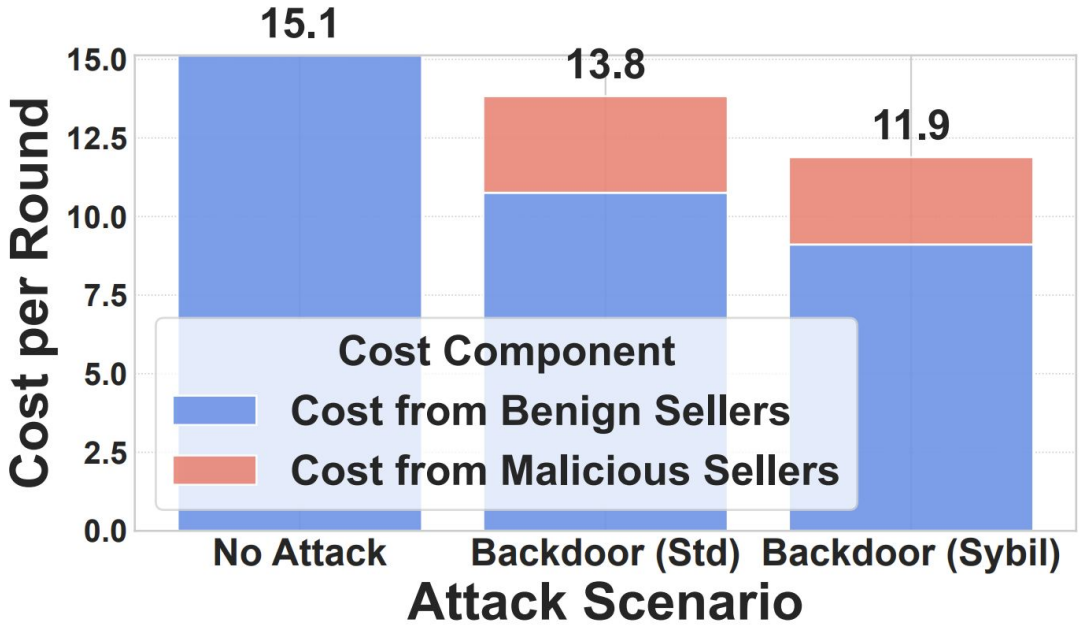


Economic Fallout: Who Really Pays the Price?

No Attack: Honest sellers earn 100% of the revenue.

Sybil Attack: Honest seller revenue plummets by 40%. Attackers successfully extract nearly a quarter of all payments.

The market isn't more efficient. Attackers are simply crowding out and defunding honest contributors, creating an unsustainable economy.



Data Discovery Dilemma: Diversity vs. Security

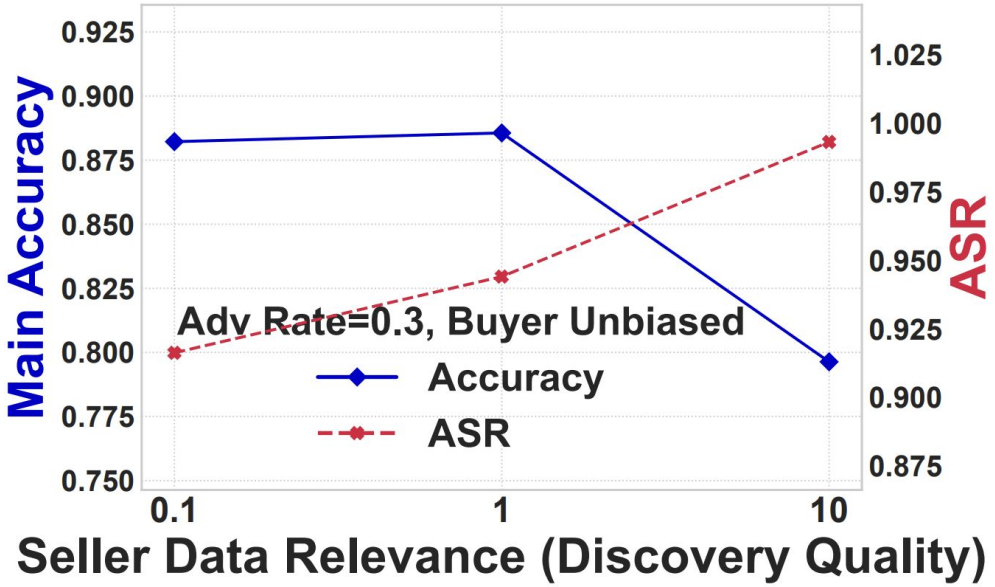
1. The Marketplace Dilemma

A realistic marketplace needs data diversity. However, this creates a fundamental conflict for similarity-based security filters.

2. The Mechanism of Failure

With Homogeneous Data: The filter works. Malicious gradients are easy-to-spot outliers. ✅

With Heterogeneous Data: The filter collapses. It cannot distinguish between "benign diversity" and "malicious intent". ❌



Key Takeaways

Finding	Implication
1. The Attack Surface Has Shifted.	The primary vulnerability is now the marketplace's **economic and selection mechanisms.**
2. Standard Metrics Are Deceptive.	High accuracy and low cost can **mask catastrophic security failures** and unfair outcomes.
3. Similarity-Based Defenses Are Brittle.	They are fundamentally vulnerable to mimicry and **fail in diverse, realistic environments.**

Saibot: Differentially Private Task-based Search

(back to centralized search)
Huang et al. VLDB 23

Task-based Search: Basic Algorithm

```
D = initial training dataset
for next augmentation α
    if eval(apply A to D) is best so far
        Keep α in A
return best A
```

Greedy

Use sketches

But can we enforce differential privacy?

Differential Privacy

Privatization: Differential Privacy(DP) Algorithm

$$\Pr[M(D) \in S] \leq \exp(\epsilon) * \Pr[M(D') \in S] + \delta$$

Informally, an algorithm satisfies DP if no single record can be inferred

- Hides individuals in a dataset by adding noise to results
- Each query consumes part of a dataset's finite budget
- Consumed budget $\propto$ noise added to result

Differential Privacy: Privacy Budget

Privacy budget: (ϵ, δ)

SELECT SUM(Y) FROM D

D

A	Y
a_1	1
a_1	2

Remaining budget:

(ϵ, δ)

Differential Privacy: Privacy Budget

Privacy budget: (ϵ, δ)

SELECT SUM(Y) FROM D

$1 + 2 + noise_{\epsilon, \delta} = 4.2$

D

A	Y
a_1	1
a_1	2

Remaining budget:

(0, 0)

Differential Privacy: Privacy Budget

Privacy budget: (ϵ, δ)

SELECT SUM(Y*Y) FROM D

D

A	Y
a_1	1
a_1	2

Remaining budget:

(0, 0)

Differential Privacy: Privacy Budget

Privacy budget: (ϵ, δ)

SELECT SUM(Y*Y) FROM D

No privacy budget, cannot access

D

A	Y
a_1	1
a_1	2

Remaining budget:

(0, 0)

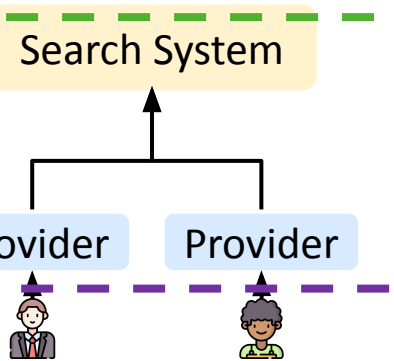
Differential Privacy Mechanisms Available

Uses budget on every model eval

GDP

Too much noise to be useful

LDP



Data Task

ML data augmentation search

- More samples to union with.
- More features to join with.

Example: Predicting churn

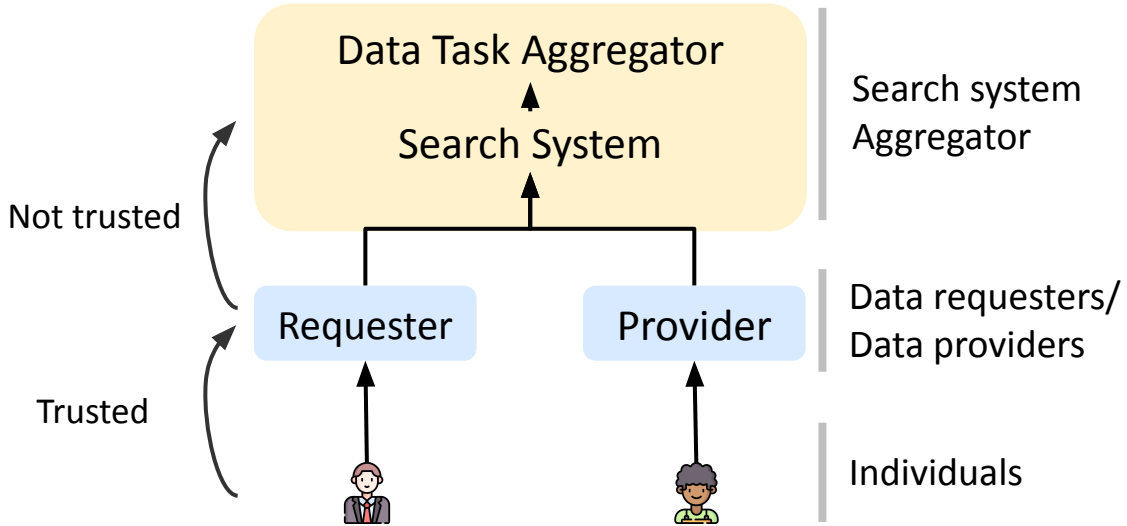
Churned	Customer	Subscription Date	Most Visited	Unemployment Rate
Yes	Alice	Jan 2023	Products	6.5%
No	Bob	May 2023	Support	3.2%
Yes	Charlie	Feb 2023	Support	8.1%
No	David	Jan 2023	Home	6.5%

DP ML Data Augmentation: Input and Output

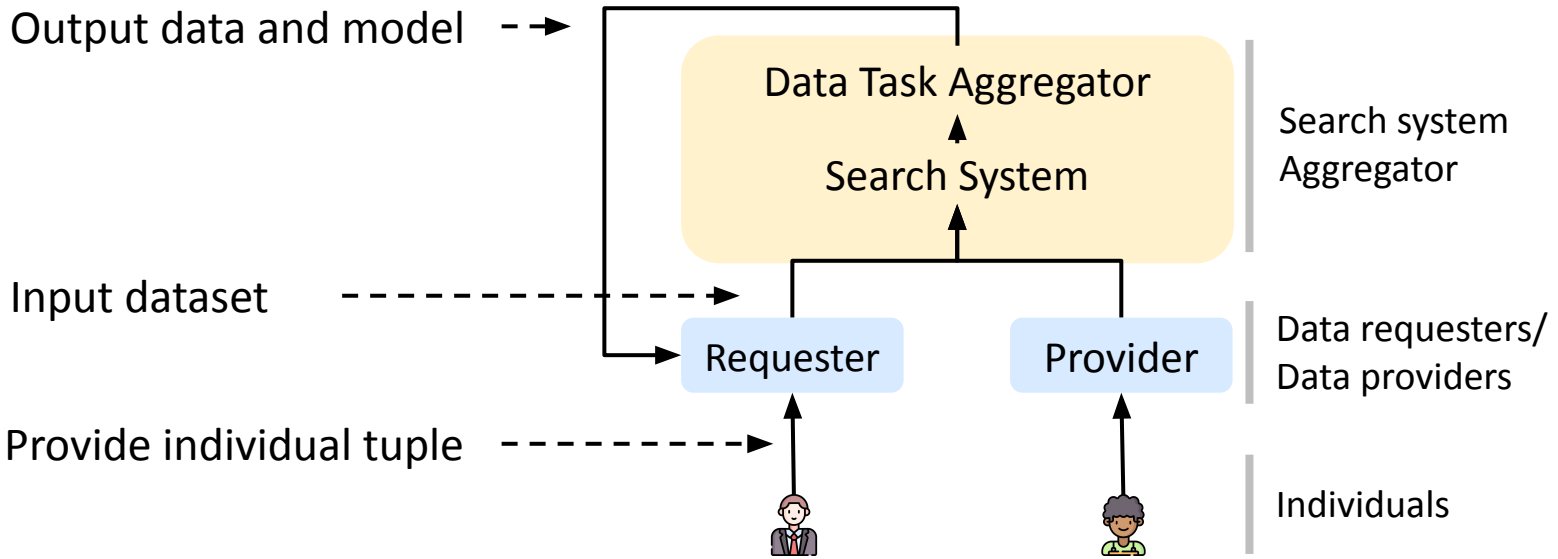
Want to find health data to improve cardiac prediction models

Patients don't trust Google to use health data for ads.

Patients trust their health tracking App, like Fitbit.



DP ML Data Augmentation: Input and Output

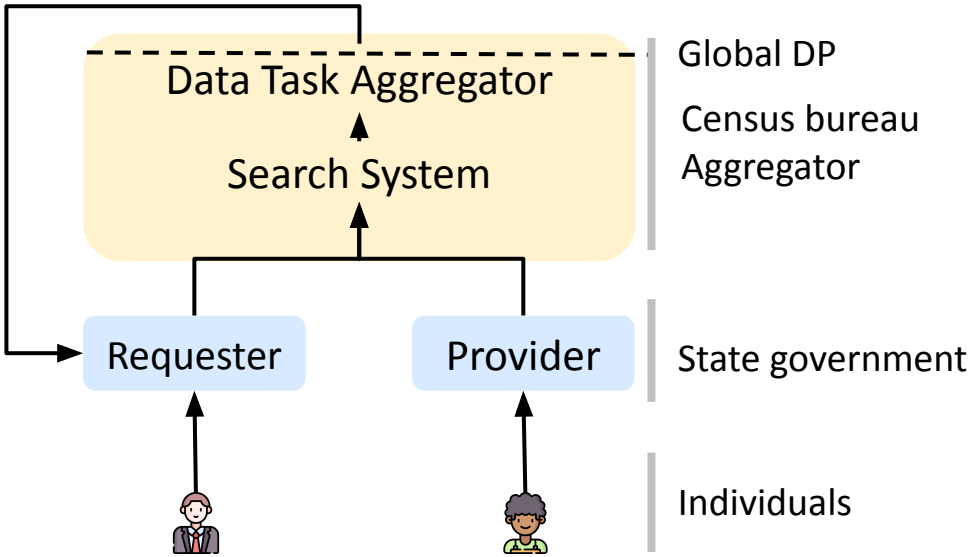


Existing Approach Limitations: Global DP

Global DP mechanisms add noise before releasing the output.

Evaluating each combination drains privacy budget.

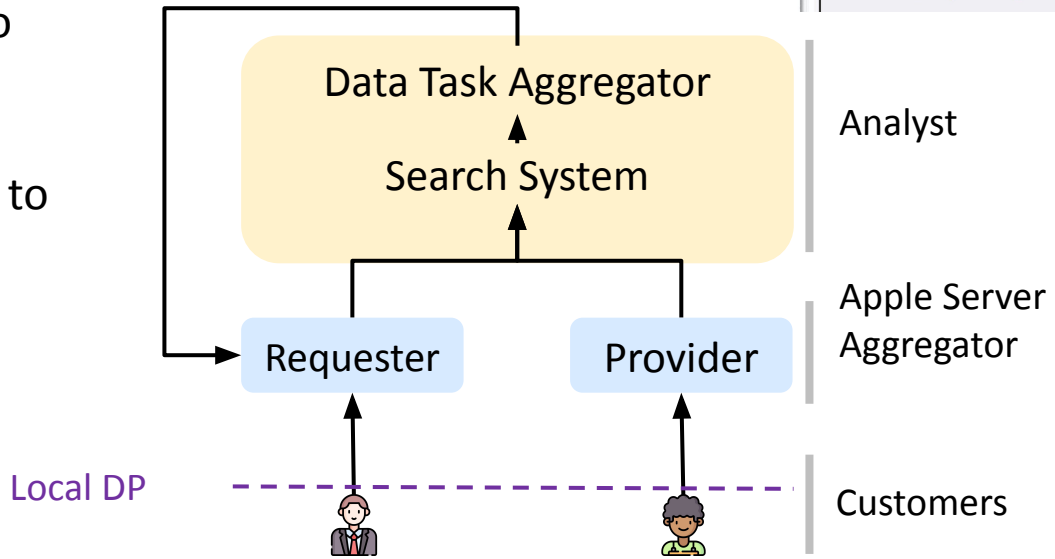
Exponential combinations of join/union-compatible sets.



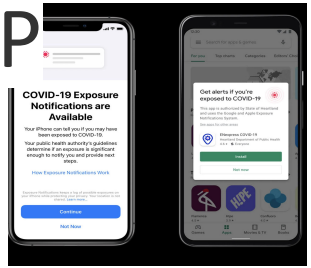
Existing Approach Limitations: Local DP

Local DP mechanisms add noise to each customer's data.

Augmentations too noisy, difficult to distinguish useful ones.

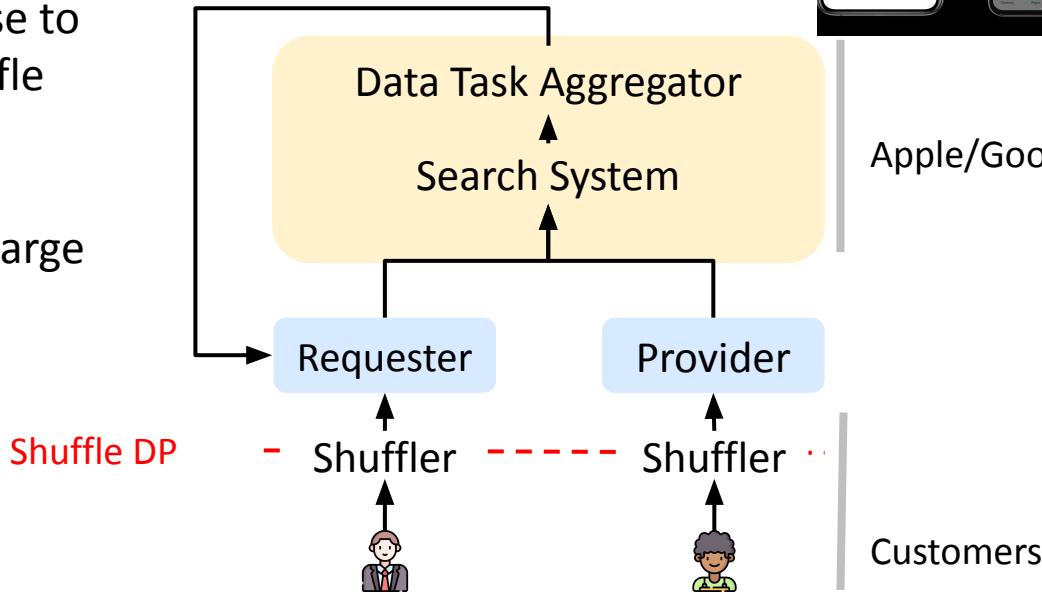


Existing Approach Limitations: Shuffle DP



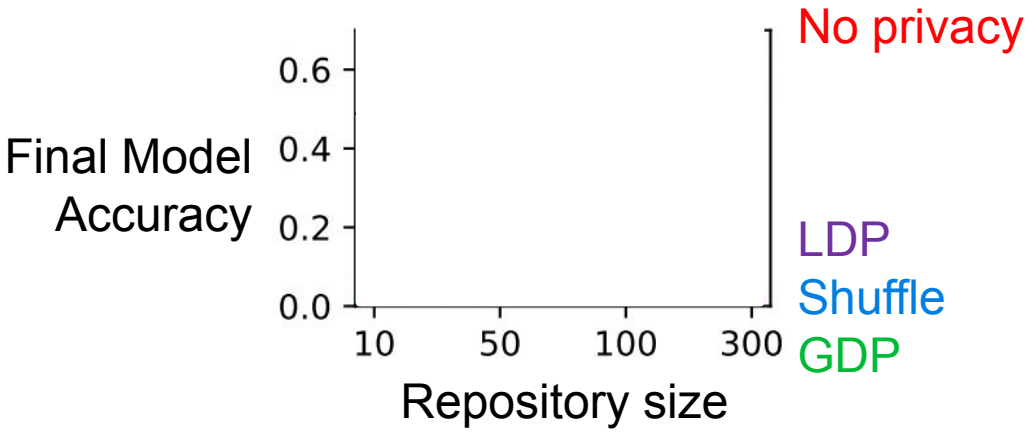
Shuffle DP mechanisms add noise to each customer's data, then shuffle to enhance privacy.

Only enhance privacy levels for large datasets.

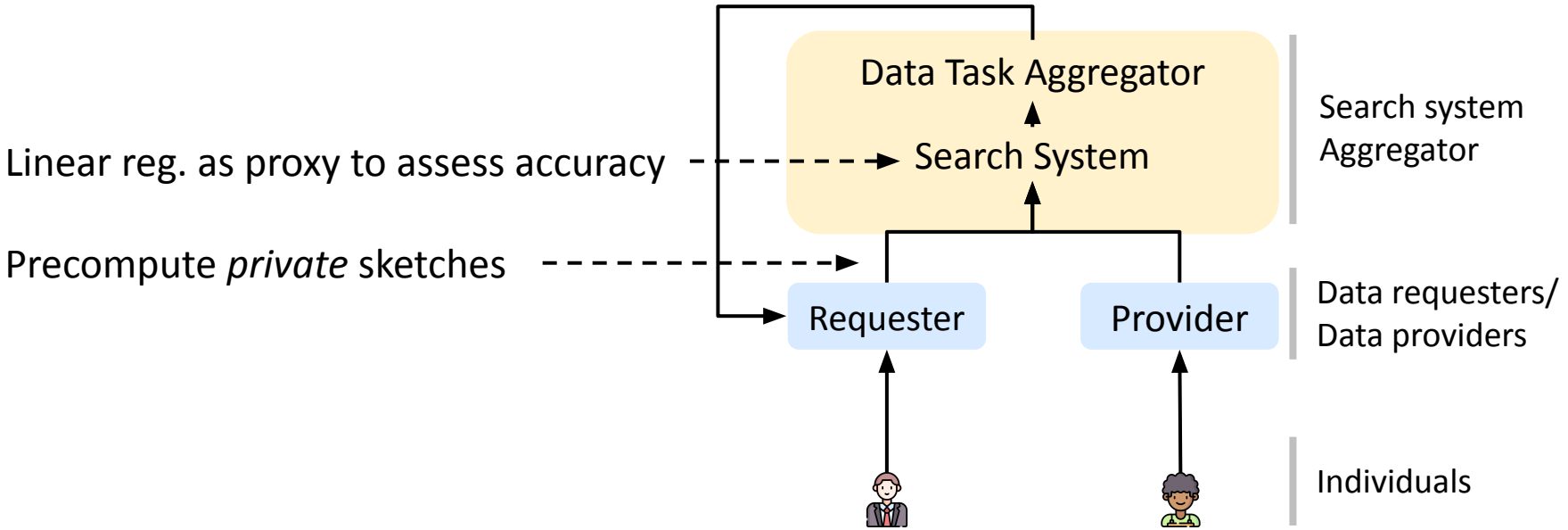


Prior Mechanisms Don’t Scale

To repository size & number of requests
Vary between 10 - 329 NYC Open Datasets in Repo
Query: Grad table to predict 2016-17 graduation outcomes.

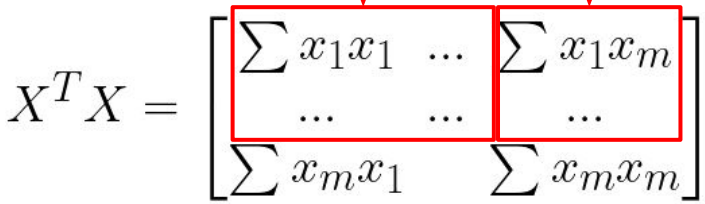


Sketch-based Approach



Sketch-based Search

Linear regression has closed form solution $\widehat{\beta} = (X^T X)^{-1} X^T y$



Sketch-based Search

How to compute sum of pairwise product between features?

D

A	Y
a_1	1
a_1	2

Compute aggregates as sketches

Monomial semi-ring

A	sum(Y^2)	sum(Y)	count
a_1	1^2+2^2	1+2	2

Sum of 0^{th}, 1^{st}, 2^{nd}-order monomials

Sketch-based Search

Linear regression on D⋈R requires computing $\sum 1, \sum B, \sum Y, \sum BY$

D

A	Y
a_1	1
a_1	2

⋈

R

A	B
a_1	1
a_1	2

D ⋈ R

A	Y	B
a_1	1	1
a_1	1	2
a_1	2	1
a_1	2	2

= 9

Sketch-based Search

Linear regression on D⋈R requires computing $\sum 1, \sum B, \sum Y, \sum BY$

D

A	Y
a_1	1
a_1	2

R

A	B
a_1	1
a_1	2

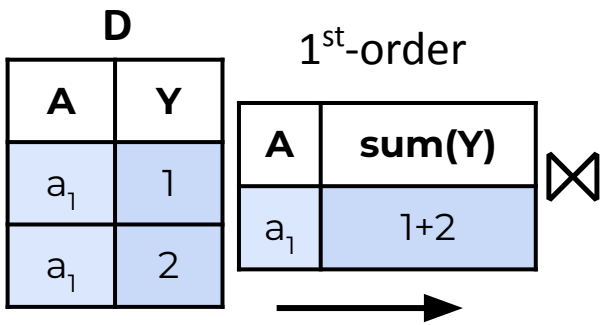
D ⋈ R

A	Y	B
a_1	1	1
a_1	1	2
a_1	2	1
a_1	2	2

= 9

Sketch-based Search

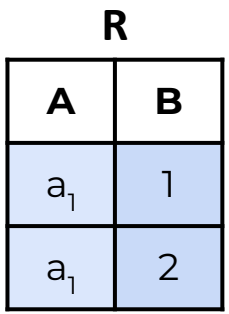
Linear regression on D⋈R requires computing $\sum 1, \sum B, \sum Y, \sum BY$



1st-order

A	sum(Y)
a_1	1+2

⋈



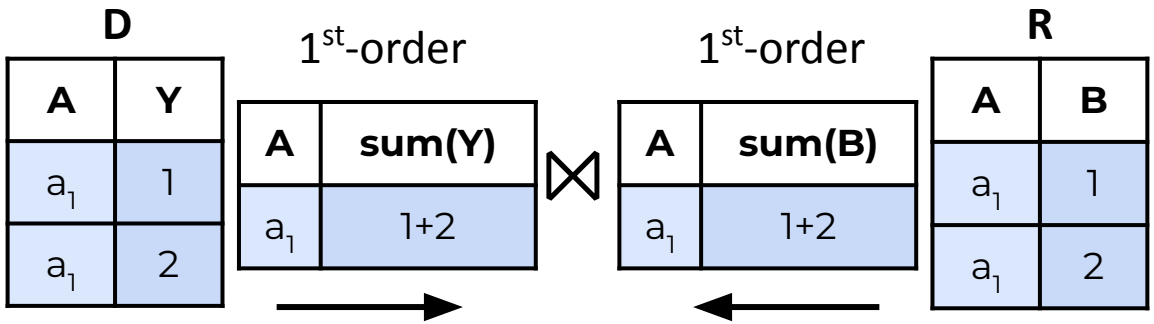
D ⋈ R

A	Y	B
a_1	1	1
a_1	1	2
a_1	2	1
a_1	2	2

= 9

Sketch-based Search

Linear regression on D⋈R requires computing $\sum 1, \sum B, \sum Y, \sum BY$



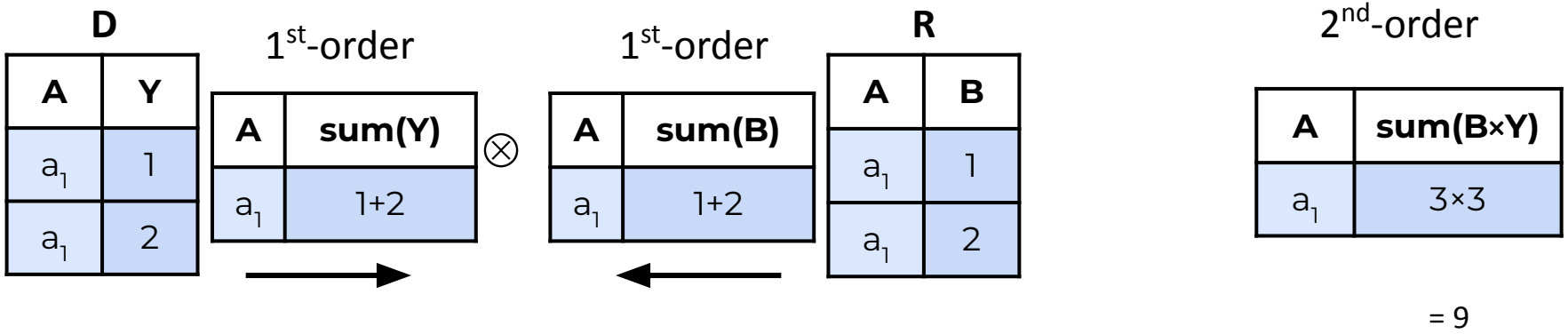
$D \bowtie R$

A	Y	B
a_1	1	1
a_1	1	2
a_1	2	1
a_1	2	2

= 9

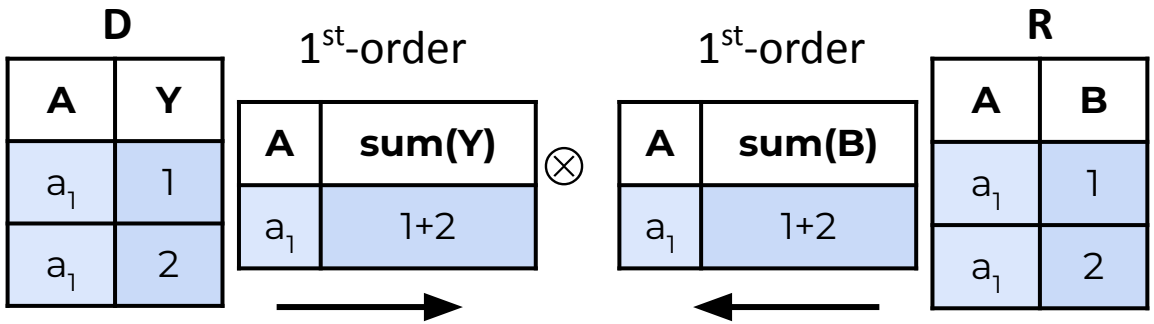
Sketch-based Search

Linear regression on D⋈R requires computing ∑1, ∑B, ∑Y, ∑BY



Sketch-based Search

Linear regression on D⋈R requires computing $\sum 1, \sum B, \sum Y, \sum BY$



D ⋈ R

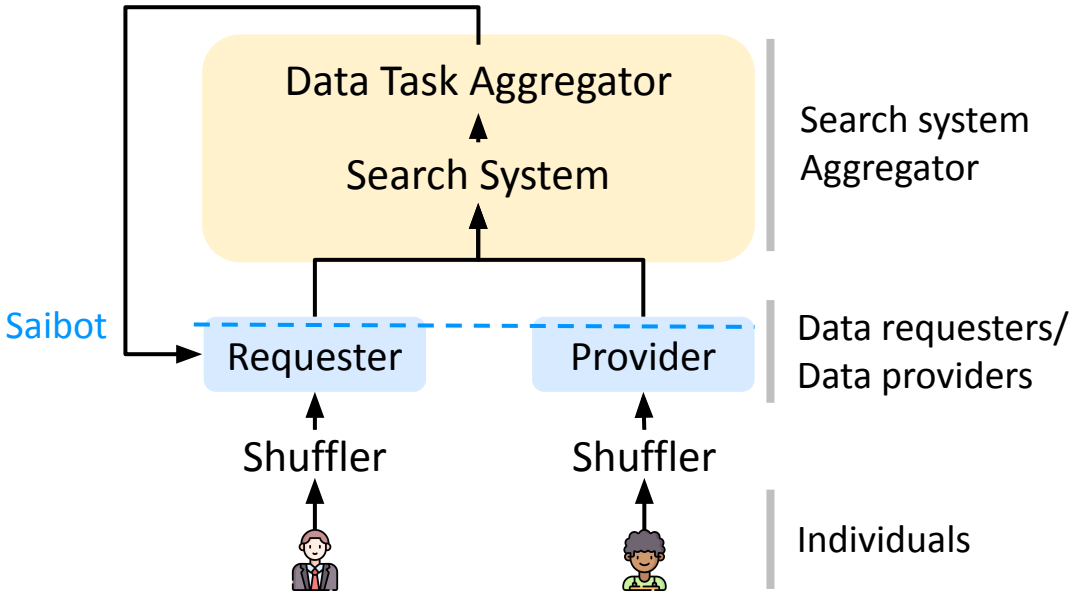
A	Y	B
a_1	1	1
a_1	1	2
a_1	2	1
a_1	2	2

= 9

Saibot: Our Contribution

Factorized Privacy Mechanism

Privatize sketches so downstream search will be privatized.



Intuition: aggregate datasets as much as possible before adding noise to them.

Saibot: Technical Details

- Factorized Privacy Mechanism (FPM).
- Noise allocation optimization.
- Unbiased estimation.
- Proofs

Saibot: Technical Details

- Factorized Privacy Mechanism (FPM).
- Noise allocation optimization.
- Unbiased estimation.
- Proofs

Saibot: Assumptions

The schema and join keys for datasets owned by providers are public

- Oblivious intersection techniques can be applied.

All tuples are L2 bounded by B (for analysis)

- Categorical features numericalized

FPM:Privatize sketches with privacy budget (ϵ, δ)

Use existing DP query engine

D

A	Y
a_1	1

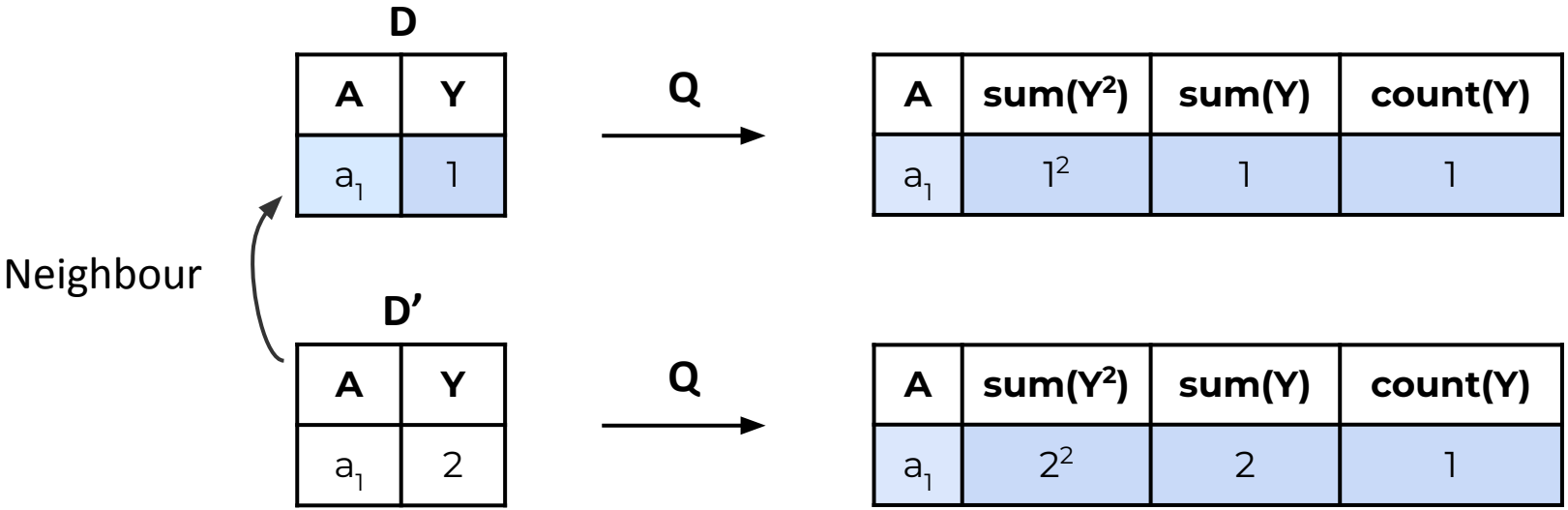
Q →

A	sum(Y^2)	sum(Y)	count(Y)
a_1	1^2	1	1

Q: SELECT SUM(**Y**^{2}), SUM(**Y**), COUNT(**Y**) from **D** GROUP BY **A**

FPM:Privatize sketches with privacy budget (ϵ, δ)

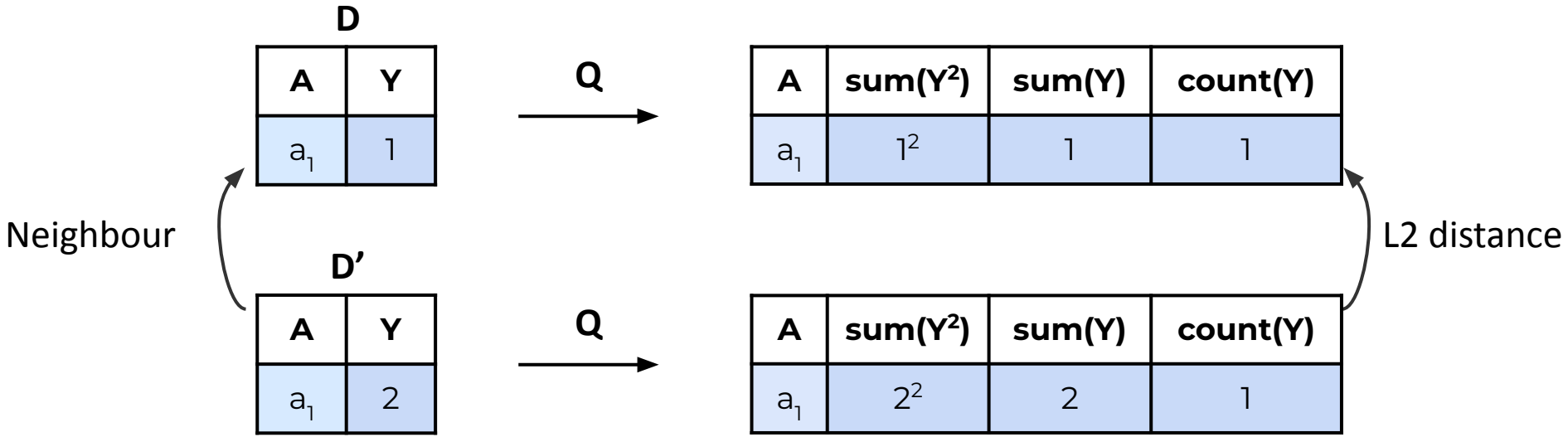
Use existing DP query engine



Q: SELECT SUM(**Y**^{2}), SUM(**Y**), COUNT(**Y**) from **D** GROUP BY **A**

FPM:Privatize sketches with privacy budget (ϵ, δ)

Use existing DP query engine



Sensitivity of Q: $\Delta(\mathbf{Q}) = \|\mathbf{Q}(\mathbf{D}) - \mathbf{Q}(\mathbf{D'})\|_2$

FPM:Privatize sketches with privacy budget (ϵ, δ)

Use existing DP query engine

D

A	Y
a_1	1

Q →

A	sum(Y^2)	sum(Y)	count(Y)
a_1	1^2	1	1

$$\mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}\Delta(Q)}{\epsilon}\right)$$

Budget

Q: SELECT SUM(**Y**^{2}), SUM(**Y**), COUNT(**Y**) from **D** GROUP BY **A**

FPM:Privatize sketches with privacy budget (ϵ, δ)

Use existing DP query engine

D

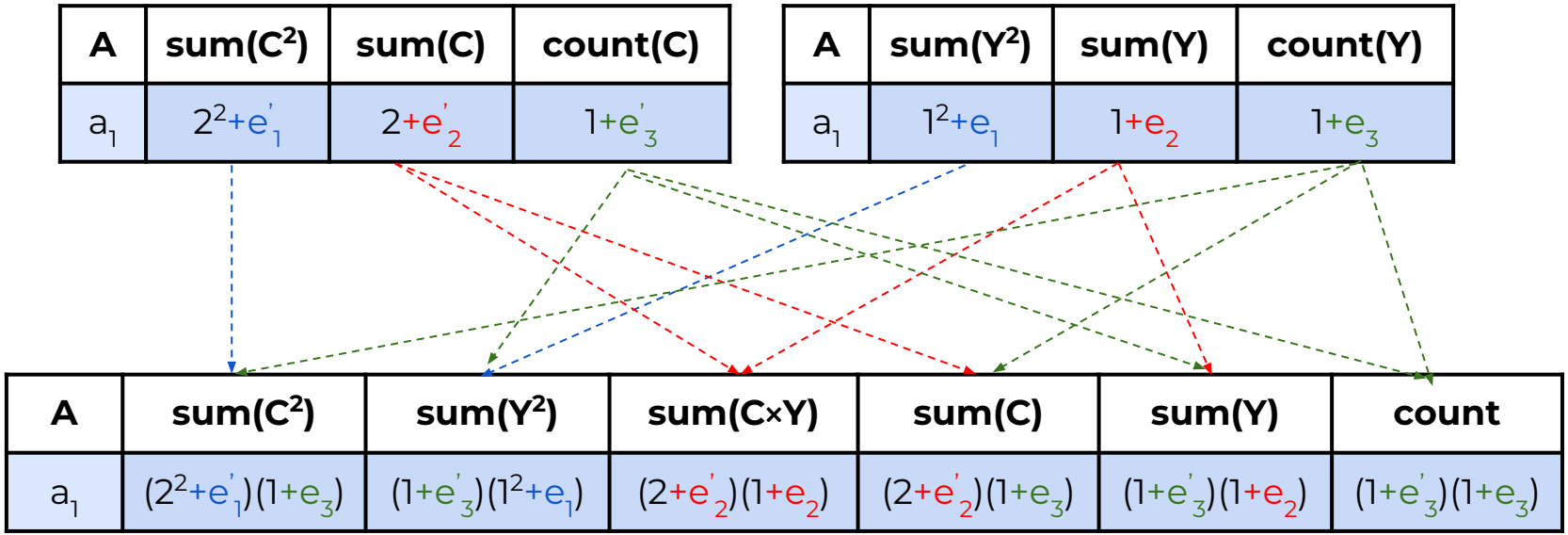
A	Y
a_1	1

A	sum(Y^2)	sum(Y)	count(Y)
a_1	1^2+e_1	$1+e_2$	$1+e_3$

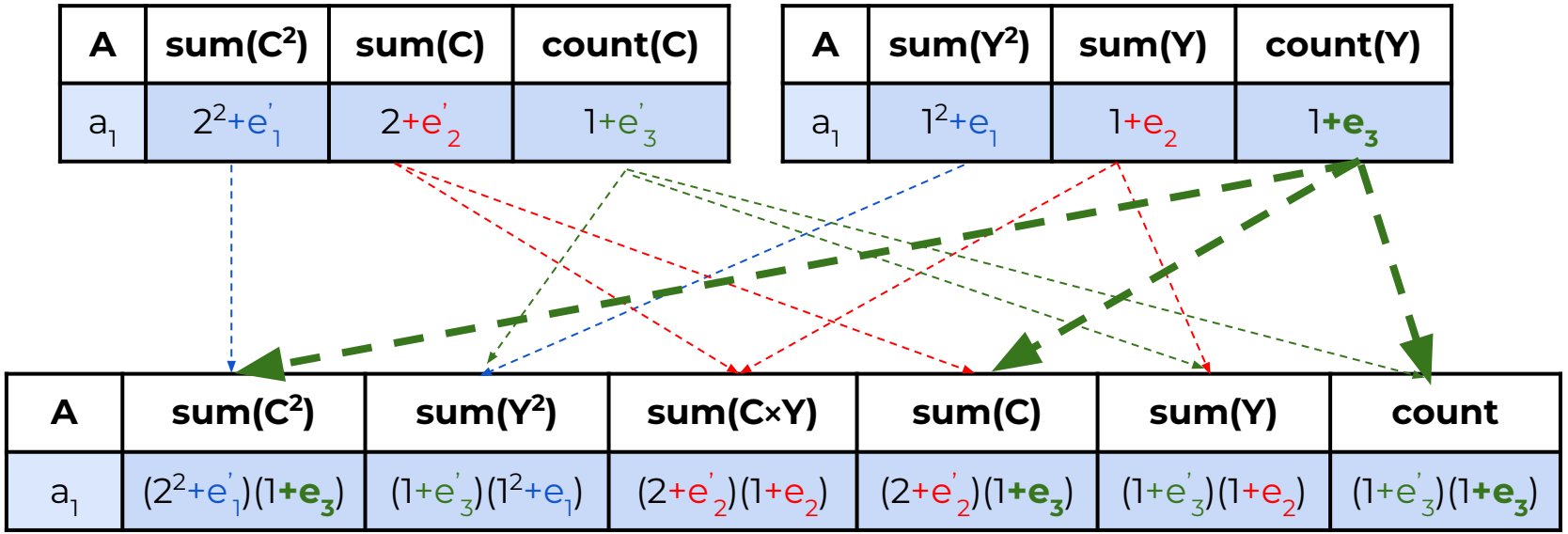
e_1, e_2, e_3 $\quad \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}\Delta(Q)}{\epsilon}\right)$

Q: SELECT SUM(**Y**^{2}), SUM(**Y**), COUNT(**Y**) from **D** GROUP BY **A**

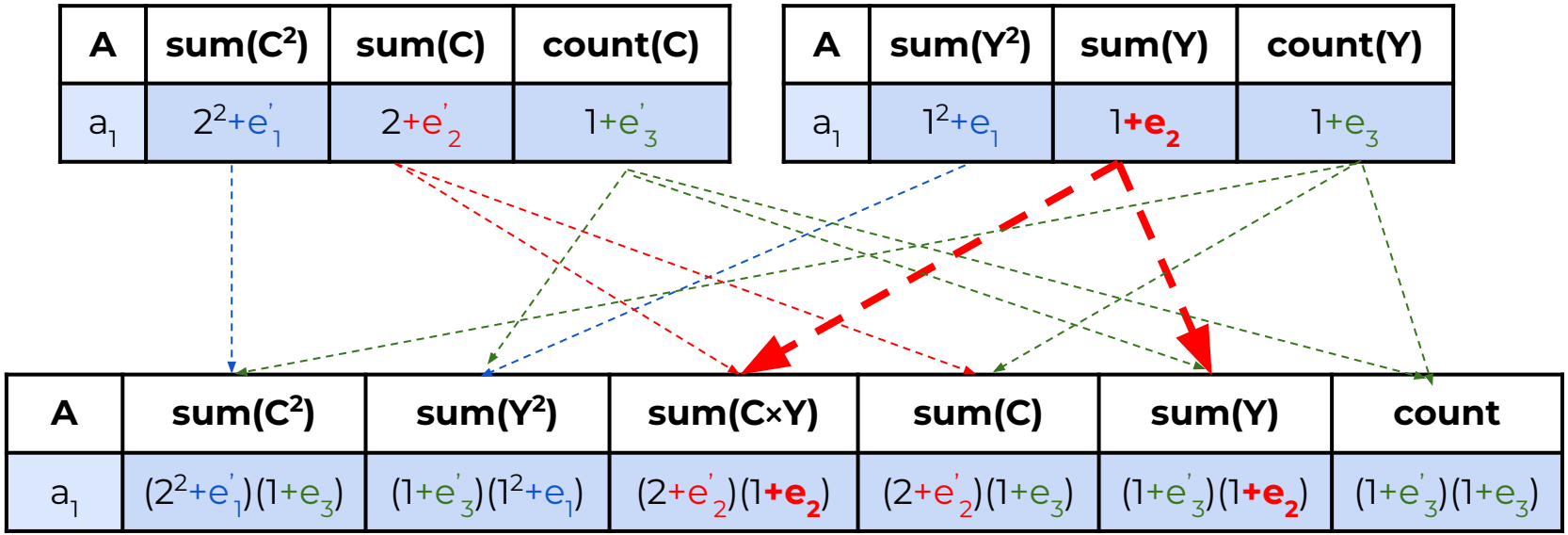
Naive Solution Limitation:Combining Sketches



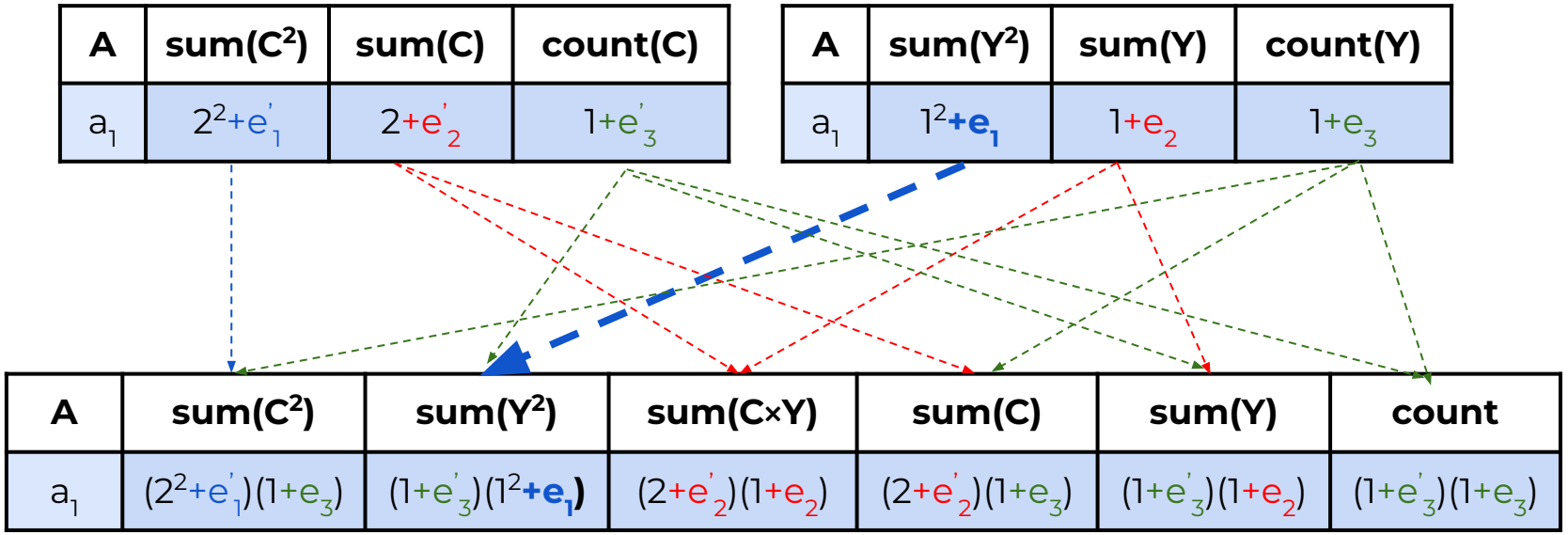
Naive Solution Limitation:Combining Sketches



Naive Solution Limitation:Combining Sketches



Naive Solution Limitation:Combining Sketches



Noise Allocation

How to draw noise from different distributions to aggregations?

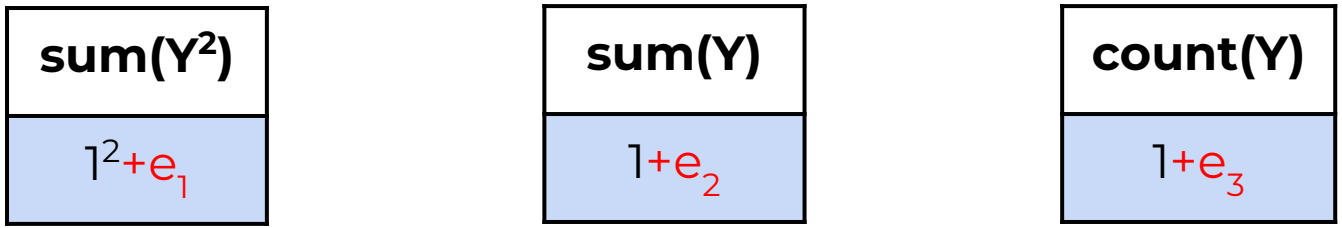
sum(Y^2)
1^2+e_1

sum(Y)
$1+e_2$

count(Y)
$1+e_3$

Noise Allocation

How to draw noise from different distributions to aggregations?



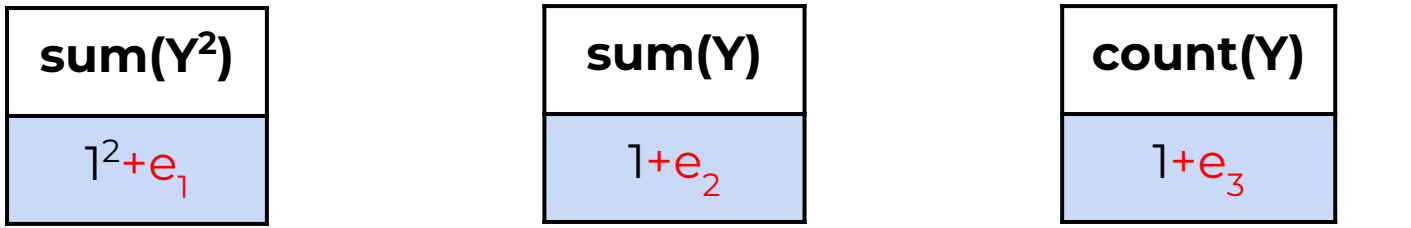
Sensitivity O(B^2)

Sensitivity O(B)

Sensitivity O(1)

Noise Allocation

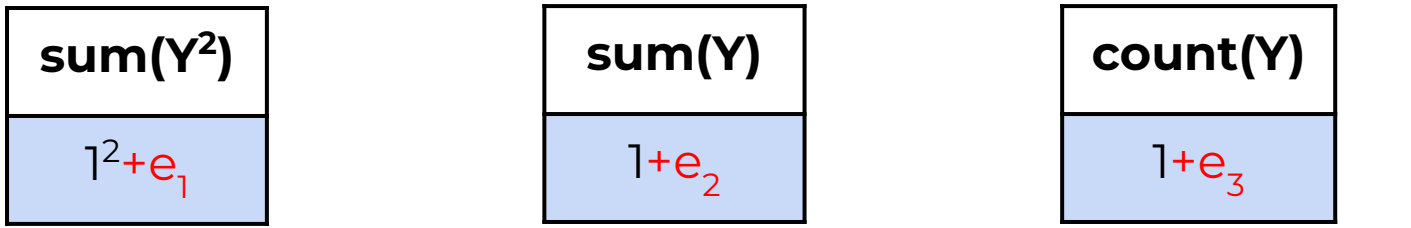
How to draw noise from different distributions to aggregations?



$e_1 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}B^2}{\epsilon_1}\right)$ $e_2 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}B}{\epsilon_2}\right)$ $e_3 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}}{\epsilon_3}\right)$

Noise Allocation

How to draw noise from different distributions to aggregations?



$$e_1 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}B^2}{\epsilon/3}\right) \qquad e_2 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}B}{\epsilon/3}\right) \qquad e_3 \sim \mathcal{N}\left(0, \frac{\sqrt{2\ln(1.25/\delta)}}{\epsilon/3}\right)$$

Noise Allocation: Analysis

Bounding linear regression estimator: $|\hat{\beta}_x - \tilde{\beta}_x| \leq \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)$

Noise Allocation: Analysis

Bounding linear regression estimator: $|\hat{\beta}_x - \tilde{\beta}_x| \leq \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)$

Naive Method: $\tau_1 = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}\right)$ $\tau_2 = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma_x^2}}\right)$

Noise Allocation: Analysis

Bounding linear regression estimator: $|\hat{\beta}_x - \tilde{\beta}_x| \leq \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)$

Naive Method: $\tau_1 = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}\right)$ $\tau_2 = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma_x^2}}\right)$

Optimization: $\tau_1 = O\left(\frac{B^2\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}\right), \tau_2 = O\left(\frac{B^2\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma_x^2}}\right)$

Noise Allocation: Analysis

Bounding linear regression estimator: $|\hat{\beta}_x - \tilde{\beta}_x| \leq \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)$

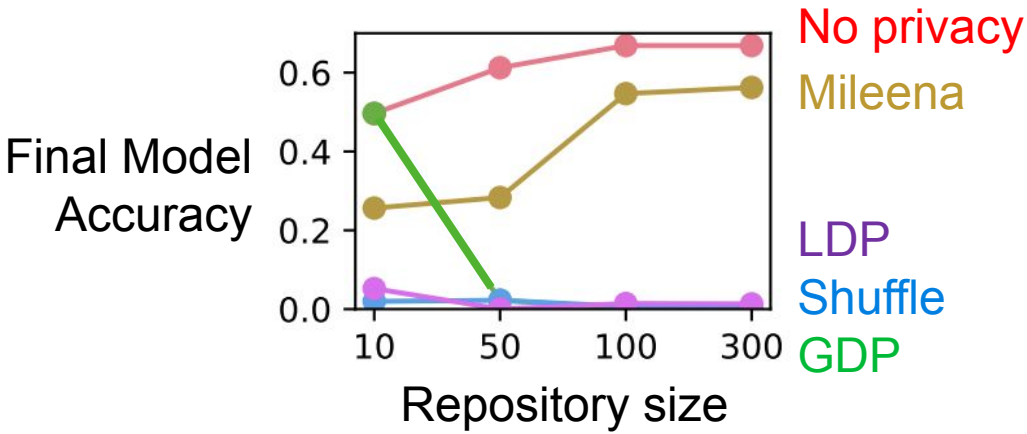
Naive Method: $\tau_1 = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}\right)$ $\tau_2 = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma_x^2}}\right)$

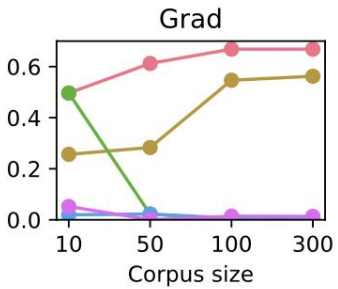
Optimization: $\tau_1 = O\left(\frac{B^2\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}\right), \tau_2 = O\left(\frac{B^2\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma_x^2}}\right)$

Reduce the bound on linear regression parameter by O(**B^2**)

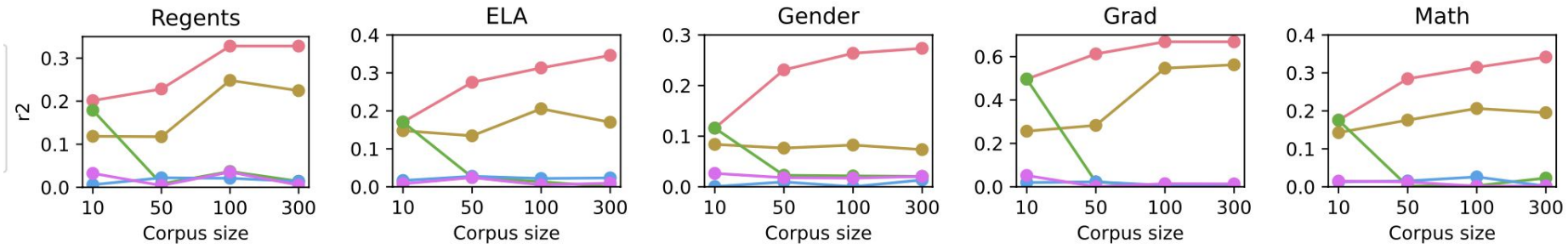
FPM Scales

To repository size & number of requests
Vary between 10 - 329 NYC Open Datasets in Repo





No privacy LDP Shuffle GDP Mileena



No privacy LDP Shuffle GDP Mileena